University of Zurich[UZH]

Department of Informatics

Integrating Advanced Machine Learning Methods Into Market Mechanisms

Dissertation submitted to the Faculty of Business,
Economics and Informatics
of the University of Zurich

to obtain the degree of
Doktor der Wissenschaften, Dr. sc.
(corresponds to Doctor of Science, PhD)

presented by
Jakob Weissteiner
from Austria

approved in February 2023

at the request of
Prof. Sven Seuken, Ph.D.
Prof. Craig Boutilier, Ph.D.
Prof. Benjamin Lubin, Ph.D.

The Faculty of Business, Economics and Informatics of the University of Zurich hereby authorizes the printing of this dissertation, without indicating an opinion of the views expressed in the work.

Zurich, February 15, 2023

The Chairperson of the Doctoral Board: Prof. Elaine Huang, Ph.D.

Abstract

Many economic problems require finding an efficient combinatorial assignment of multiple indivisible items to multiple agents. Popular examples include *combinatorial auctions (CAs)*, where a set of heterogeneous items is allocated among a set of bidders, and *combinatorial course allocation*, where course seats are allocated among students at universities.

What all of these domains have in common is that the agents can report their values on *bundles* (sometimes also called *packages*) of items rather than only on individual items. This allows them to express more complex preferences, i.e., an agent's value of a bundle is not simply the sum of each individual item's value, but it can be more (*complementarity*) or less (*substitutability*). A mechanism that allows agents to report values for *bundles* rather than just for individual items can achieve significantly higher efficiency. However, since the bundle space grows exponentially in the number of items, agents cannot report values for all bundles, even in domains with a modest number of items. Therefore, the key challenge in combinatorial assignment is the design of a *preference elicitation* algorithm that is (i) *practically feasible* with respect to elicitation costs and (ii) *smart*, i.e., it should elicit the information that is "most useful" for achieving high efficiency. In this thesis, I study how to design such smart and practically feasible preference elicitation algorithms using various machine learning (ML) algorithms.

In Research Question 1, I study how to use *neural networks (NNs)* and *Fourier transforms (FTs) for set functions* to enhance existing *value query*-based preference elicitation algorithms in combinatorial auctions (CAs). NNs enable the auctioneer to learn more complex bidder preferences than prior approaches; via FTs an auctioneer can leverage Fourier sparsity of bidders' preferences to reduce dimensionality and facilitate the learning task. Based on that, I introduce two novel *value query*-based ML-powered iterative CA mechanisms (NN-ICA and Hybrid-ICA) and experimentally show that they achieve state-of-the-art efficiency.

In Research Question 2, I study how to quantify posterior model uncertainty (i.e., epistemic uncertainty) for NNs in regression. Specifically, I propose *neural optimization-based model uncertainty (NOMU)*. In contrast to a fully Bayesian approach, NOMU *directly* estimates posterior model uncertainty (without explicitly defining a prior) by enforcing five important desiderata that any method should satisfy. I experimentally show that NOMU works particularly well in settings with zero or small data noise and scarce training data. This and the fact that NOMU's posterior model uncertainty estimate can be represented by a *single* NN in contrast to ensemble methods, makes NOMU particularly well suited for preference elicitation in combinatorial assignment.

In Research Question 3, I frame the design of an iterative combinatorial assignment mechanism as a combinatorial Bayesian optimization (BO) task with an expensive-to-evaluate function. In classic BO a well-suited domain-specific prior and an acquisition function based on a notion of uncertainty are key. To address the domain-specific prior, I propose a new class of NNs: *monotone-value neural networks (MVNNs)*. MVNNs are specifically designed to model monotone combinatorial value functions. I experimentally show that incorporating this important prior knowledge leads to better generalization performance, specifically in settings with few training data points. Regarding a notion of uncertainty over agents' preferences, I combine NOMU and MVNNs to define a new method for estimating an *upper uncertainty bound (uUB)*

that can then be used to define an acquisition function to determine the next query. This results in the design of a value query-based *Bayesian optimization-based combinatorial assignment (BOCA)* mechanism that uses MVNNs as domain-specific prior and makes use of NOMU's posterior model uncertainty in its query generation module. I experimentally evaluate BOCA and show that not only exploiting but also properly exploring the bundle space during the preference elicitation phase indeed leads to allocations with higher efficiency.

In Research Question 4, I study the course allocation problem, an instance of combinatorial assignment where monetary transfers are not permitted. I introduce a machine learning-powered course allocation mechanism. Concretely, I extend the state-of-the-art *Course Match* mechanism with an MVNN-based preference elicitation algorithm. I call the proposed mechanism *machine learning-powered course match (MLCM)*. MLCM generates in an iterative, asynchronous manner, *pairwise comparison queries* that are tailored to each individual student. I perform computational experiments using a simulator for students' preferences that was fitted to real-world data. I find that, compared to *Course Match*, MLCM is able to significantly increase both average and minimum student utility, even with only ten additional pairwise comparison queries.

Acknowledgements

First and foremost, I would like to thank my advisor Sven Seuken for guiding me trough the past four years of my PhD. When I look back on myself to when I started my PhD and we wrote our first paper together, and compare that to the present me, it is truly remarkable how many different things I was able to learn from him. Sven's dedication and passion for our research projects has constantly motivated me to keep going even during more difficult times. Specifically, I thank Sven for teaching me how to communicate and present my work and for giving me the freedom to select my own research topics and collaborators. His constant support, sheer availability and careful advice did not end at research but he continued to be of great help beyond and ultimately shaped the person I am today. Thank you!

I deeply thank everyone with whom I had the pleasure to collaborate. This thesis would certainly not have been possible without you. I thank Hanna Wutte and Jakob Heiss from the Department of Mathematics at ETH, with whom I share a very special and long-lasting relationship since day one of our Bachelor's studies in mathematics back in Vienna in 2012. I would like to thank you both for our countless (fun) meetings and interesting discussions in the last 10+ years on all kinds of different topics including fruitful research discussion. However, more importantly, I thank you for your close friendship in all those years. I would like to thank Chris Wendler and Markus Püschel from the Advanced Computing Laboratory at ETH and Benjamin Lubin from the Questrom School of Business at Boston University for the amazing interdisciplinary work we have done together and for being so patient with the research project of combining Fourier analysis and auctions. Specifically, I would like to thank Chris for not only being a great collaborator but also for becoming a close friend since we started our project. I would like to thank Josef Teichmann from the Department of Mathematics at ETH for our project on uncertainty quantification for neural networks and for organizing many get-togethers and pizza dinners, which helped me to socialize and meet many interesting people over the past years in Zurich. I would like to thank Julien Siems for being a great collaborator, pytorch specialist and friend since we started our first research project in 2021. Luckily, at some point we met in person and not only virtually when he showed me around in Berlin, thanks! Finally, I would like to thank my most recent collaborators Behnoosh Zamanlooy and Ermis Soumalias from the Computation and Economics Research Group at the University of Zurich, with whom I not only had the pleasure to work together on the course allocation project but with whom I also enjoyed many dinner evenings, hikes, and at least some –I was pushing for more– meetups in the ASVZ sport centers.

I thank Craig Boutilier for providing extremely valuable feedback on my PhD proposal and for serving as an external reviewer. Moreover, I also thank Benjamin Lubin for serving as an additional external reviewer. I am honored to have both of them on my dissertation committee.

I thank my colleagues from the Computation and Economics Research Group at the University of Zurich for their company, support and many interesting discussions over the past years: Dmitry Moor, Steffen Schuldenzucker, Gianluca Brero, Ludwig Dierks, Vitor Bosshard, Nils Olberg, Paul Friedrich, Behnoosh Zamanlooy and Ermis Soumalias. Specifically, I would like to thank Gianluca for introducing me to machine learning-based market design. Moreover, I would like to thank Nils for sharing an office with me over many years, for always having an open ear,

and for countless tennis matches.

I thank Fabio Isler for providing excellent support on the spectrum auction test suite. Furthermore, I would like to thank the students I had the pleasure to supervise: Marius Högger and Aurelio Dolfini.

I would like to thank my family and friends for their infinite love and unwavering support that I have received over the past years. Without you I certainly would not be the person I am today. First and foremost, I thank my parents Andreas and Monika Weissteiner for their selflessness and devotion to my well-being. In particular, thank you that you enabled me to pursue a career in research in the very first place starting with my Bachelor studies in Vienna. Moreover, I thank my brother Matthias Weissteiner, my sister Theresa Grafeneder-Weissteiner and her family Peter Grafeneder, Sophie Grafeneder, and Nikolaus Grafeneder, for their constant presence and encouragement. You all are an invaluable part of my life. I would like to also thank my friends Christoph Grabmer, Heinz Raab and Matthias Waltenberger, with whom I regularly enjoyed going on vacation to relax from stressful periods. Furthermore, I thank all family members from Hanna Wutte's side: Max Wutte, Antonia Wutte, Stefanie Klünsner, Nora Palfner-Wutte, Michael Palfner, Maximilian Palfner, Moritz Palfner, Eva Wutte, Gert Jerneischek, for accompanying me on my journey during the past decade.

Finally, I would like to thank the most important person in my life, my love Hanna Sophia Wutte. I quickly realized that it is impossible for me to acknowledge here in a few lines all what you have done for me in the past years and what you really mean to me. However, one thing I definitely wanted to let you know: thank you for being so understanding and patient with me!

To Hanna Sophia

Contents

1 Introduction[1]

In recent years, artificial intelligence (AI) and machine learning (ML) in particular have found widespread application in many real-world market mechanisms (with and without money). This ranges from the development of sophisticated ML-based recommender systems on popular platforms like TikTok, Instagram, Netflix, to the use of natural language processing to predict a seller's quality on eBay (Masterov et al., 2015), to ML-based optimal reserve prices in Google's Ad auctions (Milgrom and Tadelis, 2019), or to optimal refugee resettlement matching mechanisms via gradient boosting regression trees (Bansak et al., 2018). In all these examples, ML has helped greatly to *design a better marketplace*, e.g., by improving customers' experiences, by facilitating trades, by increasing a seller's revenue or by achieving better societal outcomes.

In my thesis, I focus on *combinatorial assignment*. Many important economic problems involve the combinatorial assignment of multiple indivisible items to multiple agents. In domains *with money*, prominent examples include *combinatorial auctions (CAs)* and *combinatorial exchanges (CEs)*. In CAs, heterogeneous items are allocated amongst a set of bidders, e.g., for the sale of spectrum licenses (Cramton, 2013). In CEs, a set of items is allocated between multiple agents who can be sellers *and* buyers at the same time, e.g., for the reallocation of catch shares (Bichler et al., 2019). In domains *without money*, a popular example is *combinatorial course allocation*, where course seats are allocated to students in business schools (Budish, 2011).

What all of these domains have in common is that the agents can report their values on *bundles* of items rather than only on individual items. This allows them to express more complex preferences, i.e., their value for a bundle is not just the sum of each individual item's value, but it can be more (*complementarity*) or less (*substitutability*). A mechanism that allows agents to report values for bundles rather than just for individual items can achieve significantly higher efficiency. However, this also implies that agents' preferences are exponentially-sized (i.e., for m items there are 2^m different bundles), and thus agents cannot report values for all bundles, even in settings with a modest number of items. Therefore, the key challenge in combinatorial assignment is the design of a *preference elicitation* algorithm that is (i) *practically feasible* with respect to elicitation costs and (ii) *smart*, i.e., it should elicit the information that is "most useful" for achieving high efficiency.

In my thesis, I study how to use ML for the design of preference elicitation algorithms in combinatorial assignment. On a high level, the *ML-based preference elicitation* algorithms I consider in my thesis proceed iteratively in rounds and involve the following two key steps: First, to use ML algorithms to learn agents' preferences (including some notion of uncertainty) from observed data points; second, to use those trained ML models to generate smart and informative queries for the agents (I focus on *value queries*, i.e., *"How much do you value bundle*

[1]Some parts of this chapter are adapted from my own prior work (Weissteiner and Seuken, 2020; Heiss et al., 2022; Weissteiner et al., 2022b,a, 2023; Soumalias et al., 2023)

A" and *pairwise comparison queries*, i.e., *"Do you prefer bundle A or B?"*). Specifically, for the second step, a key requirement is to efficiently solve the resulting *ML-based combinatorial optimization problem.* For example, in combinatorial auctions, a key step is to solve ML-based winner determination problems (WDPs), i.e., finding allocations that maximize the predicted social welfare with respect to the trained ML models. In combinatorial course allocation, a key step is to determine for each student a feasible course schedule that maximizes the predicted utility with respect to the student's trained ML model.

1.1 Background, Problem Statements and Research Questions

In this section, I frame the four main research questions for my thesis. Each of those will address the overarching question:

How to design better market mechanisms by integrating advanced ML methods?

Machine Learning-based Combinatorial Auctions

First, I study the design of ML-based combinatorial auction (CA) mechanisms, that aim for approximately *efficient allocations*, i.e., allocations of items to bidders such that the total sum of bidders' values, i.e., *the social welfare*, is approximately maximized.

For general value functions, Nisan and Segal (2006) have shown that to guarantee full efficiency in CAs, exponential communication in the number of items is needed. Thus, practical CA designs cannot provide efficiency guarantees in large domains. Instead, recent proposals have focused on *iterative combinatorial auctions (ICAs)*, where the auctioneer interacts with bidders over multiple rounds, eliciting a *limited* amount of information, aiming to find a highly efficient allocation. Prior work by Brero et al. (2018, 2021), proposed the first *ML-powered ICA*. At the core of their design is an *ML-powered preference elicitation algorithm.* As part of their algorithm, they used kernelized support vector regressions (SVRs) to learn the highly nonlinear value functions of bidders. Recently, Brero et al. (2021) showed that their ML-based ICA achieves even higher efficiency than the state-of-the-art non-ML-based *combinatorial clock auction*, which is a widely-used CA mechanism that has already generated more than $20 billion in total revenue over the past years (Ausubel and Baranov, 2017). However, because of runtime complexity issues, Brero et al. (2018, 2021) focused on SVRs with less expressive linear and quadratic kernels. Note that a quadratic kernel, while more expressive than a linear kernel, can still at most model two-way interactions between the items. This leaves room for improvement, since bidders' value functions can have more complex structures than can be captured by linear or quadratic kernels and brings me to my first research question:

Research Question 1 *How can we enhance ML-based preference elicitation in CAs and address the limitations of prior work?*

Quantifying Model Uncertainty for Neural Networks

Before, I motivate my next research question, I first briefly review the definition of *model uncertainty (a.k.a. epistemic uncertainty)* and *data noise (a.k.a. aleatoric uncertainty)* (see (Heiss

et al., 2022, Section 2) for more details).

Remark 1 (Model Uncertainty versus Aleatoric Uncertainty) *Let $X \subset \mathbb{R}^d, Y \subset \mathbb{R}$ denote some input and output space and let $f\colon X \to Y$ denote the unknown ground truth function. Let $D^{train} := \{(x_i^{train}, y_i^{train}) \in X \times Y, i \in \{1, \dots, n^{train}\}\}$, with $n^{train} \in \mathbb{N}$ be i.i.d samples from the data generating process $y = f(x) + \varepsilon$, where $\varepsilon | x \sim \mathcal{N}(0, \sigma_n^2(x))$. Let σ_n refer to the* data noise (aleatoric uncertainty)*. I follow the classic Bayesian uncertainty framework by modelling the unknown ground truth function f as a random variable. Hence, with a slight abuse of notation, I use the symbol f to denote* both *the unknown ground truth function as well as the corresponding random variable. Given a prior distribution for f, known data noise σ_n, and training data points D^{train}, the posterior of f and y are well defined. The* model uncertainty (epistemic uncertainty) *$\sigma_f(x)$ is the posterior standard deviation of $f(x)$, i.e., $\sigma_f(x) := \sqrt{\mathbb{V}[f(x)|D^{train}, x]}$, $x \in X$. Assuming independence between f and ε, the variance of the predictive distribution of y can be decomposed as $\mathbb{V}[y|D^{train}, x] = \sigma_f^2(x) + \sigma_n^2(x)$. Intuitively, model uncertainty describes the uncertainty that arises from a lack of observed training data points while aleatoric uncertainty describes the irreducible uncertainty in the environment, which, in contrast to model uncertainty, cannot be reduced by observing more training data points.*

In my thesis, I mainly focus on neural networks (NNs) as the ML algorithm in the design of ML-based preference elicitation mechanisms. NNs are nowadays part of many state-of-the-art systems across different ML disciplines (LeCun et al., 2015). This trend has also propagated to the design of market mechanisms, where NNs are becoming an increasingly important tool, e.g., in auction design (Dütting et al., 2019; Weissteiner and Seuken, 2020; Rahme et al., 2020; Weissteiner et al., 2022b), in the design of combinatorial assignment mechanisms (Weissteiner et al., 2022a, 2023), or in the design of course allocation mechanisms (Soumalias et al., 2023). However, the increased popularity also demands for methods to quantify the posterior *model uncertainty (i.e., epistemic uncertainty)* of such employed NNs.

More concretely, in this section, I consider NNs $\mathcal{N}_i\colon X \to Y$ for learning the ground truth function f from D^{train}, i.e., to obtain an estimate $\hat{f}(x)$ for $\mathbb{E}[f(x)|D^{\text{train}}, x]$, and I am further interested to also obtain an estimate $\hat{\sigma}_f(x) \approx \sigma_f(x)$ via techniques that are based on NNs.

In applications such as autonomous driving or automated passport control, overconfident predictions can even be dangerous (Amodei et al., 2016). Furthermore, good estimates of posterior model uncertainty are important for the design of smart preference elicitation mechanisms, where exploration is steered by (functions of) these posterior model uncertainty estimates.

Specifically, in combinatorial assignment it is often important to quantify posterior model uncertainty in *scarce* and *small data noise* settings.

Remark 2 (Scarcity) *Scarcity comes from the fact that in the combinatorial assignment settings I consider in this thesis, i.e., combinatorial auctions and combinatorial course allocation, agents typically can only report their value for a small number of bundles relative to the full exponential bundle space.*

Remark 3 (Small Data Noise) *Small data noise relative to overall uncertainty can be justified as follows: First, it is important to note that in all combinatorial assignment domains I consider, one is only interested in posterior model uncertainty (of the true value function)*

and not in posterior predictive uncertainty (uncertainty with respect to the response variable), i.e., for a data generating process $y_j = v(x_j) + \varepsilon(x_j)$ *with* $\varepsilon(\cdot)$ *representing the heteroskedastic data noise term, one only tries to capture the uncertainty corresponding to an agent's true value function* v*, since the goal is to maximize expressions based on the true agents' values. However, the posterior model uncertainty of* v *entails the following two sources of uncertainty: (i)* non-data-noise-induced posterior model uncertainty *and (ii)* data-noise-induced posterior model uncertainty*.*

In general, in scarce settings, typically non-data-noise-induced posterior model uncertainty for not yet elicited bundles dominates the corresponding data-noise-induced posterior model uncertainty, i.e., out-of-sample uncertainty is larger than in-sample uncertainty.

Moreover, in spectrum auctions, the most prominent application of combinatorial auctions, agents' reports are typically assumed to be observed with small data noise, i.e., network operators can quite precisely value bundles they are offered by employing expert teams, such that for not yet elicited bundles non-data-noise-induced posterior model uncertainty is even more dominant than data-noise-induced posterior model uncertainty.[2]

In the course allocation domain, students usually make more significant mistakes when answering queries (Budish and Kessler, 2022). Thus, to capture such reporting mistakes in course allocation, I propose a reporting mistake simulation scheme (Soumalias et al., 2023, Section 4.2) that was fitted to real-world data (Budish and Kessler, 2022). Nonetheless, even in the course allocation domain where data noise is potentially larger, the domain scarcity typically implies that for not yet elicited bundles non-data-noise-induced posterior model uncertainty is still the dominant source of overall uncertainty.

All considered ML algorithms that provide point predictions (i.e., NNs, Fourier transforms, monotone-value neural networks), can in general also be used in settings with larger data noise by adapting their hyperparameters, e.g., incorporating explicit and implicit regularization. Nevertheless, my proposed uncertainty quantification method for NNs, i.e., neural optimization-based model uncertainty (NOMU)*, is specifically designed for settings with small data noise. However, I also evaluate NOMU in settings with data noise (Heiss et al., 2022, Section 4.1.4) and showed that it performs on par with other considered benchmarks. Furthermore, in (Heiss et al., 2022, Appendix C.1), I provide an extension on how to incorporate a data noise estimation in NOMU. Its evaluation is postponed to future work.*

However, estimating posterior model uncertainty well in a computationally efficient way for NNs is still an open research problem. Especially for such settings with scarce training data and negligible data noise, where posterior model uncertainty is the main source of uncertainty, popular state-of-the-art methods show severe deficiencies (Heiss et al., 2022). Moreover, an important step for the design of an ML-based preference elicitation mechanism in combinatorial assignment is to find the *predicted* social welfare-maximizing allocation, i.e., solving the *ML-*

[2] Even in the case of large data noise, agents' reports are typically legally binding in the considered mechanisms, such that the mechanism's trained ML models should *exactly* fit the reports, i.e., it could be particularly problematic if an agent reports for a bundle that her value is 10$ but the corresponding ML model predicts 5$ and subsequently the agent accuses the mechanism designer of falsely representing her reports. Overall, instead of explicitly modelling that reports are noisy to address the imprecision of agents' reports in spectrum auctions, it would be better to allow agents to report intervals instead of single points to the mechanism (see Beyeler et al. (2021) for an extension of MLCA to interval reports). Future work could combine the interval method by Beyeler et al. (2021) with my proposed ML algorithms.

based winner determination problem (WDP). This is often achieved by reformulating the ML-based WDP into a mixed integer linear program (MILP). Thus, to integrate posterior model uncertainty estimates, one needs to be able to represent these estimates in a functional form that can be encoded as a succinct, practically feasible MILP. In this context my second research question is:

Research Question 2 *How can we capture model uncertainty for NNs for scarce regression settings with small data noise and ensure that the resulting model uncertainty estimates can be encoded as a MILP?*

Bayesian Optimization-based Combinatorial Assignment

In an (ML-based) iterative combinatorial assignment mechanism (e.g., an iterative combinatorial auction), the central agency (e.g., the auctioneer) interacts iteratively with agents over multiple rounds, eliciting a *limited* amount of information (e.g., via value queries), aiming to find a highly efficient allocation.

Thus, it is apparent that these mechanisms can be seen as a Bayesian optimization (BO) maximization task with an expensive-to-evaluate function (Frazier, 2018): First, the central agency's (e.g., an auctioneer) objective is to find a highly efficient allocation, i.e., to maximize the social welfare function. Second, this objective in general lacks known structure and when evaluating it (e.g., via value queries) one only observes it at a single input point and without derivatives such that gradient-based optimization cannot be used. Third, in practice, one can only elicit a very limited amount of information (expensive-to-evaluate) to find an approximately optimal allocation. For example, in a real-world spectrum auction, the auctioneer could only ask each bidder to answer on the order of hundreds of value queries for different bundles, even though the space of possible bundles is exponential in the number of items m, i.e., there are 2^m possible bundles and $(n+1)^m$ possible allocations for n bidders. However, in addition to the challenges that arise in classic BO, the combinatorial nature in combinatorial assignment adds its own set of challenges. For example, Gaussian process-based BO often does not extend beyond 10–20 input dimensions, which is problematic as in combinatorial assignment the input space can be much larger, e.g., in the multi-region value model (Weiss et al., 2017) for $m = 98$ items and $n = 10$ bidder the input space is 980-dimensional. In addition, integrality constraints to obtain only whole items (i.e., combinatorial assignment deals with assigning m *indivisible* items to agents) and feasibility constraints that ensure each item is only allocated once also need to be enforced.

Overall, in (combinatorial) BO a well-suited *domain-specific prior* and an acquisition function based on a *notion of uncertainty* are key. For example, in many combinatorial assignment settings, agents can freely dispose of unwanted items. Thus, a common assumption about agents' value functions is monotonocity (a.k.a. *free disposal* in CAs), i.e., *"additional items increase value"*. However, prior work on ML-based iterative combinatorial assignment (Weissteiner and Seuken, 2020; Brero et al., 2021; Weissteiner et al., 2022b) has neither taken this important monotonicity property into account nor did they integrate a notion of uncertainty into their mechanisms. Specifically, the latter means that these approaches are *myopic* in the sense that the resulting mechanisms simply query the allocation with the highest predicted welfare. In

particular, the mechanisms do not have any model of *uncertainty* over bidders' values for not yet elicited bundles, although handling uncertainty in a principled manner is one of the key requirements when designing BO algorithms. Thus, these mechanisms cannot properly control the *exploration-exploitation trade-off* inherent to BO. Concretely, this can imply that these mechanisms may get stuck in local minima, repeatedly querying one part of the allocation space while not exploring other, potentially more efficient allocations. This brings me to my third research question:

Research Question 3 *How can we perform Bayesian optimization in combinatorial assignment via domain-specific NNs as prior?*

ML-based Combinatorial Course Allocation

Finally, I study *course allocation*. The course allocation problem arises when educational institutions assign bundles of courses to students (Budish and Cantillon, 2012). Each course has a limited number of seats, seats are indivisible, and monetary transfers are not permitted for fairness reasons. What makes this problem particularly challenging is that students' preferences over bundles of courses exhibit a combinatorial structure because students may view certain courses as complements or substitutes.

Popular mechanisms traditionally used in course allocation were the *Harvard Business School (HBS) mechanism* and a *bidding points auction (BPA)* with artificial currency. However, both of these approaches have severe design flaws that significantly impact the efficiency of the final allocation: For HBS, there are significant opportunities for students to strategically misreport their preferences for courses (Budish and Cantillon, 2012) and BPA treats the artificial currency as if it were real assuming that students have value for left-over currency. To address these design flaws, Budish (2011) proposed a new course allocation mechanism *A-CEEI: An approximation to competitive equilibrium from equal incomes* that is approximately efficient, satisfies two fairness criteria and is strategyproof in the large (i.e., if enough students participate, it is optimal for them to report their true preferences). While attractive in theory, A-CEEI assumes students can *exactly* report their *full* preferences to the mechanism.

In response to that, Budish et al. (2017) proposed the *Course Match (CM)* mechanism, a practically feasible implementation of the A-CEEI mechanism. CM uses a simple reporting language to elicit students' preferences over *schedules* (i.e., course bundles). Concretely, CM offers students a graphical user interface to enter a *base value* between 0 and 100 for each course, and an *adjustment value* between -200 and 200 for each *pair* of courses. These adjustments allow students to report complementarities and substitutabilities between courses, up to *pairwise* interactions. The total value of a schedule is then the sum of the base values reported for each course in that schedule plus any adjustments (if both courses are in the schedule). CM provides a good trade-off between efficiency, fairness, and incentives and has now been adopted in many universities such as the Wharton School at the University of Pennsylvania and Columbia Business School.

However, already Budish et al. (2017) were concerned that the CM language may not be able to fully capture every student's preferences. Furthermore, they mentioned that some students might find it non-trivial to use the CM language and might therefore make mistakes when

reporting their preferences. Indeed, the field experiment by Budish and Kessler (2022) revealed several shortcomings of CM in this regard. First, students made very limited use of the CM language: on average, students only reported a base value for half of the 25 courses in the experiment. Furthermore, the average number of pairwise adjustments was only 1.08 (out of 300). This suggests that cognitive limitations negatively affect how well students can report their preferences using the CM language. Second, in addition to not reporting part of their preferences, students are also *inaccurate* when they do report their preferences. Budish and Kessler (2022) found that both of these reporting mistakes negatively affected the welfare of CM. In the experiment, about 16% of students would have preferred another schedule of courses, with a median utility difference for these schedules of 13%. Thus, preference elicitation in course allocation still remains an important challenge. In this context my fourth research question is:

Research Question 4 *How can we design an ML-based preference elicitation mechanism for course allocation, that enables students to express in a simple way more complex preferences and is more robust to errors, ultimately leading to better allocations?*

1.2 Further Related Work

1.2.1 Preference Elicitation in Combinatorial Auctions

Preference elicitation in combinatorial auctions (CAs) has been extensively studied in the past (see Sandholm and Boutilier (2006) for an overview). Conen and Sandholm (2001) proposed *constraint networks*, a general framework for how to represent a bidder's incompletely specified value function which can be updated with various different query types. Using constraint networks, Conen and Sandholm (2001, 2002, 2003) introduced a series of different *rank lattice-based* elicitation (i.e., search) algorithms that provably find Pareto optimal or efficient allocations in unstructured settings (i.e., without assumptions on bidders' value functions), relying on rank, value and bound queries. Similarly, Hudson and Sandholm (2004) proposed several elicitation policies in unstructured settings that are restricted only to *value queries*. Moreover, Hudson and Sandholm (2004) also studied elicitation policies that alternate between value and *order queries* and finally they introduced the use of *bound-approximation queries* for CAs, where bidders are asked to tighten their lower and upper bounds on their value for a given bundle.

However, the practical effectiveness of all the aforementioned algorithms is rather limited, i.e., they often need to elicit huge portions of the bundle space to guarantee an optimal allocation. In contrast, in my thesis I study the design of ML-powered preference elicitation mechanisms for CA that aim only for *approximately* efficient allocations but are practically feasible with respect to elicitation costs, even for larger settings with many items and bidders.

Using demand queries for preference elicitation in CAs for unstructured settings was most intensively studied in the context of ascending CAs (Parkes, 1999; Wurman and Wellman, 2000; Kwasnica et al., 2005). There exist ascending discriminatory bundle-price CAs that yield efficient allocations. These formats are based on primal-dual (de Vries et al., 2007) or subgradient algorithms (Parkes and Ungar, 2000; Ausubel and Milgrom, 2002). Finally, Blum et al. (2004) and Lahaie and Parkes (2004) showed that bundle-price queries have more power than item-price queries. In contrast, in the part of my research on ML-powered preference elicitation

mechanisms for CAs, I only use *value queries*.

Furthermore, there has also been considerable research for preference elicitation in CAs in *structured* domains, i.e., when restricting bidders' value functions to certain function classes. Zinkevich et al. (2003), Conitzer et al. (2005), and Santi et al. (2004) introduced combinatorial value function classes that can be exactly learned via value queries in *polynomial* time. Moreover, Blum et al. (2004) showed that while learning certain value function classes via value queries is hard (i.e., requiring super-polynomially many value queries), an efficient allocation can be determined with only polynomially many value queries. In contrast, in my thesis I first study in Research Question 1 ML-powered preference elicitation in CAs for general value functions, i.e., making no assumptions on bidders' value functions and then assume in Research Question 3 and Research Question 4 that bidders' value functions are monotone (i.e., fulfill *free disposal*), a relatively mild assumption that is fulfilled in most CA settings.

As in unstructured domains, most research on demand queries in structured domains has been conducted in the context of ascending CAs, more concretely, in settings where ascending item-price auctions yield efficient allocations. For example, Kelso Jr and Crawford (1982),Gul and Stacchetti (2000), Ausubel (2006), and Nisan and Segal (2006) proposed ascending CAs for substitute valuations.

1.2.2 Machine Learning-based Preference Elicitation and Mechanism Design

More recently, researchers have successfully used ML for preference elicitation. Early work by Blum et al. (2004) and Lahaie and Parkes (2004) laid the foundation for this by studying the relationship between computational learning theory and preference elicitation in CAs. As mentioned, most related to this thesis is the work by Brero et al. (2017), Brero et al. (2018), and Brero et al. (2021), who proposed a value query-based *ML-powered preference elicitation algorithm* for CAs. Beyeler et al. (2021) extended their approach by allowing bidders to only report upper and lower bounds on bundle values instead of value queries.

Using demand queries, Brero and Lahaie (2018) and Brero et al. (2019) introduced a Bayesian iterative combinatorial auction using probabilistic item-price updates. In contrast to the mechanisms I consider in my thesis, their approach does not maximize the per-instance efficiency but rather tries to clear (i.e., to find an efficient allocation) as many instances as possible. However, since in general item-prices are not expressive enough to support an efficient allocation, their current approach is rather limited in practice, i.e., even in relatively small synthetic settings (Combinatorial Auction Test Suite (CATS), 12 items and 10 bidders) their approach could only find market clearing prices, in around 60% of the cases.

In active learning, Chu and Ghahramani (2005) and Guo et al. (2010) use Gaussian processes (GPs) for preference elicitation. Specifically, Chu and Ghahramani (2005) use GPs for pairwise preference elicitation over instances as well as for the more general framework of label ranking (i.e, learning preference relations over the instances' set of labels instead over the instances) and Guo et al. (2010) use GPs and pairwise comparison queries that maximize the expected value of information (EVOI). However, in the settings considered in this thesis, GPs are less well suited to design a combinatorial assignment algorithm (due to the high dimensionality of the input space, the combinatorial, i.e., integrality and feasibility, constraints, and the computational challenge of quickly solving the ML-based WDP several hundreds of times.

While in my thesis I focus on the use of ML algorithms for preference elicitation, there has been also a considerable amount of research on using ML to learn whole mechanisms from data following the automated mechanism design paradigm. For auctions, Dütting et al. (2019), Shen et al. (2019), Rahme et al. (2020), and Peri et al. (2021) used NNs to find revenue optimal incentive compatible auction mechanisms, i.e., they use ML algorithms to learn the mapping from input bids to an allocation and payment rule. Moreover, ML-based automated mechanism design has also been studied in domains without money: Narasimhan et al. (2016) used support vector machines to design strategyproof mechanisms for social choice and stable mechanisms for two-sided matching problems. Golowich et al. (2018) used NNs to design strategyproof, multi-facility mechanisms that minimize expected social cost.

1.2.3 Encoding Neural Networks as Mixed Integer Linear Programs

In my thesis, I formulated the (MV)NN-based winner determination problem in combinatorial auctions and the NN-based utility maximization problem for a single student in course allocation as a mixed integer linear program (MILP). These MILP formulations are related to a recent line of research that uses MILP encodings of trained NNs for various tasks. For example, Cheng et al. (2017) studied resilience properties of trained NNs using a MILP, Fischetti and Jo (2018) used a MILP for finding adversarial examples of trained NNs in image recognition, Mladenov et al. (2017) considered linearized NNs as response models in logistic MDPs for user modeling in advertising and recommendation, and Say et al. (2017) used MILP formulations of NN-based transition models for planning problems. Finally, Anderson et al. (2020) provide a generic framework and overview of strong mixed integer programming formulations for trained NNs.

1.2.4 Quantifying Model Uncertainty for Neural Networks

Over the last decade, researchers have developed various methods to quantify posterior model uncertainty for NNs.[3] One strand of research considers Bayesian neural networks (BNNs), where distributions are placed over the NN's parameters (Graves, 2011; Blundell et al., 2015; Hernández-Lobato and Adams, 2015). However, variational methods approximating BNNs are usually computationally prohibitive and require careful hyperparameter tuning. Thus, BNNs are rarely used in practice (Wenzel et al., 2020a).

In practice, *ensemble methods* are more established. Gal and Ghahramani (2016) proposed *Monte Carlo dropout (MCDO)* to estimate posterior model uncertainty via stochastic forward passes. Interestingly, they could show that training an NN with dropout can also be interpreted as variational inference approximating a BNN. Lakshminarayanan et al. (2017) experimentally evaluated ensembles of NNs and showed that they perform as well as or even better than BNNs. They proposed using *deep ensembles (DE)*, which use NNs with two outputs for model prediction and data noise, and they estimate posterior model uncertainty via the empirical standard deviation of the ensemble. DE is the most established state-of-the art ensemble method in terms of robustness and uncertainty quantification, shown to consistently outperform other ensemble methods (Ovadia et al., 2019; Fort et al., 2019; Gustafsson et al., 2020; Ashukha et al., 2020). Recently, Wenzel et al. (2020b) proposed *hyper deep ensembles (HDE)*, an extension of DE

[3] Please, see Remark 1 for a brief review on *model uncertainty* and *data noise* in the classical Bayesian framework.

where additional diversity is created via different hyperparameters and experimentally showed that HDE outperforms DE in the experiments they considered.

Further lines of work have contributed to modeling uncertainties for NN-based estimates. Nix and Weigend (1994) were among the first to introduce NNs with two outputs: one for model prediction and one for *data noise (aleatoric uncertainty)*, using the Gaussian negative log-likelihood as loss function. However, such a data noise output cannot be used as an estimator for model (i.e., epistemic) uncertainty (see (Heiss et al., 2022, Appendix G) for a discussion). To additionally capture posterior model uncertainty, Kendall and Gal (2017) combined the idea of Nix and Weigend (1994) with MCDO.

Similarly, NNs with two outputs for lower and upper UBs, trained on specifically-designed loss functions, were previously considered by Khosravi et al. (2010) and Pearce et al. (2018). However, the method by Khosravi et al. (2010) again only accounts for data noise and does not consider posterior model uncertainty. The method by Pearce et al. (2018) also does not take posterior model uncertainty into account in the design of their loss function and only incorporates it via ensembles (as in DE).

Besides the state-of-the art ensemble methods HDE and DE, there exist many other papers on ensemble methods that, for example, promote the ensemble's diversity on the function space (Wang et al., 2019; Tiulpin and Blaschko, 2022) or reduce computational cost (Wen et al., 2020; Havasi et al., 2021).

For classification, Malinin and Gales (2018) introduced prior networks, which explicitly model in-sample and out-of-distribution uncertainty, where the latter is realized by minimizing the reverse KL-distance to a selected flat point-wise defined prior. In a recent working paper, Malinin et al. (2020) report on progress extending their idea to regression. While the idea of introducing a separate loss for learning posterior model uncertainty is related to NOMU, there are several important differences (loss, architecture, behavior of the model prediction, theoretical motivation) that are discussed in detail in (Heiss et al., 2022, Appendix E). Furthermore, their experiments suggest that DE still performs weakly better than their proposed method.

In contrast to BNNs, which perform approximate inference over the entire set of weights, neural linear models (NLMs) perform *exact* inference on only the last layer. NLMs have been extensively benchmarked in (Ober and Rasmussen, 2019) against MCDO and the method from (Blundell et al., 2015). Their results suggest that MCDO and (Blundell et al., 2015) perform competitively, even to carefully-tuned NLMs.

Neural processes, introduced by Garnelo et al. (2018a,b), have been used to express posterior model uncertainty for image completion tasks, where one has access to thousands of different images interpreted as functions f_i instead of input points x_i. See (Heiss et al., 2022, Appendix F) for a detailed comparison to NOMU.

Furthermore, it is important to distinguish the following two (almost) orthogonal problems regarding uncertainty quantification: First, the fundamental concept of how to estimate relative posterior model uncertainty, i.e., how much more posterior model uncertainty does one have at one point x compared to any other point x'? (methods such as MCDO, DE, HDE or NOMU are concerned with this question) and second, the calibration of the resulting uncertainty bounds, i.e., an $\alpha\%$ credible interval should contain the true outcome $\alpha\%$ of the time. For calibration, Kuleshov et al. (2018) and Kuleshov and Deshpande (2022) show that BNNs are in general badly

calibrated and propose effective calibration methods for any given uncertainty quantification method in regression (including NOMU) and classification.

1.2.5 Monotone Neural Networks

Several other approaches for incorporating monotonicity into NNs have previously been proposed. However, for these architectures, it is either not known how the NN-based winner determination problem (WDP) could be solved quickly, or they have other limitations: Sill (1997) proposes only a shallow architecture which violates the normalization property (i.e., $0 \mapsto 0$). You et al. (2017) propose a complicated non-standard architecture, where no computationally feasible MILP formulation of the corresponding WDP is known. Wehenkel and Louppe (2019) implement monotonicity by representing the target function as an integral of an NN and thus the WDP would result in a computationally infeasible MILP. Liu et al. (2020) train NNs with successively higher regularization until a MILP based verification procedure guarantees monotonicity. The repeated retraining and verification leads to high computational cost. In contrast, my proposed monotone-value neural networks (MVNNs) are particularly well suited for combinatorial assignment, because (i) the MVNN-based WDP can be formulated as a succinct MILP and thus solved quickly[4] and (ii) I propose a generic fully-connected feed-forward architecture with an arbitrary number of hidden layers which can be trained efficiently.

1.3 Publications Contained in this Thesis

This thesis consists of six papers that answer the four research questions presented in Section 1.1. In what follows, I restate the research questions and provide the corresponding list of papers that address each research question.

Research Question 1 *How can we enhance ML-based preference elicitation in CAs and address the limitations of prior work?*

Publications

1. Deep Learning-powered Iterative Combinatorial Auctions.
   Jakob Weissteiner and Sven Seuken.
   In *Proceedings of the Thirty-fourth AAAI Conference on Artificial Intelligence* (AAAI'20), New York, USA, February 2020.

2. Fourier Analysis-based Iterative Combinatorial Auctions.
   Jakob Weissteiner*, Chris Wendler*, Sven Seuken, Ben Lubin, and Markus Püschel.

[4] Specifically, note that the particular selection of bReLU as the activation function for MVNNs is in a certain sense even the optimal choice with respect to expressivity of the network and computational complexity of the corresponding MILP. This can be seen as follows: (i) The constraints on the weights and biases enforce monotonicity of MVNNs (in fact for *any* monotone activation). (ii) For universality, one needs however a *bounded* monotone non-constant activation, e.g., with ReLUs and our constraints one cannot express substitutabilities. (iii) for the MILP, one needs a *piecewise linear* activation, e.g., with sigmoids one could not formulate a MILP. Taking all together, bReLU is the simplest bounded, monotone, non-constant, piecewise-linear activation function.

*These authors contributed equally.

In *Proceedings of the Thirty-first International joint Conference on Artificial Intelligence* (IJCAI'22), Vienna, AUT, July 2022.

Research Question 2 *How can we capture model uncertainty for NNs for scarce regression settings with small data noise and ensure that the resulting model uncertainty estimates can be encoded as a MILP?*

Publications

3. NOMU: Neural Optimization-based Model Uncertainty.
   Jakob Heiss*, Jakob Weissteiner*, Hanna Wutte*, Sven Seuken, and Josef Teichmann.
   In *Proceedings of the Thirty-ninth International Conference on Machine Learning* (ICML'22), Baltimore, USA, July 2022.

Research Question 3 *How can we perform Bayesian optimization in combinatorial assignment via domain-specific NNs as prior?*

Publications

4. Monotone-Value Neural Networks: Exploiting Preference Monotonicity in Combinatorial Assignment.
   Jakob Weissteiner*, Jakob Heiss*, Julien Siems* and Sven Seuken.
   In *Proceedings of the Thirty-first International joint Conference on Artificial Intelligence* (IJCAI'22), Vienna, AUT, July 2022.

5. Bayesian Optimization-based Combinatorial Assignment.
   Jakob Weissteiner*, Jakob Heiss*, Julien Siems* and Sven Seuken.
   In *Proceedings of the Thirty-seventh AAAI Conference on Artificial Intelligence* (AAAI'23), Washington, D.C., USA, February 2023.

Research Question 4 *How can we design an ML-based preference elicitation mechanism for course allocation, that enables students to express in a simple way more complex preferences and is more robust to errors, ultimately leading to better allocations?*

Publications

6. Machine Learning-powered Course Allocation.
   Ermis Soumalias*, Behnoosh Zamanlooy*, Jakob Weissteiner and Sven Seuken.
   ArXiv preprint, March 2023, [pdf].

1.4 Summary of Contributions

In this section, I provide a summary of all six research papers and explain how they answer the four corresponding research questions.

*These authors contributed equally.

1.4.1 Deep Learning-powered Iterative Combinatorial Auctions

This paper provides the first answer to Research Question 1 by introducing a deep learning-powered iterative combinatorial auction (ICA), which outperforms state-of-the-art prior ICA designs with respect to allocative efficiency.

We build on prior work by Brero et al. (2018, 2021), who proposed a value query-based *ML-powered iterative combinatorial auction mechanism (MLCA)*[5]. At the core of their mechanism is an ML-powered preference elicitation algorithm, which consists of the following two steps: (1) an *estimation step*, where given some already elicited bundle-value pairs (i.e., bids) MLCA uses a distinct ML algorithm to learn each bidder's true value function and (2) an *optimization step*, where MLCA solves an *ML-based winner determination problem (WDP)* to determine the allocation with the highest predicted social welfare (i.e., using the trained ML-models as representations of bidders' true value functions), and MLCA then uses this allocation to generate the next set of value queries to all bidders. This process repeats in an iterative fashion until a fixed number of queries has been asked. As their ML algorithm, they used kernelized support vector regressions (SVRs) to learn the nonlinear value functions of bidders. Recently, Brero et al. (2021) experimentally showed on synthetic data generated via the *spectrum auction test suite (SATS)* (Weiss et al., 2017) that MLCA achieves even higher efficiency than the widely-used *combinatorial clock auction*.

However, because of runtime complexity issues of the *optimization step*, Brero et al. (2018, 2021) focused on SVRs with linear and quadratic kernels. This leaves room for improvement, since bidders' valuations can have more complex structures than can be captured by linear or quadratic kernels. Concretely, a quadratic kernel, while more expressive than a linear kernel, can still at most model *two-way (i.e., pairwise)* interactions between the items.

In this paper, we show how these shortcomings can be addressed by using fully-connected feed-forward neural networks (NNs) instead of SVRs in the *estimation step* and *the optimization step* of MLCA. In each round of the auction, we approximate bidders' value functions by NNs (*estimation step*) and subsequently solve an NN-based WDP (*optimization step*) to determine which queries to ask each bidder in the next round of the auction. NNs have the advantage that in contrast to SVRs, they do not use predefined feature transformations. While with SVRs, the choice of a good kernel usually relies on prior domain knowledge, NNs automatically learn features in the process of training. Moreover, they are more expressive than quadratic kernels, i.e., can model preferences that are more complex than two-way interactions. Since our design involves solving the NN-based WDP in each round of the auction, a key requirement for the practical implementation of our auction mechanism is to efficiently solve these NN-based WDPs. Therefore, we present a theorem, which shows how the NN-based WDP in the case of ReLU activation functions can be reformulated into a mixed integer linear program (MILP) (Weissteiner and Seuken, 2020, Theorem 1). Moreover, in contrast to SVRs with nonlinear kernels, one always obtains a MILP for the NN-based WDP whose size grows linearly in the number of bidders and items, for any number of layers and nodes.

To experimentally evaluate the performance of our NN-ICA, we follow prior work (Brero

[5] To be more specific, Section 1.4.1 builds on a predecessor version of MLCA, which was introduced by Brero et al. (2018) and which they called *pseudo VCG mechanism (PVM)*. However, for ease of exposition and since the main conceptual ideas of MLCA and PVM are the same, I only discuss MLCA at this point.

	GSVM 6 Regional (R), 1 National (N) Bidders				LSVM 5 Regional (R), 1 National (N) Bidders			
Auction Mechanism	**NN Hidden Layers**	**Efficiency in %**	**Revenue in %**	**t-test:Efficiency** $\mathcal{H}_0 : \mu_{\text{SVR}} = \mu_{\text{NN}}$	**NN Hidden Layers**	**Efficiency in %**	**Revenue in %**	**t-test:Efficiency** $\mathcal{H}_0 : \mu_{\text{NN}} \leq \mu_{\text{SVR}}$
NN-ICA	R:[32,32], N:[10,10]	98.63 ± 0.35	67.81	$p_{\text{VALUE}} = 0.3$	R:[32,32], N:[10,10,10]	97.74 ± 0.47	62.45	$p_{\text{VALUE}} = 2e^{-5}$
SVR-ICA	-	98.85 ± 0.25	77.80		-	96.03 ± 0.65	65.60	

Table 1.1: NN-ICA vs. SVR-ICA. All results are averaged on a test set of 100 auction instances. For efficiency, a 95%-confidence interval is shown. The winner based on a pairwise t-test with a significance level 0.05 is marked in grey. Additionally, we present the NN architectures (hidden layers) for the regional (R) and national (N) bidders.

et al., 2021) and use SATS to generate synthetic auction instances in two domains: the *Global Synergy Value Model (GSVM)*, and the *Local Synergy Value Model (LSVM)*. First, we compared the prediction (i.e., generalization) performance of NNs against SVRs, where we already observed that NNs better capture bidders' value functions (particularly in the more complex LSVM). Next, we compared the allocative efficiency of our NN-ICA against the SVR-ICA. The results are shown in Table 1.1. In GSVM (a domain perfectly suited for the quadratic kernel), NN-ICA matches the efficiency of the SVR-ICA, while in the more complex LSVM, NN-ICA outperforms SVR-ICA by 1.74% points. Finally, we also demonstrated that the NN-ICA scales well to a very large domain, by evaluating it in the *Multi Region Value Model (MRVM)* (with $m = 98$ items, 2^{98} bundles and $n = 10$ bidders).

This paper introduces a deep learning-powered ICA and provides the first answer to Research Question 1. Overall, our results show that, perhaps surprisingly, even small-sized NNs can be advantageous for the design of ICAs and thus we can conclude that using more expressive (deep) NNs for the design of preference elicitation mechanisms in CA leads to higher allocative efficiency.

1.4.2 Fourier Analysis-based Iterative Combinatorial Auctions

This paper completes the answer to Research Question 1 by introducing a Fourier analysis-based iterative combinatorial auction (ICA), which uses Fourier transforms (FTs) for set functions to leverage different notions of sparsity of value functions in preference elicitation.

Recent advances in Fourier analysis have brought new tools to efficiently represent and learn *set* functions (Stobbe and Krause, 2012; Amrollahi et al., 2019; Wendler et al., 2021). In this paper, we build on our own prior work (Weissteiner and Seuken, 2020) and bring the power of Fourier analysis to the design of combinatorial auctions (CAs). The goal of preference elicitation in CAs is to learn bidders' value functions using a small number of informative queries. Mathematically, value functions are *set* functions, i.e., they map a set of items represented as an indicator vector to a non-negative real number.

However, those set functions are in general exponentially large objects that are notoriously hard to represent or learn. To control for this complexity, we leverage Fourier analysis for set functions. In particular, we consider *Fourier-sparse approximations*, which are represented by a small number of parameters. These parameters are the non-zero *Fourier coefficients (FCs)* obtained by a base change with the *Fourier transform (FT)*. The motivation behind this approach is that we expect bidders' value functions to be *sparse*, meaning that their preferences can be

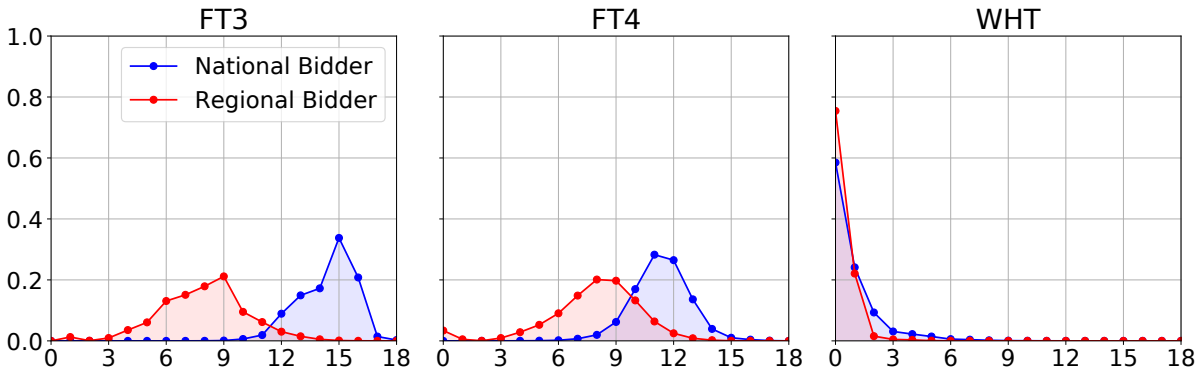


Figure 1.1: Spectral energy distribution in LSVM for three different FTs: FT3 (i.e., polynomial representation), FT4, and WHT. For each cardinality (x-axis), we collect the spectral energy (y-axis) of all FCs of that cardinality and normalize by the total spectral energy.

described with much less data than is contained in the (exponentially-sized) full value function. While this sparsity may be difficult to uncover when looking at bidders' value reports, it may reveal itself in the Fourier domain (where then most FCs are zero).

We considered three different FTs: the classic *Walsh-Hadamard transform (WHT)* (Bernasconi et al., 1996), *FT3* (Püschel and Wendler, 2020) and *FT4* (Püschel and Wendler, 2020). The benefit of considering multiple FTs is that they offer *different, non-equivalent* notions of sparsity. This implies that each FT provides us with a new *lens on the bidder's value function*, potentially revealing structure and thus reducing dimensionality.

In Figure 1.1, we present this key advantage of representing bidders' value functions in different Fourier domains. First, we compute the FTs of all bidders and then calculate their corresponding *spectral energy* distribution, i.e., the (normalized) distribution of the squared FCs corresponding to a fixed cardinality (i.e., number of items) of the input set. In Figure 1.1, we present the mean over 30 CA instances and bidder types in the complex Local Synergy Value Model (LSVM), a synthetic spectrum auction domain (Weiss et al., 2017). We see that while the spectral energy is spread among FCs of various degrees in FT3 and FT4, in WHT the low degree (≤ 3) FCs contain most of the energy, i.e., the WHT has much fewer important FCs that accurately describe each bidder's value function, and thus can be learned more easily. Moreover, in our paper we show that the FT3 is identical to the widely-used *polynomial value function representation* (Lahaie, 2010). Therefore, we conclude that solely converting the polynomial representation into another FT basis (here WHT), without changing the value function at all, can significantly reduce dimensionality and thus be very helpful for the design of ML-based ICAs.

In our paper, we then leverage this key observation to design a new ICA mechanism based on neural networks (NNs) *and* FTs, which we call *Hybrid ICA* (Weissteiner et al., 2022b, Algorithm 3). On a high level, Hybrid ICA first learns a good support (i.e., the most dominant FCs) of each bidder via NNs and second builds a Fourier-sparse approximation based on those supports. Finally, we use this Fourier-sparse approximation to create value queries similar to the original MLCA (Brero et al., 2021). In (Weissteiner et al., 2022b, Table 3), we show that Hybrid ICA achieves higher efficiency than the NN-powered ICA mechanism proposed in (Weissteiner and Seuken, 2020).

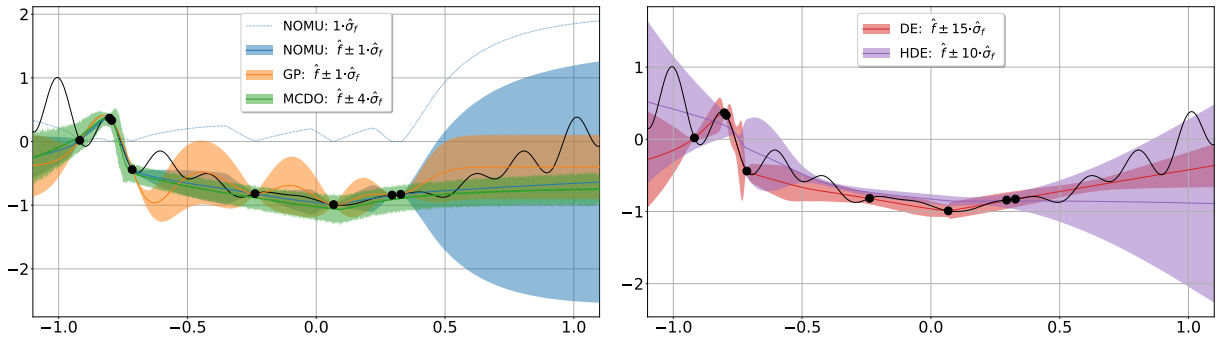


Figure 1.2: UBs resulting from NOMU, GP, MCDO, DE, and HDE for the Levy function (solid black line). For NOMU, we also show $\hat{\sigma}_f$ as a dotted blue line. Training points are shown as black dots.

This paper completes the answer to Research Question 1. We have brought together the two fields of Fourier analysis for set functions and CA design. We have proposed a new state-of-the-art *Hybrid ICA* mechanism based on both NNs and Fourier analysis that leverages different notions of sparsity resulting in higher allocative efficiency than prior ML-based ICAs. We can conclude that using sparser representations of value functions can significantly reduce the dimensionality and facilitate the learning task.

1.4.3 NOMU: Neural Optimization-based Model Uncertainty

This paper provides a complete answer to Research Question 2. In this paper, we present a new approach for estimating posterior model uncertainty for NNs in regression settings with scarce training data observations and small data noise, which we call *neural optimization-based model uncertainty (NOMU)*. In contrast to a fully Bayesian approach for NNs, where defining a realistic prior and subsequently approximating the posterior (e.g., by variational inference or Markov chain Monte Carlo techniques) is in general a very challenging task, we take a different approach and *directly* estimate posterior model uncertainty (i.e., without explicitly defining a prior) by enforcing five desiderata of posterior model uncertainty that any method should satisfy.

Specifically, we first introduce five desiderata that we argue posterior model uncertainty bounds (UBs) should satisfy (Heiss et al., 2022, Section 3.1). We then introduce NOMU, whose main idea is to design a network architecture consisting of two connected sub-NNs (Heiss et al., 2022, Section 3.2), one for the model (i.e., mean) prediction and one for the posterior model uncertainty prediction, and to train it using a carefully-designed loss function (Heiss et al., 2022, Section 3.3), such that the estimated posterior model UBs fulfill our five desiderata. NOMU is easy to implement, scales well to large NNs and its posterior model uncertainty estimate can be represented as a *single* NN (in contrast to ensemble methods). This is particularly important, since we can then use the NN-MILP encoding proposed by Weissteiner and Seuken (2020) to encode the posterior model upper UB as a MILP and use it to define an acquisition function in iterative combinatorial assignment mechanisms (e.g., MLCA) to better control the exploration-exploitation trade-off (see Section 1.4.5). Moreover, because of its modular architecture, NOMU can easily be used to obtain posterior model UBs for already trained NNs.

Figure 1.2 exemplifies our findings, showing typical model UBs, i.e., $[\hat{f} \pm \hat{\sigma}_f]$, where $\hat{f}$ and $\hat{\sigma}_f$

represent the estimated mean prediction and the estimated standard deviation of the posterior model uncertainty, for all considered algorithms for the Levy test function. We find that *Monte Carlo dropout (MCDO)* (Gal and Ghahramani, 2016) consistently yields tube-like UBs; in particular, its UBs do not narrow at training points, i.e., failing an in-sample desideratum that is required by Bayesian theory (Heiss et al., 2022, Desideratum D2). Moreover, it only fulfills our out-of-sample desideratum, i.e., (Heiss et al., 2022, Desideratum D3), to a limited degree. We frequently observe that *deep ensembles (DE)* (Lakshminarayanan et al., 2017) leads to UBs of somewhat arbitrary shapes. This can be seen most prominently in Figure 1.2 around $x \approx -0.75$ and at the edges of its input range, where DE's UBs are very different in width with no clear justification. Thus, also DE is limited in our out-of-sample desideratum, i.e., (Heiss et al., 2022, Desideratum D3). In addition, we sometimes see that also DE's UBs do not narrow sufficiently at training points, i.e., they do not fulfil our in-sample desideratum, i.e., (Heiss et al., 2022, Desideratum D2). *Hyper deep ensemble's (HDE)* (Wenzel et al., 2020b) UBs are even more random, i.e., predicting large posterior model uncertainty at training points and sometimes zero posterior model uncertainty in gaps between them (e.g., $x \approx -0.75$).[6] In contrast, NOMU displays the behaviour it is designed to show. Its UBs nicely tighten at training points and expand in-between (fulfilling desiderata D1–D3 formulated in (Heiss et al., 2022), for desideratum D4 see (Heiss et al., 2022, Appendix D.4)). Like NOMU, the GP fulfills desiderata D1–D3 from (Heiss et al., 2022) well, but cannot account for desideratum D4 (Metric Learning) formulated in (Heiss et al., 2022), since a fixed kernel does not depend on the model prediction.

To benchmark NOMU's posterior model uncertainty estimates also from a quantitative point of view, we experimentally evaluate NOMU in various different regression settings: in scarce and noiseless settings to isolate posterior model uncertainty (Heiss et al., 2022, Sections 4.1.1 and 4.1.2) and on real-world data sets (Heiss et al., 2022, Sections 4.1.3 and 4.1.4). We show that NOMU performs well across all these settings while state-of-the-art methods (MCDO, DE, and HDE) exhibit several deficiencies.[7] Specifically, in (Heiss et al., 2022, Section 4.1.4), we test NOMU's performance on the UCI data sets proposed in (Hernández-Lobato and Adams, 2015), a common benchmark for uncertainty quantification in *noisy*, real-world regression. Our results show, that even the current version of NOMU, which does not explicitly model data noise, already performs on par with existing benchmarks on these real-world regression tasks *with* data noise. Incorporating data noise estimation in the current version of NOMU might further boost its performance. Finally, we evaluate the performance of NOMU in high-dimensional noiseless Bayesian optimization with costly evaluations where good posterior model uncertainty estimates are key and show that NOMU performs as well or better than all considered benchmarks (Heiss et al., 2022, Section 4.2).

[6] Possible reasons for HDE's particularly bad performance in our setting are: (i) the scarcity of training/validation data. HDE trains its NNs based on 80% of the training points and uses the remaining 20% to build an ensemble based on a score, whilst the other methods can use 100% of the training points for training. In a scarce data setting this implies that first, the mean prediction of HDE does not fit through all the training points, and second, the scoring rule is less reliable and (ii) in a noiseless setting one already knows that the L2-regularization should be small, and thus optimizing this parameter is less useful here.

[7] We also conducted experiments using the method proposed by Blundell et al. (2015). However, we found that this method did not perform as well as the other considered benchmarks. Moreover, it was shown in (Gal and Ghahramani, 2016; Lakshminarayanan et al., 2017) that DE and MCDO outperform the methods by Hernández-Lobato and Adams (2015) and Graves (2011), respectively. Therefore, we do not include the methods by Graves (2011), Blundell et al. (2015), and Hernández-Lobato and Adams (2015) in our experiments.

This paper completes the answer to Research Question 2 by introducing a novel method to capture posterior model uncertainty of NNs specifically suited to settings with small data noise and scarce training data. Furthermore, NOMU's posterior model uncertainty estimate can be represented by a single NN and thus can be encoded as a succinct MILP making NOMU particularly well suited for combinatorial assignment problems.

1.4.4 Monotone-Value Neural Networks: Exploiting Preference Monotonicity in Combinatorial Assignment

This paper provides the first answer to Research Question 3 by introducing a novel class of neural networks: *monotone-value neural networks (MVNNs)*. MVNNs by their design incorporate important domain-specific prior knowledge (i.e., a monotonicity and normalization property) about agents' preferences in combinatorial assignment settings. This hard-coded prior knowledge helps to improve the generalization performance, especially in settings with few training data points (e.g., bids) and makes MVNNs the state-of-the-art algorithm to learn monotone and normalized preferences in combinatorial assignment.

Recall that in Section 1.4.1, we enhanced the MLCA mechanism by integrating NNs instead of SVRs with quadratic kernel, which are more expressive and ultimately led to allocations with higher efficiency. However, from a Bayesian optimization point of view, there still remain two main deficiencies of the NN-based MLCA: First, MLCA uses plain feed-forward fully-connected NNs as prior to learn the bidders value functions, which are "too generic" and do not account for the important monotonicity property of bidders' value functions. Second, the query generation in MLCA only exploits (i.e., it maximizes the mean prediction without any notion of uncertainty) and does not properly explore the bundle space. In this paper, we address the first deficiency.

Two common assumptions of agents' value functions in combinatorial assignment are: *normalization (i.e., "zero value for the empty bundle of items")* and *monotonicity (i.e., "additional items increase value")*. These properties are satisfied in many economic market domains. For example, in many combinatorial auctions (where this property is often called *free disposal*), bidders can freely dispose of unwanted items; in combinatorial course allocation, students can just drop courses they have been assigned.

To account for this, we introduce MVNNs, which are specifically suited to model monotone combinatorial value functions. Concretely, we implement the *normalization* and *monotonicity* property of MVNNs via constraints on the weights (i.e., non-negative weights) and biases (i.e., non-positive biases) and use as activation function the *bounded ReLU (bReLU)*. Furthermore, we show in (Weissteiner et al., 2022a, Theorem 1) that with this design one can represent *any* value function with arbitrarily complex substitutabilities and complementarities that satisfies the *normalization* and *monotonicity* property *exactly* as a finite-width MVNN. Finally, to efficiently solve MVNN-based winner determination problems (WDPs), we provide for MVNNs a succinct MILP formulation. Finding tight bounds on the neurons for NN-based MILPs is an active area of research and critical to scale NN-based MILPs to larger architectures. For a plain ReLU NN, bounds based on interval arithmetic (a.k.a. box constraints) are not tight. However, for MVNNs, it turns out that these bounds are always perfectly tight, because of their encoded monotonicity. The upper bound of an arbitrary neuron is the value the neuron would output for the full bundle and the lower bound is the corresponding value for the empty bundle.

We experimentally evaluated the learning performance of MVNNs vs. plain NNs in four different synthetic spectrum auction domains from the spectrum auction test suite (SATS) (Weiss et al., 2017) and showed that MVNNs are significantly better at modelling bidders' combinatorial value functions. Furthermore, we experimentally investigated the performance of MVNNs vs. plain NNs when integrated into an existing ML-based iterative combinatorial auction mechanism (i.e., MLCA proposed by Brero et al. (2021)) and compared them also to the recently introduced Fourier transform-based method by Weissteiner et al. (2022b). We showed that using MVNNs in MLCA leads to significantly smaller efficiency losses than all other considered auction mechanisms.

This paper introduces MVNNs as a domain-specific prior in combinatorial assignment where agents are assumed to have monotone combinatorial value functions and provides the first answer to Research Question 3. Overall, our results show that MVNNs improve the prediction performance, they yield state-of-the-art allocative efficiency in the auction, and they also reduce the runtime of the ML-based WDPs. We conclude that incorporating domain-specific prior knowledge in combinatorial assignment into the ML algorithm can indeed be advantageous, especially in settings with scarce data points.

1.4.5 Bayesian Optimization-based Combinatorial Assignment

This paper provides a complete answer to Research Question 3. Based on MVNNs as domain-specific prior ML algorithm, we propose in this paper a *Bayesian optimization-based combinatorial assignment (BOCA)* mechanism which includes a notion of posterior *model uncertainty* to properly *explore* and not just *exploit* the bundle space during its preference elicitation phase.

The main idea of previously introduced ML-powered preference elicitation algorithms (Brero et al., 2018, 2021; Weissteiner and Seuken, 2020; Weissteiner et al., 2022b,a) is two-fold: first, they train a separate ML model to learn each bidder's full value function from a small set of bids; second, they solve an *ML-based winner determination problem (WDP)* to determine the allocation with the highest predicted social welfare, and they use this allocation to generate the next set of queries to all bidders. This process repeats in an iterative fashion until a fixed number of queries has been asked. Thus, their ML-powered ICA can be interpreted as combinatorial *Bayesian optimization (BO)* algorithm, with the goal of maximizing the (true) social welfare function.

However, in light of BO, the main shortcoming of this prior work is that all of these approaches are *myopic* in the sense that these mechanisms simply query the allocation with the highest predicted welfare. In particular, these mechanisms do not have an explicit model of *uncertainty* over an agent's values for not yet elicited bundles, although handling uncertainty in a principled manner is one of the key requirements in BO and when designing a smart preference elicitation algorithm (Guo et al., 2010). Thus, prior mechanisms cannot properly control the *exploration-exploitation trade-off* inherent to BO. For ML-based iterative combinatorial assignment mechanisms, this means that these mechanisms may get stuck in local minima, repeatedly querying one part of the allocation space while not exploring other, potentially more efficient allocations.

In this paper, we address this shortcoming and show how to integrate a notion of posterior *model uncertainty* (i.e., epistemic uncertainty) over agents' preferences into iterative combinato-

rial assignment. Concretely, we design a *Bayesian optimization-based combinatorial assignment (BOCA)* mechanism that makes use of posterior model uncertainty in its query generation module. The main technical challenge is to design a new method for estimating an *upper uncertainty bound (uUB)* that can be used to define an acquisition function to determine the next query.[8] To this end, we combine *MVNNs* (Weissteiner et al., 2022a) with *NOMU* (Heiss et al., 2022). Specifically, we make the following contributions.

First, we present a modified NOMU algorithm, tailored to combinatorial assignment, exploiting monotonicity of agents' preferences and the discrete (finite) nature of this setting. Concretely, our new NOMU algorithm is based on the following two key characteristics of combinatorial assignment: (i) since agents' value functions are monotonically increasing, the uUBs need to be monotonically increasing too (Weissteiner et al., 2023, Propositions 1 and 2), and (ii) due to the (finite) discrete input space, one can derive a closed-form expression of the 100%-uUB as an MVNN. We then use this closed form MVNN expression of the 100%-uUB together with the MVNN mean prediction in the design of a new NOMU loss function to enforce that our uUB lies between the mean and the 100%-uUB, i.e, approximating an α%-uUB with $\alpha \in [50, 100]$.

Second, we show that generic parameter initialization for monotone NNs (including MVNNs) can fail and propose a new initialization method for MVNNs based on uniform mixture distributions. Concretely, the non-negativity constraints of the weights in an MVNN imply that when using a generic parameter initialization of standard (non-constrained) NNs that the conditional mean of the output of each neuron in the MVNN either explodes or its conditional variance vanishes (depending on the scaling of the distribution). To overcome this issue, we propose an i.i.d. mixture distribution of two different uniform distributions such that the conditional mean *and* the conditional variance neither explode nor vanish with increasing width of the network but rather stay constant.

Third, we present a more succinct mixed integer linear program (MILP) for MVNNs to solve the ML-based WDP using our proposed MVNN-based uUB as ML algorithm. Our new MILP only contains half the linear constraints compared to the MILP proposed in (Weissteiner et al., 2022a).

Finally, we experimentally compared BOCA in the spectrum auction test suite (SATS) (Weiss et al., 2017) against state-of-art-approaches and showed that BOCA outperforms these approaches in terms of allocative efficiency.

This paper completes the answer to Research Question 3. With this paper, we have proposed a practical fully *Bayesian optimization-based combinatorial assignment (BOCA)* mechanism. On a conceptual level, our main contribution is the integration of posterior model uncertainty over agents' preferences into ML-based preference elicitation. On a technical level, we have designed a new method for estimating an uUB that exploits the monotonicity of agents' preferences in the combinatorial assignment domain and the finite nature of this setting. From our experimental results, we conclude that using a notion of posterior model uncertainty that enables the mechanism to properly *explore* and not just *exploit* the bundle space during its preference elicitation phase, can significantly boost its performance.

[8] In the literature, upper uncertainty bound (uUB) is also sometimes called *upper confidence bound (UCB)* and BO algorithms using the UCB as acquisition function are referred to as UCB-algorithms.

1.4.6 Machine Learning-powered Course Allocation

This paper provides a complete answer to Research Question 4. In this paper, we introduce a machine learning-powered course allocation mechanism. Concretely, we extend the state-of-the-art *Course Match (CM)* mechanism (Budish et al., 2017) with a machine learning-based preference elicitation module. We call our proposed mechanism *machine learning-powered course match (MLCM)*. MLCM generates in an iterative, asynchronous manner, carefully selected *pairwise comparison queries* that are tailored to each individual student.

With this approach, we build on the ideas developed in a recent stream of papers on *ML-powered combinatorial auctions*. Brero et al. (2018) were the first to integrate an ML-powered preference elicitation component into a practical combinatorial auction mechanism. They used support vector regression to learn bidders' value functions and to iteratively generate new informative queries in each auction round. In (Brero et al., 2021), the authors proposed the MLCA mechanism and showed that it achieves higher allocative efficiency than the widely-used *combinatorial clock auction* (Ausubel et al., 2006).

While these works are important precursors to this paper, there are several noteworthy differences. First, these papers used *value queries* as the interaction paradigm, which would be unnatural in course allocation. Instead, we use *pairwise comparison queries* (i.e., asking students "Do you prefer course schedule A or B?"). Importantly, a pairwise comparison query is a simpler type of query, known to have low cognitive load (Conitzer, 2007; Chajewska et al., 2000). Second, our goal is to build on top of the CM's simple reporting language (recall that CM offers students a graphical user interface (GUI) to enter a *base value* between 0 and 100 for each course, and an *adjustment value* between -200 and 200 for each *pair* of courses). Thus, we must be able to handle the *cardinal* input that students provide via the CM reporting language as well as the *ordinal* feedback from answering pairwise comparison queries. Finally, while an auctioneer can require bidders in an auction to participate in a synchronous way (i.e., submitting a bid in every round), we must allow students to interact with the mechanism in an asynchronous manner.

The high-level idea of MLCM is as follows. First, students use the CM reporting language (i.e., the same GUI as in CM). As in CM, this input is required from all students. Second, MLCM uses these initial reports to train a distinct ML model for each student so that it can predict each student's value for any possible course schedule. Third, MLCM uses an ML-powered preference elicitation algorithm to generate *comparison queries* that are tailored to each student, and students simply answer which schedule they prefer. Based on this feedback, the ML model is retrained and the next query is generated. Importantly, this phase is *optional* – each student can answer as many of such queries as she wants (including none). However, the more queries she answers, the better the ML model will typically approximate her true preferences, which will benefit her in the last phase, where MLCM computes the final allocation based on all trained ML models.[9]

To evaluate the welfare achieved by MLCM, we introduce a new course allocation simulation

[9] Note that in MLCM, the final allocation is determined based on the trained ML models. This is different to our prior work on ML-based iterative combinatorial auctions where the final allocation is calculated *only* based on reported/elicited bundle-value pairs. The main reason for only using reported/elicited bundle-value pairs in the combinatorial auction setting is, that, when allocating based on predicted bundle-value pairs (i.e., based on trained ML models), one would lose individual rationality of the auction mechanism.

framework. The first component is a realistic student preference generator, which is designed such that each student's complete preferences can be encoded as a succinct mixed integer linear program (MILP). This allows us to compute a benchmark allocation given students' true preferences. The second component models students' *reporting mistakes* when interacting with the CM language. We calibrate the framework's parameters based on real-world data from the field experiment in (Budish and Kessler, 2022).

Regarding the ML algorithm used for MLCM, we show experimentally that the recently introduced *monotone-value neural networks (MVNNs)* (Weissteiner et al., 2022a) exhibit the best generalization performance in our domain, while also being MILP-formalizable, such that the corresponding individual student utility maximization problem can be solved fast enough in practice. Furthermore, we show how the cardinal input from the CM language and the ordinal feedback from the pairwise comparison queries can be combined when training MVNNs.

Finally, we empirically compare the performance of MLCM and CM. We find that MLCM significantly outperforms CM in terms of *average student utility* as well as *minimum student utility*, even with only ten additional pairwise comparison queries. Furthermore, we show that these results are robust to changes in students' reporting mistakes and across various different settings of our proposed student preference generator.

This paper completes the answer to Research Question 4 by proposing MLCM, a new practical ML-powered course allocation mechanism. Specifically, MLCM addresses the main shortcomings of CM's reporting language, revealed in the field experiment by Budish and Kessler (2022). We thus conclude that using ML to carefully select good pairwise comparison queries that better elicit students' preferences by correcting their reporting mistakes can significantly increase students' welfare whilst at the same time not creating too much cognitive burden to students.

1.5 Conclusion and Future Work

Integrating machine learning methods into market mechanisms such as combinatorial assignment is an intricate task, but one of critical importance in today's large-scale, complex environments. In particular, properly eliciting all agents' preferences plays an integral part in the design of well functioning market mechanisms.

Since agents' preferences are exponentially-sized objects, full elicitation is rarely possible in real-world settings. Therefore, a key challenge in combinatorial assignment is the design of a *preference elicitation* algorithm that is (i) *practically feasible* with respect to elicitation costs and (ii) *smart*, i.e., it should elicit the information that is "most useful" for achieving high efficiency. Integrating ML into the design of a preference elicitation algorithm first involves learning agents' preferences via ML algorithms and then use those trained ML algorithms to select the "most useful" queries to achieve allocations with high efficiency.

In this thesis, I have studied how to leverage a variety of machine learning algorithms to design such smart and practically feasible ML-powered preference elicitation algorithms in combinatorial assignment. Specifically, I have both proposed some tailor-made ML-based algorithms specifically designed for combinatorial assignment (i.e., NOMU in Section 1.4.3 and MVNNs in Section 1.4.4) as well as integrated existing ML algorithms into combinatorial assignment mech-

anisms (e.g., NNs in Section 1.4.1, Fourier transforms for set functions in Section 1.4.2). These ML models help to guide the elicitation process by eliciting the "most useful" information from the agents and thereby trying to overcome the curse of dimensionality inherent to combinatorial assignment.

However, even though learning agents' preferences via ML algorithms can be seen as a standard supervised learning task, it is particularly challenging in combinatorial assignment, because the underlying setting is typically extremely scarce, i.e., practical feasibility of the preference elicitation algorithm implies that only few training data points can be collected.

Moreover, the main computational bottleneck of the considered preference elicitation algorithms lies in the next-query generation method where one has to solve a hard combinatorial optimization problem using as inputs the trained ML models (i.e., the ML-based winner determination problem). This implies that highly overparameterized ML models cannot be used in practice and thus carefully controlling the trade-off between the ML models' expressivity and the computational feasibility of the resulting ML-based winner determination problem is of utmost importance.

I have shown that in this specific scarce setting, it is even more important to use as much prior knowledge as possible (Weissteiner et al., 2022a) and to reduce the dimensionality of the learning task (Weissteiner et al., 2022b) to get the best possible generalization performance, which, when combined with a smart next-query generation procedure, then leads to allocations with higher efficiency.

Additionally, I have shown that integrating uncertainty over not yet elicited bundles in preference elicitation (i.e., following more closely the Bayesian optimization paradigm) can prevent from getting "stuck" in local optima and helps to properly explore bundle space (Weissteiner et al., 2023).

Furthermore, when designing a practical combinatorial assignment mechanism, it is also important to consider the communication/interaction paradigm with the central agency, i.e., one should reflect which query type is the most practical in the considered domain. For example, in course allocation, to avoid creating too much cognitive burden for students, I have proposed the use of simpler pairwise comparison queries (Soumalias et al., 2023) instead of value queries.

Overall, all these subtle details in the choice of the ML model used for preference elicitation and how they interact with each other ultimately play a key role for the success of the resulting combinatorial assignment mechanism.

Future Work I have evaluated all my proposed ML-based preference elicitation algorithms in synthetic settings, i.e., in the spectrum auction test suite (SATS) by Weiss et al. (2017) or via the student preference generator by Soumalias et al. (2023), assuming that all agents report truthfully. Therefore, I see the following two important directions for future work.

First, future work could analyze the mechanisms when agents do strategically misreport. Even though we argue that all considered mechanisms provide good incentives in practice, none of these guarantees hold in a formal, provable way. Therefore, it would be particularly interesting to find approximately optimal unilateral misreporting strategies and analyze the characteristics of them in more detail, e.g., how do these strategies differ from truthful reports, how does the final allocation compare to the one obtained when all agents report truthfully, which agents

are worse/better of. I could envision analyzing this by framing the iterative combinatorial assignment problem I studied as a reinforcement learning task and then finding approximately optimal strategies by assuming that all other agents report truthfully. Once this reasonably succeeds, one can then experimentally analyze which of the proposed ML algorithms provide the best robustness with respect to strategic behaviour.

Second, another interesting direction for future work is to evaluate the proposed mechanisms not only in synthetic settings but also when put into practice. Specifically, it would be of great interest to define suitable laboratory experiments to evaluate the proposed ML-powered mechanisms. Given that in practice, agents are often limited by their cognitive abilities (Scheffel et al., 2012) and thus (even unintentionally) make mistakes when interacting with the mechanism, such laboratory experiments would provide valuable insights on the real-world performance of the proposed ML-powered mechanisms.

Furthermore, in the design of new preference elicitation algorithms in the combinatorial auction domain, I have focused on integrating/designing new ML components and did not change the interaction paradigm, i.e., value queries, proposed by prior work. An interesting direction for future work would be to additionally also consider new or combine existing interaction paradigms in my proposed mechanisms, e.g., to alternate between value queries and demand queries in the preference elicitation phase.

Finally, another interesting avenue to be explored in future work is to test the key conceptual idea of BOCA, i.e, using the upper uncertainty bound based on NOMU and NNs as acquisition function, not only in the context of combinatorial assignment but also in other combinatorial Bayesian optimization tasks (e.g., see (Baptista and Poloczek, 2018, Section 4) for other different combinatorial BO tasks). Similarly, one could test the performance of MVNNs in other monotone regression tasks than the spectrum auction setting.

Bibliography

D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané. Concrete problems in AI safety. *CoRR*, abs/1606.06565, 2016. 3

Andisheh Amrollahi, Amir Zandieh, Michael Kapralov, and Andreas Krause. Efficiently learning Fourier sparse set functions. In *Advances in Neural Information Processing Systems*, volume 32, pages 15120–15129, 2019. 14

Ross Anderson, Joey Huchette, Will Ma, Christian Tjandraatmadja, and Juan Pablo Vielma. Strong mixed-integer programming formulations for trained neural networks. *Mathematical Programming*, 183(1):3–39, 2020. 9

A. Ashukha, A. Lyzhov, D. Molchanov, and D. Vetrov. Pitfalls of in-domain uncertainty estimation and ensembling in deep learning. *arXiv preprint arXiv:2002.06470*, 2020. URL https://arxiv.org/abs/2002.06470. 9

L. Ausubel and O. Baranov. A practical guide to the combinatorial clock auction. *Economic Journal*, 127(605):F334–F350, 2017. 2

Lawrence M Ausubel. An efficient dynamic auction for heterogeneous commodities. *American Economic Review*, 96(3):602–629, 2006. 8

Lawrence M Ausubel and Paul R Milgrom. Ascending auctions with package bidding. *The BE Journal of Theoretical Economics*, 1(1), 2002. 7

Lawrence M Ausubel, Peter Cramton, and Paul Milgrom. The clock-proxy auction: A practical combinatorial auction design. In Peter Cramton, Yoav Shoham, and Richard Steinberg, editors, *Combinatorial Auctions*, pages 115–138. MIT Press, 2006. 21

K. Bansak, J. Ferwerda, J. Hainmueller, A. Dillon, D. Hangartner, D. Lawrence, and J. Weinstein. Improving refugee integration through data-driven algorithmic assignment. *Science*, 359(6373):325–329, 2018. 1

Ricardo Baptista and Matthias Poloczek. Bayesian optimization of combinatorial structures. In *International Conference on Machine Learning*, pages 462–471. PMLR, 2018. 24

A Bernasconi, B Codenotti, and J Simon. On the fourier analysis of boolean functions. *preprint*, pages 1–24, 1996. 15

Manuel Beyeler, Gianluca Brero, Benjamin Lubin, and Sven Seuken. imlca: Machine learning-powered iterative combinatorial auctions with interval bidding. In *Proceedings of the 22nd ACM Conference on Economics and Computation*, pages 136–136, 2021. 4, 8

Martin Bichler, Vladimir Fux, and Jacob K Goeree. Designing combinatorial exchanges for the reallocation of resource rights. *Proceedings of the National Academy of Sciences*, 116(3): 786–791, 2019. 1

Avrim Blum, Jeffrey Jackson, Tuomas Sandholm, and Martin Zinkevich. Preference elicitation and query learning. *Journal of Machine Learning Research*, 5:649–667, 2004. 7, 8

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. In *32nd International Conference on Machine Learning*, 2015. 9, 10, 17

G. Brero, B. Lubin, and S. Seuken. Probably approximately efficient combinatorial auctions via machine learning. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, 2017. 8

G. Brero, B. Lubin, and S. Seuken. Combinatorial auctions via machine learning-based preference elicitation. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 2018. 2, 8, 13, 19, 21

G. Brero, B. Lubin, and S. Seuken. Machine learning-powered iterative combinatorial auctions. *arXiv preprint arXiv:1911.08042*, Jan 2021. URL `https://arxiv.org/abs/1911.08042`. 2, 5, 8, 13, 15, 19, 21

Gianluca Brero and Sébastien Lahaie. A bayesian clearing mechanism for combinatorial auctions. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, 2018. 8

Gianluca Brero, Sébastien Lahaie, and Sven Seuken. Fast iterative combinatorial auctions via bayesian learning. In *Proceedings of the 33rd AAAI Conference of Artificial Intelligence*, 2019. 8

E. Budish, G. Cachon, J. Kessler, and A. Othman. Course match: A large-scale implementation of approximate competitive equilibrium from equal incomes for combinatorial allocation. *Operations Research*, 65(2):314–336, 2017. 6, 21

Eric Budish. The combinatorial assignment problem: Approximate competitive equilibrium from equal incomes. *Journal of Political Economy*, 119(6):1061–1103, 2011. 1, 6

Eric Budish and Estelle Cantillon. The multi-unit assignment problem: Theory and evidence from course allocation at harvard. *American Economic Review*, 102(5):2237–71, 2012. 6

Eric Budish and Judd B Kessler. Can market participants report their preferences accurately (enough)? *Management Science*, 68(2):1107–1130, 2022. 4, 7, 22

Urszula Chajewska, Daphne Koller, and Ronald Parr. Making rational decisions using adaptive utility elicitation. In *Aaai/Iaai*, pages 363–369, 2000. 21

Chih-Hong Cheng, Georg Nührenberg, and Harald Ruess. *Automated Technology for Verification and Analysis. Lecture Notes in Computer Science*, volume 10482, chapter Maximum resilience of artificial neural networks. Springer, Cham, 2017. 9

Wei Chu and Zoubin Ghahramani. Preference learning with gaussian processes. In *Proceedings of the 22nd international conference on Machine learning*, pages 137–144, 2005. 8

Wolfram Conen and Tuomas Sandholm. Preference elicitation in combinatorial auctions. In *Proceedings of the 3rd ACM Conference on Electronic Commerce*, pages 256–259, 2001. 7

Wolfram Conen and Tuomas Sandholm. Partial-revelation vcg mechanism for combinatorial auctions. In *AAAI/IAAI*, pages 367–372, 2002. 7

Wolfram Conen and Tuomas Sandholm. Differential-revelation vcg mechanisms for combinatorial auctions. In *Proceedings of the 4th ACM conference on Electronic commerce*, pages 196–197, 2003. 7

Vincent Conitzer. Eliciting single-peaked preferences using comparison queries. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, pages 1–8, 2007. 21

Vincent Conitzer, Tuomas Sandholm, and Paolo Santi. Combinatorial auctions with k-wise dependent valuations. In *AAAI*, volume 5, pages 248–254, 2005. 8

Peter Cramton. Spectrum auction design. *Review of Industrial Organization*, 42(2):161–190, 2013. 1

Sven de Vries, James Schummer, and Rakesh V Vohra. On ascending vickrey auctions for heterogeneous objects. *Journal of Economic Theory*, 132(1):95–118, 2007. 7

P. Dütting, Z. Feng, H. Narasimhan, D. Parkes, and S. Ravindranath. Optimal auctions through deep learning. In *Proceedings of the 36th International Conference on Machine Learning*, 2019. 3, 9

Matteo Fischetti and Jason Jo. Deep neural networks and mixed integer linear optimization. *Constraints*, 23(3):296–309, Jul 2018. ISSN 1572-9354. doi: 10.1007/s10601-018-9285-6. 9

S. Fort, H. Hu, and B. Lakshminarayanan. Deep ensembles: A loss landscape perspective. *arXiv preprint arXiv:1912.02757*, 2019. URL https://arxiv.org/abs/1912.02757. 9

Peter I Frazier. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018. URL https://arxiv.org/abs/1807.02811. 5

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *33rd International Conference on Machine Learning*, pages 1050–1059, 2016. 9, 17

Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and SM Ali Eslami. Conditional neural processes. In *International Conference on Machine Learning*, pages 1704–1713. PMLR, 2018a. 10

Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J Rezende, SM Eslami, and Yee Whye Teh. Neural processes. *arXiv preprint arXiv:1807.01622*, 2018b. URL https://arxiv.org/abs/1807.01622. 10

Noah Golowich, Harikrishna Narasimhan, and David C Parkes. Deep learning for multi-facility location mechanism design. In *Proceedings of the Twenty-seventh International Joint Conference on Artificial Intelligence and the Twenty-third European Conference on Artificial Intelligence*, pages 261–267, 2018. 9

Alex Graves. Practical variational inference for neural networks. In *Advances in neural information processing systems*, volume 24, pages 2348–2356, 2011. 9, 17

Faruk Gul and Ennio Stacchetti. The english auction with differentiated commodities. *Journal of Economic theory*, 92(1):66–95, 2000. 8

Shengbo Guo, Scott Sanner, and Edwin V Bonilla. Gaussian process preference elicitation. In *Advances in Neural Information Processing Systems*, volume 23, 2010. 8, 19

F. Gustafsson, M. Danelljan, and T. Schon. Evaluating scalable bayesian deep learning methods for robust computer vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 318–319, 2020. 9

M. Havasi, R. Jenatton, S. Fort, J. Liu, J. Snoek, B. Lakshminarayanan, A. Dai, and D. Tran. Training independent subnetworks for robust prediction. In *International Conference on Learning Representations*, 2021. 10

Jakob M Heiss, Jakob Weissteiner, Hanna S Wutte, Sven Seuken, and Josef Teichmann. NOMU: Neural optimization-based model uncertainty. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 8708–8758. PMLR, 17–23 Jul 2022. URL `https://proceedings.mlr.press/v162/heiss22a.html`. 1, 2, 4, 10, 16, 17, 20

José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning*, pages 1861–1869, 2015. 9, 17

Benoit Hudson and Tuomas Sandholm. Effectiveness of query types and policies for preference elicitation in combinatorial auctions. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 386–393, 2004. 7

Alexander S Kelso Jr and Vincent P Crawford. Job matching, coalition formation, and gross substitutes. *Econometrica: Journal of the Econometric Society*, pages 1483–1504, 1982. 8

Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in neural information processing systems*, volume 30, pages 5574–5584, 2017. 10

A. Khosravi, S. Nahavandi, D. Creighton, and A. Atiya. Lower upper bound estimation method for construction of neural network-based prediction intervals. *IEEE transactions on neural networks*, 22(3):337–346, 2010. 10

Volodymyr Kuleshov and Shachi Deshpande. Calibrated and sharp uncertainties in deep learning via density estimation. In *Proceedings of the 39th International Conference on Machine Learning*, pages 11683–11693. PMLR, July 2022. 10

Volodymyr Kuleshov, Nathan Fenner, and Stefano Ermon. Accurate uncertainties for deep learning using calibrated regression. In *Proceedings of the 35th International Conference on Machine Learning*, pages 2796–2804, Stockholmsmässan, Stockholm Sweden, July 2018. PMLR. 10

Anthony M Kwasnica, John O Ledyard, Dave Porter, and Christine DeMartini. A new and improved design for multiobject iterative auctions. *Management science*, 51(3):419–434, 2005. 7

S. Lahaie. Kernel methods for revealed preference analysis. In *ECAI*, pages 439–444, 2010. 15

Sebastien M Lahaie and David C Parkes. Applying learning algorithms to preference elicitation. In *Proceedings of the 5th ACM Conference on Electronic Commerce*, 2004. 7, 8

B Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in neural information processing systems*, volume 30, pages 6402–6413, 2017. 9, 17

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015. 3

Xingchao Liu, Xing Han, Na Zhang, and Qiang Liu. Certified monotonic neural networks. *arXiv preprint arXiv:2011.10219*, 2020. URL https://arxiv.org/abs/2011.10219. 11

Andrey Malinin and Mark Gales. Predictive uncertainty estimation via prior networks. In *Advances in Neural Information Processing Systems*, volume 31, pages 7047–7058, 2018. 10

Andrey Malinin, Sergey Chervontsev, Ivan Provilkov, and Mark Gales. Regression prior networks. *arXiv preprint arXiv:2006.11590*, 2020. URL https://arxiv.org/pdf/2006.11590.pdf. 10

D. Masterov, U. Mayer, and S. Tadelis. Canary in the e-commerce coal mine: Detecting and predicting poor experiences using buyer-to-seller messages. In *16th Conference on Economics and Computation*, 2015. 1

Paul R Milgrom and Steven Tadelis. *23. How Artificial Intelligence and Machine Learning Can Impact Market Design*. University of Chicago Press, 2019. 1

Martin Mladenov, Craig Boutilier, Dale Schuurmans, Ofer Meshi, Gal Elidan, and Tyler Lu. Logistic markov decision processes. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 2486–2493, 2017. 9

Harikrishna Narasimhan, Shivani Brinda Agarwal, and David C Parkes. Automated mechanism design without money via machine learning. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, 2016. 9

Noam Nisan and Ilya Segal. The communication requirements of efficient allocations and supporting prices. *Journal of Economic Theory*, 129(1):192–224, 2006. 2, 8

D. Nix and A. Weigend. Estimating the mean and variance of the target probability distribution. In *Proceedings of IEEE international conference on neural networks (ICNN'94)*, pages 55–60. IEEE, 1994. 10

S. Ober and C Rasmussen. Benchmarking the neural linear model for regression. *arXiv preprint arXiv:1912.08416*, 2019. URL `https://arxiv.org/abs/1912.08416`. 10

Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, D. Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems*, volume 32, 2019. 9

David C Parkes. i bundle: An efficient ascending price bundle auction. In *Proceedings of the 1st ACM Conference on Electronic Commerce*, pages 148–157, 1999. 7

David C Parkes and Lyle H Ungar. Iterative combinatorial auctions: Theory and practice. In *Proceedings of the seventeenth national conference on artificial intelligence*, page 74–81, 2000. 7

T. Pearce, M. Zaki, A. Brintrup, and A. Neely. High-quality prediction intervals for deep learning: A distribution-free, ensembled approach. In *35th International Conference on Machine Learning*, 2018. 10

Neehar Peri, Michael Curry, Samuel Dooley, and John Dickerson. Preferencenet: Encoding human preferences in auction design with deep learning. In *Advances in Neural Information Processing Systems*, volume 34, pages 17532–17542, 2021. 9

M. Püschel and C. Wendler. Discrete signal processing with set functions. *arXiv preprint arXiv:2001.10290*, 2020. URL `https://arxiv.org/abs/2001.10290`. 15

J. Rahme, S. Jelassi, J. Bruna, and M. Weinberg. A permutation-equivariant neural network architecture for auction design. *arXiv preprint arXiv:2003.01497*, Sep 2020. URL `url={https://arxiv.org/abs/2003.01497},`. 3, 9

Tuomas Sandholm and Craig Boutilier. Preference elicitation in combinatorial auctions. *Combinatorial auctions*, 10, 2006. 7

Paolo Santi, Vincent Conitzer, and Tuomas Sandholm. Towards a characterization of polynomial preference elicitation with value queries in combinatorial auctions. In *International Conference on Computational Learning Theory*, pages 1–16. Springer, 2004. 8

Buser Say, Ga Wu, Yu Qing Zhou, and Scott Sanner. Nonlinear hybrid planning with deep net learned transition models and mixed-integer linear programming. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 750–756, 2017. 9

Tobias Scheffel, Georg Ziegler, and Martin Bichler. On the impact of package selection in combinatorial auctions: an experimental study in the context of spectrum auction design. *Experimental Economics*, 15(4):667–692, 2012. 24

W. Shen, P. Tang, and S. Zuo. Automated mechanism design via neural networks. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, 2019. 9

Joseph Sill. Monotonic networks. In *Advances in Neural Information Processing Systems*, volume 10, 1997. 11

Ermis Soumalias, Behnoosh Zamanlooy, Jakob Weissteiner, and Sven Seuken. Machine learning-powered course allocation. *arXiv preprint arXiv:2210.00954*, 2023. URL https://arxiv.org/abs/2210.00954. 1, 3, 4, 23

P. Stobbe and A. Krause. Learning Fourier sparse set functions. In *Artificial Intelligence and Statistics*, pages 1125–1133, 2012. 14

Aleksei Tiulpin and Matthew B. Blaschko. Greedy bayesian posterior approximation with deep ensembles. *Transactions on Machine Learning Research*, 2022. 10

Z. Wang, T. Ren, J. Zhu, and B. Zhang. Function space particle optimization for bayesian neural networks. *arXiv preprint arXiv:1902.09754*, 2019. URL https://arxiv.org/abs/1902.09754. 10

Antoine Wehenkel and Gilles Louppe. Unconstrained monotonic neural networks. In *Advances in Neural Information Processing Systems*, volume 32, 2019. 11

Michael Weiss, Benjamin Lubin, and Sven Seuken. Sats: A universal spectrum auction test suite. In *Proceedings of the 16th Conference on Autonomous Agents and Multi-Agent Systems*, 2017. 5, 13, 15, 19, 20, 23

Jakob Weissteiner and Sven Seuken. Deep learning—powered iterative combinatorial auctions. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(02):2284–2293, Apr. 2020. doi: 10.1609/aaai.v34i02.5606. URL https://ojs.aaai.org/index.php/AAAI/article/view/5606. 1, 3, 5, 13, 14, 15, 16, 19

Jakob Weissteiner, Jakob Heiss, Julien Siems, and Sven Seuken. Monotone-value neural networks: Exploiting preference monotonicity in combinatorial assignment. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 541–548. International Joint Conferences on Artificial Intelligence Organization, 7 2022a. doi: 10.24963/ijcai.2022/77. URL https://doi.org/10.24963/ijcai.2022/77. Main Track. 1, 3, 18, 19, 20, 22, 23

Jakob Weissteiner, Chris Wendler, Sven Seuken, Ben Lubin, and Markus Püschel. Fourier analysis-based iterative combinatorial auctions. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 549–556. International Joint Conferences on Artificial Intelligence Organization, 7 2022b. doi: 10.24963/ijcai.2022/78. URL https://doi.org/10.24963/ijcai.2022/78. Main Track. 1, 3, 5, 15, 19, 23

Jakob Weissteiner, Jakob Heiss, Julien Siems, and Sven Seuken. Bayesian optimization-based combinatorial assignment. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37, 2023. 1, 3, 20, 23

Yeming Wen, Dustin Tran, and Jimmy Ba. Batchensemble: an alternative approach to efficient ensemble and lifelong learning. In *International Conference on Learning Representations*, 2020. 10

Chris Wendler, Andisheh Amrollahi, Bastian Seifert, Andreas Krause, and Markus Püschel. Learning set functions that are sparse in non-orthogonal fourier bases. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(12):10283–10292, May 2021. 14

Florian Wenzel, Kevin Roth, Bastiaan Veeling, Jakub Swiatkowski, Linh Tran, Stephan Mandt, Jasper Snoek, Tim Salimans, Rodolphe Jenatton, and Sebastian Nowozin. How good is the Bayes posterior in deep neural networks really? In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 10248–10259. PMLR, 13–18 Jul 2020a. 9

Florian Wenzel, Jasper Snoek, Dustin Tran, and Rodolphe Jenatton. Hyperparameter ensembles for robustness and uncertainty quantification. In *Advances in Neural Information Processing Systems*, volume 33, pages 6514–6527, 2020b. 9, 17

Peter R Wurman and Michael P Wellman. A k ba: A progressive, anonymous-price combinatorial auction. In *Proceedings of the 2nd ACM Conference on Electronic Commerce*, pages 21–29, 2000. 7

Seungil You, David Ding, Kevin Canini, Jan Pfeifer, and Maya Gupta. Deep lattice networks and partial monotonic functions. In *Advances in Neural Information Processing Systems*, volume 30, 2017. 11

Martin A Zinkevich, Avrim Blum, and Tuomas Sandholm. On polynomial-time preference elicitation with value queries. In *Proceedings of the 4th ACM Conference on Electronic Commerce*, pages 176–185, 2003. 8

2 Deep Learning-powered Iterative Combinatorial Auctions

The content of this chapter has previously appeared in

Deep Learning-powered Iterative Combinatorial Auctions.
Jakob Weissteiner and Sven Seuken.
In *Proceedings of the Thirty-fourth AAAI Conference on Artificial Intelligence* (AAAI'20), New York, USA, February 2020.

For its full updated version including appendix, please see

Deep Learning-powered Iterative Combinatorial Auctions.
Jakob Weissteiner and Sven Seuken.
Working paper, March 2023.
URL: arxiv.org/pdf/1907.05771.pdf

Deep Learning–powered Iterative Combinatorial Auctions*

Jakob Weissteiner
University of Zurich & ETH AI Center
weissteiner@ifi.uzh.ch

Sven Seuken
University of Zurich & ETH AI Center
seuken@ifi.uzh.ch

Abstract

In this paper, we study the design of deep learning-powered iterative combinatorial auctions (ICAs). We build on prior work where preference elicitation was done via kernelized support vector regressions (SVRs). However, the SVR-based approach has limitations because it requires solving a machine learning (ML)-based winner determination problem (WDP). With expressive kernels (like gaussians), the ML-based WDP cannot be solved for large domains. While linear or quadratic kernels have better computational scalability, these kernels have limited expressiveness. In this work, we address these shortcomings by using deep neural networks (DNNs) instead of SVRs. We first show how the DNN-based WDP can be reformulated into a mixed integer program (MIP). Second, we experimentally compare the prediction performance of DNNs against SVRs. Third, we present experimental evaluations in two medium-sized domains which show that even ICAs based on relatively small-sized DNNs lead to higher economic efficiency than ICAs based on kernelized SVRs. Finally, we show that our DNN-powered ICA also scales well to very large CA domains.

1 Introduction

Combinatorial auctions (CAs) are used to allocate multiple heterogeneous items to bidders in domains where these items may be substitutes or complements. Specifically, in a CA, bidders are allowed to submit bids on *bundles* of items rather than on individual items. CAs are widely used in practice, including for the sale of airport landing and take-off slots (Rassenti, Smith, and Bulfin 1982), in industrial procurement (Bichler et al. 2006), and for the sale of spectrum licenses (Cramton 2013).

One of the main challenges in large CAs is that the bundle space grows exponentially in the number of items. This typically makes it impossible for the bidders to report their full value function, even for medium-sized domains. Thus, careful preference elicitation is needed in CAs.

Nisan and Segal (2006) have shown that to achieve full efficiency and support general value functions, exponential communication in the number of items is needed in the worst case. Thus, practical auction designs cannot provide efficiency guarantees in large CA domains. Instead, many recent proposals for CAs have focused on *iterative combinatorial auctions (ICAs)* where the auctioneer interacts with bidders over multiple rounds, eliciting a *limited* amount of information, aiming to find a highly efficient allocation.

ICAs have found widespread application in practice. For example, just between 2008 and 2014, the combinatorial clock auction (CCA) (Ausubel, Cramton, and Milgrom 2006) has been used to conduct more than 15 spectrum auctions and has generated more than $20 Billion in total revenue (Ausubel and Baranov 2017). Another important application of ICAs are auctions for building offshore wind farms (Ausubel and Cramton 2011). Given the value of the resources allocated in these real-world ICAs, increasing their efficiency by 1-2% points already translates into welfare gains of millions of dollars. Therefore, improving the efficiency of ICAs is an important research challenge.

1.1 Machine Learning and Mechanism Design

Researchers have proposed various ways to further increase the efficiency of CAs by integrating machine learning (ML) methods into the mechanism. This research goes back to Blum et al. (2004) and Lahaie and Parkes (2004), who studied the relationship between computational learning theory and preference elicitation in CAs. More recently, Brero and Lahaie (2018) and Brero, Lahaie, and Seuken (2019) introduced a Bayesian CA where they integrated ML into a CA to achieve faster convergence. In a different strand of research, Dütting et al. (2015; 2019), Narasimhan, Agarwal, and Parkes (2016) and Golowich, Narasimhan, and Parkes (2018) used ML to directly learn a new mechanism (following the automated mechanism design paradigm).

Most related to the present paper is the work by Brero, Lubin, and Seuken (2017; 2018; 2019), who proposed an *ML-powered ICA*. The core of their auction is an *ML-powered preference elicitation algorithm*. As part of their algorithm, they used kernelized support vector regressions (SVRs) to learn the nonlinear value functions of bidders. Recently, Brero, Lubin, and Seuken (2019) showed that their ML-based ICA achieves even higher efficiency than the CCA. However, because of runtime complexity issues, Brero, Lubin, and Seuken (2018; 2019) focused on SVRs with linear and quadratic kernels. This leaves room for improvement, since bidders' valuations can have more complex structures

*This paper is the slightly updated version of Weissteiner and Seuken (2020) published at AAAI'20 including the appendix.

than can be captured by linear or quadratic kernels.

1.2 Our Approach Using Deep Neural Networks

In this paper, we propose using DNNs instead of SVRs in ML-powered ICAs. In each round of the auction, we approximate bidders' value functions by DNNs and subsequently solve an optimization problem, a *DNN-based winner determination problem (WDP)* , to determine which query to ask every bidder in the next round. Since our design involves doing this in each round of the auction, a central requirement for the practical implementation of the auction mechanism is to efficiently solve these DNN-based WDPs. Therefore, we show how to reformulate the WDP based on DNNs with rectified linear units (ReLUs) as activation functions into a (linear) mixed integer program (MIP) (Section 4).

Our approach is related to a recent line of research that uses MIP formulations to study specific properties of DNNs. For example, Cheng, Nührenberg, and Ruess (2017) studied resilience properties of DNNs. Similarly, Fischetti and Jo (2018) used a MIP formulation for finding adversarial examples in image recognition.

To experimentally evaluate the performance of our DNN-based approach, we use the Spectrum Auction Test Suite (SATS) (Weiss, Lubin, and Seuken 2017) to generate synthetic CA instances (Section 5). We first compare the prediction performance of DNNs against SVRs in the two medium-sized domains GSVM and LSVM. Then we compare the economic efficiency of our DNN-powered ICA against the SVR-powered ICA. In GSVM (a domain perfectly suited for the quadratic kernel), our DNN-powered ICA matches the efficiency of the SVR-powered ICA, while in the more complex LSVM domain, our DNN-powered ICA outperforms the SVR-powered ICA by 1.74% points. Finally, we also demonstrate that our DNN-based approach scales well to a very large domain, by evaluating it in the MRVM domain (with 98 items and 10 bidders). Overall, our results show that, perhaps surprisingly, even small-sized neural networks can be advantageous for the design of ICAs.

2 Preliminaries

We now present our formal model and review the ML-powered ICA by Brero, Lubin, and Seuken (2018).[1]

2.1 Iterative Combinatorial Auction

We consider a CA setting with n bidders and m indivisible items. Let $N := \{1, \ldots, n\}$ and $M := \{1, \ldots, m\}$ denote the set of bidders and items, respectively. We denote by $x \in \mathcal{X} := \{0,1\}^m$ a bundle of items represented as an indicator vector, where $x_j = 1$ iff item $j \in M$ is contained in x. Bidders' true preferences over bundles are represented by their (private) value functions $v_i : \{0,1\}^m \to \mathbb{R}_+$, $i \in N$, i.e., $v_i(x)$ represents bidder i's true value for bundle x. Let $v := (v_1, \ldots, v_n)$ denote the vector of bidders' value functions. The (possibly untruthful) reported valuations are denoted by $\hat{v}_i$ and $\hat{v}$, respectively.

By $a := (a_1, \ldots, a_n) \in \mathcal{X}^n$ we denote an allocation of bundles to bidders, where $a_i \in \mathcal{X}$ is the bundle bidder i obtains. An allocation a is *feasible* if each item is allocated to at most one bidder, i.e., $\forall j \in M : \sum_{i \in N} a_{ij} \leq 1$. We denote the set of feasible allocations by $\mathcal{F} := \left\{a \in \mathcal{X}^n : \sum_{i \in N} a_{ij} \leq 1,\ \forall j \in M\right\}$. Payments are denoted by $p = (p_1, \ldots, p_n) \in \mathbb{R}^n$, where p_i is bidder i's payment. Furthermore, we assume that bidders have quasilinear utility functions $u_i(a) := v_i(a_i) - p_i$. The (true) *social welfare* of an allocation a is defined as $V(a) := \sum_{i \in N} v_i(a_i)$. Let $a^* \in \arg\max_{a \in \mathcal{F}} V(a)$ be a feasible, social-welfare maximizing, i.e., *efficient*, allocation given true value functions v. Then the efficiency of any feasible allocation $a \in \mathcal{F}$ is measured in terms of a^* by $\frac{V(a)}{V(a^*)}$.

An ICA *mechanism* defines how the bidders interact with the auctioneer, how the final allocation is determined, and how payments are computed. In this paper, we only consider ICAs that ask bidders to iteratively report their valuations $\hat{v}_i(x)$ for particular bundles x selected by the mechanism. A finite set of such reported bundle-value pairs of bidder i is denoted as $B_i := \left\{\left(x^{(k)}, \hat{v}_i(x^{(k)})\right)\right\}_{k \in \{1,\ldots,n_i\}}$, $n_i \in \mathbb{N}$, $x^{(k)} \in \mathcal{X}$, where n_i is the total number of bundle-value pairs reported by bidder i. We let $B := (B_1, \ldots, B_n)$ denote the tuple of reported bundle-value pairs obtained from all bidders. We define the *reported social welfare* of an allocation a given B as

$$\widehat{V}(a|B) := \sum_{i \in N:\, (a_i, \hat{v}_i(a_i)) \in B_i} \hat{v}_i(a_i), \tag{1}$$

where the condition $(a_i, \hat{v}_i(a_i)) \in B_i$ ensures that only values for reported bundles contribute to the sum. Finally, the optimal feasible allocation a^*_B given B is defined as

$$a^*_B \in \arg\max_{a \in \mathcal{F}} \widehat{V}(a|B). \tag{2}$$

In the ICA mechanisms we consider in this paper, the final outcome is only computed based on the reported values B at termination. Specifically, the mechanism determines a feasible allocation $a^*_B \in \mathcal{F}$ and charges payments p.

As the auctioneer can generally only ask each bidder i for a limited number of bundle-value pairs B_i, the ICA mechanism needs a sophisticated preference elicitation algorithm. This leads to the following preference elicitation problem, where the goal is to find an (approximately) *efficient* allocation with a limited number of value queries. More formally:

Problem 1 (PREFERENCE ELICITATION IN ICA)**.** *Given a cap c_e on the number of value queries in an ICA, elicit from each bidder $i \in N$ a set of reported bundle-value pairs B_i with $|B_i| \leq c_e$ such that the resulting efficiency of a^*_B is maximized, i.e.,*

$$B \in \arg\max_{B : |B_i| \leq c_e} \frac{V(a^*_B)}{V(a^*)}. \tag{3}$$

In practice, a small domain-dependent cap on the number of queries is chosen, e.g., $c_e \leq 500$.

[1] We compare our DNN-powered ICA against the mechanism described in (Brero, Lubin, and Seuken 2018) because, when we wrote this paper, (Brero, Lubin, and Seuken 2019) was not available yet. We slightly adopt the notation and use B_i instead of $\widehat{\vartheta}_i$.

2.2 SVR-powered ICA

We now present a brief review of the ML-based ICA introduced by Brero, Lubin, and Seuken (2018). At the core of their auction is an *ML-based preference elicitation algorithm* which we reprint here as Algorithm 1.

Algorithm 1: ML-BASED ELICITATION (Brero et al. 2018)

Parameter : Machine learning algorithm $\mathcal{A}$
1 B^0 = initial tuple of reported bundle-value pairs at $t = 0$
2 **do**
3 $\quad t \leftarrow t + 1$
4 $\quad$ *Estimation step:* $\tilde{V}^t := \mathcal{A}(B^{t-1})$
5 $\quad$ *Optimization step:* $a^{(t)} \in \arg\max_{a \in \mathcal{F}} \tilde{V}^t(a)$
6 $\quad$ **for** each bidder i **do**
7 $\quad\quad$ **if** $a_i^{(t)} \notin B_i^{t-1}$ **then**
8 $\quad\quad\quad$ Query value $\hat{v}_i(a_i^{(t)})$
9 $\quad\quad\quad$ $B_i^t = B_i^{t-1} \cup \left\{ \left(a_i^{(t)}, \hat{v}_i(a_i^{(t)}) \right) \right\}$
10 $\quad\quad$ **else**
11 $\quad\quad\quad$ $B_i^t = B_i^{t-1}$
12 $\quad\quad$ **end**
13 $\quad$ **end**
14 **while** $\exists i \in N : a_i^{(t)} \notin B_i^{t-1}$
15 Output tuple of reported bundle-value pairs B^t

This algorithm is a procedure to determine B, i.e., for each bidder i a set of reported bundle-value pairs B_i. Note that the algorithm is described in terms of a generic ML algorithm $\mathcal{A}$ which is used in the *estimation step* (Line 4) to obtain the estimated social welfare function $\tilde{V}^t$ in iteration t. In the *optimization step* (Line 5), an ML-based *winner determination problem* is then solved to find an allocation $a^{(t)}$ that maximizes $\tilde{V}^t$. Finally, given the allocation $a^{(t)}$ from iteration t, each bidder i is asked to report his value for the bundle $a_i^{(t)}$. The algorithm stops when it reaches an allocation $a^{(t)}$ for which all bidders have already reported their values for the corresponding bundles $a_i^{(t)}$.

As the ML-algorithm $\mathcal{A}$, Brero, Lubin, and Seuken (2018) used a sum of kernelized SVRs, i.e,

$$\mathcal{A}(B^{t-1}) := \sum_{i \in N} \text{SVR}_i. \tag{4}$$

Given a bundle x, each SVR_i computes the predicted value as $\text{SVR}_i(x) = w_i \cdot \phi(x)$, where the weights w_i are determined through training on the reported bundle-value pairs B_i^{t-1}. Kernelized SVRs are a popular non-linear regression technique, where a linear model is fitted on transformed data. The transformation of bundles x is implicitly conducted by setting a kernel $k(x, x') := \phi(x)^T \phi(x')$ in the dual optimization problem (Smola and Schölkopf 2004).

Brero, Lubin, and Seuken (2018) called their entire auction mechanism the *Pseudo Vickrey-Clarke-Groves* mechanism (PVM). We reprint it here as Algorithm 2. PVM calls the preference elicitation algorithm (Algorithm 1) $n + 1$ times: once including all bidders (called the *main economy*) and n times excluding a different bidder in each run (called the *marginal economies*). The motivation for this design, which is inspired by the VCG mechanism, is to obtain payments such that the auction aligns bidders' incentives with allocative efficiency. Here, $B^{(-i)}$ denotes the output of Algorithm 1 by excluding bidder i from the set of bidders. For each of the reported bundle-value pairs $B^{(-i)}$ obtained from the $n + 1$ runs, PVM calculates a corresponding allocation that maximizes the *reported social welfare* (Line 2). The final allocation a^{pvm} is determined as the allocation of the $n + 1$ runs with the largest *reported social welfare* (Line 3). Finally, VCG-style payments are calculated (Line 4).

Algorithm 2: PVM (Brero et al. 2018)

1 Run Algorithm 1 $n + 1$ times: $B^{(-\emptyset)}, B^{(-1)}, \ldots, B^{(-n)}$.
2 Determine allocations: $a^{(-\emptyset)}, a^{(-1)}, \ldots, a^{(-n)}$, where $a^{(-i)} \in \arg\max_{a \in \mathcal{F}} \widehat{V}(a|B^{(-i)})$.
3 Pick $a^{pvm} \in \{a^{(-\emptyset)}, a^{(-1)}, \ldots, a^{(-n)}\}$ with maximal $\widehat{V}$.
4 Charge each bidder i according to:

$$p_i^{pvm} := \sum_{j \neq i} \hat{v}_j \left(a_j^{(-i)} \right) - \sum_{j \neq i} \hat{v}_j \left(a_j^{pvm} \right). \tag{5}$$

3 Deep Neural Network-powered ICA

In this section, we present the high level design of our DNN-powered ICA and discuss its advantages compared to the SVR-based design by Brero, Lubin, and Seuken (2018).

Observe that the choice of the ML algorithm $\mathcal{A}$ affects Algorithm 1 in two ways: first, in the estimation step (Line 4), $\mathcal{A}$ determines how well we can predict bidders' valuations; second, in the optimization step (Line 5), it determines the complexity of the ML-based WDP. Thus, our situation is different from standard supervised learning because of the added optimization step. In particular, when choosing $\mathcal{A}$, we must also ensure that we obtain a practically feasible ML-based WDP. Given that we have to solve the optimization step hundreds of times throughout an auction, in practice, we must impose a time limit on this step. In our experiments (Section 5), we follow Brero, Lubin, and Seuken (2018) and impose a 1 hour time limit on this step.

To make the optimization step feasible, Brero, Lubin, and Seuken (2018) used SVRs with quadratic kernels, for which the ML-based WDP is a quadratic integer program (QIP) and still practically solvable within a 1 hour time limit for most settings. However, note that a quadratic kernel, while more expressive than a linear kernel, can still at most model two-way interactions between the items. To this end, Brero, Lubin, and Seuken (2017; 2019) also evaluated more expressive kernels (gaussian and exponential). Even though these kernels had good prediction performance, the corresponding ML-based WDPs were too complex such that they always timed out and had a large optimization gap, thus leading to worse economic efficiency than the quadratic kernel. However, using SVRs with quadratic kernels leaves room for improvement, since bidders' valuations can be more complex than can be captured by quadratic kernels.

In this work, we show how these shortcomings can be addressed by using DNNs instead of SVRs in the estimation and optimization steps of Algorithm 1. DNNs are a concatenation of affine and non-linear mappings (see Figure 1). They consist of several layers, which are themselves composed of multiple nodes. Between each of the layers an affine transformation is applied, which is followed by a non-linear mapping called the *activation function*.

One advantage of DNNs compared to (nonlinear) kernelized SVRs is that, for any number of layers and nodes, we always obtain a (linear) MIP for the DNN-based WDP whose size only grows linearly in the number of bidders and items (as we will show in Section 4). The key insight for this is to use *rectified linear units (ReLUs)* as activation functions. Furthermore, in contrast to SVRs, DNNs do not use predefined feature transformations. While with SVRs, the choice of a good kernel usually relies on prior domain knowledge, DNNs automatically learn features in the process of training.

Following Brero, Lubin, and Seuken (2018), we decompose the estimated social welfare function in Line 4 of Algorithm 1, $\tilde{V}^t = \mathcal{A}(B^{t-1})$, as follows:

$$\tilde{V}^t = \sum_{i \in N} \tilde{v}_i^t, \tag{6}$$

where $\tilde{v}_i^t$ is an estimate of bidder i's true value function v_i and is trained on the data set B_i^{t-1}, i.e., the values queried up to round $t-1$. In this work, for every $i \in N$, we model $\tilde{v}_i^t$ using a *fully connected feed-forward DNN* $\mathcal{N}_i : \{0,1\}^m \to \mathbb{R}_+$. Consequently, the estimated social welfare function $\tilde{V}^t$ is given as a sum of DNNs, i.e.,

$$\tilde{V}^t := \sum_{i \in N} \mathcal{N}_i. \tag{7}$$

Note that each bidder's value function is modeled as a distinct DNN (with different architectures and parameters) because bidders' preferences are usually highly idiosyncratic.[2]

4 MIP Formulation of the DNN-based Winner Determination Problem

We now present the parameterization of each DNN and show how to reformulate the DNN-based WDP into a MIP. Thus, we focus on the optimization step (Line 5) of Algorithm 1.

4.1 Setting up the DNN-based WDP

Figure 1 shows a schematic representation of a DNN $\mathcal{N}_i$. We now define the parameters of the DNNs. To simplify the exposition, we consider a fixed iteration step t and no longer highlight the dependency of all variables on t.

Each DNN $\mathcal{N}_i$ consists of $K_i - 1$ *hidden layers* for $K_i \in \mathbb{N}$, with the kth hidden layer containing d_k^i hidden nodes, where $k \in \{1, \dots, K_i - 1\}$. As $\mathcal{N}_i$ maps bundles $x \in \{0,1\}^m$ to values, the dimension of the *input layer* d_0^i is equal to the number of items m (i.e., $d_0^i := m$) and the

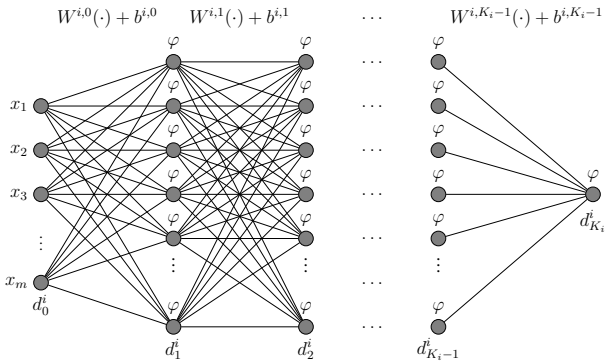


Figure 1: Schematic representation of a DNN $\mathcal{N}_i$.

dimension of the *output layer* is equal to 1 (i.e., $d_{K_i}^i := 1$). Hence, in total, a single DNN consists of $K_i + 1$ layers. Furthermore, let $\varphi : \mathbb{R} \to \mathbb{R}_+$, $\varphi(s) := \max(0, s)$ denote the ReLU activation function.[3] The affine mappings between the kth and the k+1st layer are parameterized by a matrix $W^{i,k} \in \mathbb{R}^{d_{k+1}^i \times d_k^i}$ and a bias $b^{i,k} \in \mathbb{R}^{d_{k+1}^i}$, $k \in \{0, \dots, K_i - 1\}$.

To estimate the parameters $W^{i,k}$ and $b^{i,k}$ from data (i.e., from the bundle-value pairs B_i) we use the *ADAM* algorithm, which is a popular gradient-based optimization technique (Kingma and Ba 2015).[4] This is done in the *estimation step* in Line 4 of Algorithm 1. Thus, after the estimation step, $W^{i,k}$ and $b^{i,k}$ are constants. In summary, given estimated parameters $W^i := \{W^{i,k}\}_{0 \le k \le K_i - 1}$ and $b^i := \{b^{i,k}\}_{0 \le k \le K_i - 1}$, each DNN $\mathcal{N}_i(W^i, b^i) : \{0,1\}^m \to \mathbb{R}_+$ represents the following nested function:

$$\begin{aligned} &\mathcal{N}_i(W^i, b^i)(x) = \\ &= \varphi\left(W^{i,K_i-1} \varphi\left(\dots \varphi(W^{i,0} x + b^{i,0}) \dots\right) + b^{i,K_i-1}\right). \end{aligned} \tag{8}$$

The *DNN-based WDP* in the optimization step in Line 5 of Algorithm 1 can now be formulated as follows:

$$\begin{aligned} &\max_{a \in \mathcal{X}^n} \left\{ \sum_{i \in N} \mathcal{N}_i\left(W^i, b^i\right)(a_i) \right\} && \text{(OP1)} \\ \text{s.t.} \quad &\sum_{i \in N} a_{ij} \le 1, \quad \forall j \in M \\ &a_{ij} \in \{0,1\}, \quad \forall j \in M, \forall i \in N. \end{aligned}$$

4.2 The MIP Formulation

In its general form, (OP1) is a nonlinear, non-convex optimization problem and there do not exist practically feasible algorithms that are guaranteed to find a globally optimal solution. Therefore, we now reformulate (OP1) into a MIP.

Consider bidder $i \in N$. For every layer $k \in \{1, \dots, K_i\}$ let $o^{i,k} \in \mathbb{R}_+^{d_k^i}$ denote the output of the kth layer, which can

[2] A second reason for this construction is that this prevents bidders from influencing each others' ML-models. Please see (Brero, Lubin, and Seuken 2019) for a detailed incentive analysis of PVM.

[3] φ acts on vectors componentwise, i.e., for $s \in \mathbb{R}^k$ let $\varphi(s) := (\varphi(s_1), \dots, \varphi(s_k))$.

[4] For fitting the DNNs in all of our experiments we use PYTHON 3.5.3, KERAS 2.2.4 and TENSORFLOW 1.13.1.

be recursively calculated as

$$o^{i,k} = \varphi(W^{i,k-1}o^{i,k-1} + b^{i,k-1}) = \\ = \max(0, W^{i,k-1}o^{i,k-1} + b^{i,k-1}). \tag{9}$$

For $k \in \{1, \dots, K_i\}$, we introduce d_k^i binary decision variables that determine which node in the corresponding layer is active, represented as a vector $y^{i,k} \in \{0,1\}^{d_k^i}$. We also introduce $2d_k^i$ continuous variables, represented as vectors $z^{i,k}$, $s^{i,k} \in \mathbb{R}^{d_k^i}$. Each $z^{i,k}$ corresponds to the positive components of the output value $o^{i,k}$ of each layer and each $s^{i,k}$ is used as a slack variable representing the absolute value of the negative components of $o^{i,k}$.

In our final MIP formulation, we will make use of *"big-M"* constraints. For our theoretical results to hold, we need to make the following standard assumption.

Assumption 1. (BIG-M CONSTRAINT) *For all $i \in N$ and $k \in \{1, ..., K_i\}$ there exists a large enough constant $L \in \mathbb{R}_+$, such that $\left|(W^{i,k-1}o^{i,k-1} + b^{i,k-1})_j\right| \leq L$ for $1 \leq j \leq d_k^i$.*

In the following lemma, we show how to recursively encode a layer of $\mathcal{N}_i$ given the output value of the previous layer as multiple linear constraints.[5]

Lemma 1. *Let $W^{i,k-1}o^{i,k-1} + b^{i,k-1} \in \mathbb{R}^{d_k^i}$ be fixed for a $k \in \{1, \dots, K_i\}$. Furthermore, let $z^{i,k}, s^{i,k} \in \mathbb{R}^{d_k^i}$, $y^{i,k} \in \{0,1\}^{d_k^i}$. Consider the following linear constraints:*

$$z^{i,k} - s^{i,k} = W^{i,k-1}o^{i,k-1} + b^{i,k-1} \tag{10}$$

$$0 \leq z^{i,k} \leq y^{i,k} \cdot L \tag{11}$$

$$0 \leq s^{i,k} \leq (1 - y^{i,k}) \cdot L. \tag{12}$$

The polytope defined by (10)-(12) is not empty and every element $\left(z^{i,k}, s^{i,k}, y^{i,k}\right)$ of this polytope satisfies $z^{i,k} = o^{i,k}$.

Proof. For notational convenience, let $c := W^{i,k-1}o^{i,k-1} + b^{i,k-1}$. Non-emptiness follows since $|c_j| \leq L$, $1 \leq j \leq d_k^i$, by Assumption 1. From Constraints (11) and (12) it follows, for $1 \leq j \leq d_k^i$, that if $z_j^{i,k} > 0$ then $s_j^{i,k} = 0$, and if $s_j^{i,k} > 0$ then $z_j^{i,k} = 0$. We now have to distinguish the following three cases for each component of c:

$$c_j < 0 \implies y_j^{i,k} = 0,\ s_j^{i,k} = -c_j, z_j^{i,k} = 0 = \varphi(c_j)$$
$$c_j > 0 \implies y_j^{i,k} = 1,\ s_j^{i,k} = 0, z_j^{i,k} = c_j = \varphi(c_j)$$
$$c_j = 0 \implies \left(z_j^{i,k}, s_j^{i,k}, y_j^{i,k}\right) \in \{(0,0,0), (0,0,1)\}$$

Combining all cases yields that $z^{i,k} = \varphi(c_j) = o^{i,k}$. □

Given Lemma 1, we can now reformulate the DNN-based WDP as a MIP. For this, we let W^i, b^i denote the estimated parameters corresponding to $\mathcal{N}_i(W^i, b^i)$. Furthermore, let $a \in \mathcal{X}^n, y^{i,k} \in \{0,1\}^{d_k^i}$ and $z^{i,k}$, $s^{i,k} \in \mathbb{R}^{d_k^i}$, for $1 \leq k \leq K_i$ and L be a constant satisfying Assumption 1.

[5] In the following, all constraints containing vectors are defined componentwise.

$$\max_{a \in \mathcal{X}^n, z^{i,k}, s^{i,k}, y^{i,k}} \left\{ \sum_{i \in N} z^{i,K_i} \right\} \tag{OP2}$$

s.t.

$$\left.\begin{aligned} & z^{i,0} = a_i \\ & z^{i,k} - s^{i,k} = W^{i,k-1}z^{i,k-1} + b^{i,k-1} \\ & 0 \leq z^{i,k} \leq y^{i,k} \cdot L \\ & 0 \leq s^{i,k} \leq (1 - y^{i,k}) \cdot L \\ & y^{i,k} \in \{0,1\}^{d_k^i} \end{aligned}\right\} \begin{aligned} & \forall i \in N \\ & \forall k \in \{1, \dots, K_i\} \end{aligned}$$

$$a_{ij} \in \{0,1\}, \qquad \forall j \in M, \forall i \in N$$

$$\sum_{i \in N} a_{ij} \leq 1, \qquad \forall j \in M$$

We are now ready to state our main theorem.

Theorem 1. (MIP FORMULATION) *The DNN-based WDP as defined in (OP1) is equivalent to the MIP defined in (OP2).*

Proof. Consider (OP1). For each bidder $i \in N$, we first set $z^{i,0}$ equal to the input bundle a_i. Then we proceed by using Lemma 1 for $k = 1$, i.e., we reformulate the output of the 1st layer as the linear constraints (10), (11) and (12). We iterate this procedure until we have reformulated the final layer, i.e, $k = K_i$. Doing so for each bidder $i \in N$ and adding the feasibility constraints yields (OP2). □

Using the MIP formulation (OP2), we can solve the DNN-based WDP using standard optimization packages like CPLEX.[6]

We now provide a simple worked example for how to reformulate (OP1) into (OP2) which illustrates Theorem 1.

Example 1. *Consider a setting with one bidder ($n = 1$), m items ($d_0^1 = m$) and one hidden layer ($K_1 = 2$). Given $W^{1,0} \in \mathbb{R}^{d_1^1 \times m}$, $W^{1,1} \in \mathbb{R}^{1 \times d_1^1}$, $b^{1,0} \in \mathbb{R}^{d_1^1}$, and $b^{1,1} \in \mathbb{R}$, (OP1) can be written as*

$$\max_{a_1 \in \mathcal{X}^1} \left\{ \max\left(0, W^{1,1} \max\left(0, W^{1,0}a_1 + b^{1,0}\right) + b^{1,1}\right) \right\}$$
$$\text{s.t.} \quad a_{1j} \in \{0,1\}, \quad \forall j \in M,$$

where the constraint $\sum_{i \in N} a_{ij} \leq 1$, $\forall j \in M$ is redundant in this case, since we only consider one bidder. First, we replace the inner maximum using $y^{1,1} \in \{0,1\}^{d_1^1}, z^{1,1}, s^{1,1} \in \mathbb{R}^{d_1^1}$ and arrive at the equivalent optimization problem

$$\max_{\substack{a_1 \in \mathcal{X}^1 \\ z^{1,1}, s^{1,1}, y^{1,1}}} \left\{ \max\left(0, W^{1,1}z^{1,1} + b^{1,1}\right) \right\}$$
$$\begin{aligned} \text{s.t.} \quad & z^{1,1} - s^{1,1} = W^{1,0}a_1 + b^{1,0} \\ & 0 \leq z^{1,1} \leq y^{1,1} \cdot L \\ & 0 \leq s^{1,1} \leq (1 - y^{1,1}) \cdot L \\ & a_{1j} \in \{0,1\}, \quad \forall j \in M. \end{aligned}$$

[6] For solving MIPs of the form (OP2) in our experiments we use CPLEX 12.8.0.0 with the python library DOCPLEX 2.4.61.

Applying Lemma 1 again by using $y^{1,2} \in \{0,1\}$ and $z^{1,2}, s^{1,2} \in \mathbb{R}$ yields the final MIP formulation

$$
\begin{aligned}
\max_{\substack{a_1 \in \mathcal{X}^1,\, z^{1,k} \\ y^{1,k},\, s^{1,k},\, k \in \{1,2\}}} \quad & \{z^{1,2}\} \\
\textit{s.t.} \quad & z^{1,1} - s^{1,1} = W^{1,0} a_1 + b^{1,0} \\
& 0 \leq z^{1,1} \leq y^{1,1} \cdot L \\
& 0 \leq s^{1,1} \leq (1 - y^{1,1}) \cdot L \\
& z^{1,2} - s^{1,2} = W^{1,1} z^{1,1} + b^{1,1} \\
& 0 \leq z^{1,2} \leq y^{1,2} \cdot L \\
& 0 \leq s^{1,2} \leq (1 - y^{1,2}) \cdot L \\
& a_{1j} \in \{0,1\}, \quad \forall j \in M.
\end{aligned}
$$

Remark 1. *The number of decision variables in the MIP defined in* (OP2) *is given by*

$$
\underbrace{\sum_{i \in N}}_{\textit{\#bidders}} \left(\underbrace{m}_{\textit{\#items}} + \underbrace{3}_{y^{i,k},\, s^{i,k},\, z^{i,k}} \cdot \left(\overbrace{\sum_{k=1}^{K_i - 1}}^{\textit{\#hidden layers}} \underbrace{d_k^i}_{\textit{\#nodes per layer}} + \overbrace{1}^{\textit{output}} \right) \right)
$$

5 Experimental Evaluation

In this section, we evaluate the performance of our deep learning-powered ICA and compare it against the SVR-based approach using quadratic kernels by Brero, Lubin, and Seuken (2018). We release our code under an open-source license at: https://github.com/marketdesignresearch/DL-ICA.

5.1 Experiment setup

Spectrum auctions are one of the most prominent applications of CAs, which is why we choose them for our experiments. Specifically, we use the spectrum auction test suite (SATS) version 0.6.4 (Weiss, Lubin, and Seuken 2017).[7] SATS enables us to generate 1000s of CA instances in different domains. Furthermore, we have access to each bidder's true value $v_i(x)$ for all 2^m possible bundles $x \in \mathcal{X}$ as well as the efficient allocation $a^* \in \mathcal{F}$, which we can use to measure the efficiency of any other allocation a by $V(a)/V(a^*)$. We evaluate our approach in the following three domains:

The Global Synergy Value Model (GSVM) (Goeree and Holt 2010) consists of 6 *regional bidders*, 1 *national bidder*, and 18 items. In GSVM the value of a package increases by a certain percentage with *every* additional item of interest. Thus, the value of a bundle only depends on the total number of items contained in a bundle which makes it one of the simplest models in SATS. In fact, bidders' valuations can be exactly learned by SVRs with quadratic kernels (Brero, Lubin, and Seuken (2019)) which implies that the valuations exhibit at most two-way interactions between items.

The Local Synergy Value Model (LSVM) (Scheffel, Ziegler, and Bichler 2012) consists of 5 *regional bidders*, 1 *national bidder* and 18 items. The items are arranged on a rectangle of size 3×6. The national bidder is interested in all items, while the regional bidders are only interested in certain subsets of items. Complementarities arise from spatial proximity of items and are modeled via a logistic function, which makes it more complex than GSVM.

The *Multi-Region Value Model* (MRVM) (Weiss, Lubin, and Seuken 2017) consists of 98 items and 10 bidders. It models large US and Canadian spectrum auctions and captures both geography (different regions) as well as frequency dimensions (different bands). A bidder is categorized as *national*, *regional* or *local*, depending on the magnitude of the synergies between different regions.

In Sections 5.2 and 5.3, we first evaluate our approach in detail using the two medium-sized domains GSVM and LSVM. Then, in Section 5.4, we use MRVM to evaluate how well our approach scales to very large domains.

5.2 Prediction Performance

We first compare the *prediction performance* of DNNs to SVRs. Using SATS, we generate a data set of bundle-value pairs $\{(x^{(k)}, v_i(x^{(k)}))\}$ for all bidders $i \in N$. For each auction domain, we draw 100 auction instances uniformly at random. For each such instance, we sample, for each bidder type, a training set T of equal size and a disjoint test set V consisting of all remaining bundles, i.e., $|V| := 2^{|M|} - |T|$. For each bidder type, we train the ML algorithm on T and test it on V. We report the mean absolute error (MAE) for both bidder types averaged over the 100 instances.[8]

We denote by $[d_1, d_2, d_3]$ a 3-hidden-layer DNN with d_1, d_2 and d_3 hidden nodes, respectively. For both, SVRs and DNNs, we performed a hyperparameter optimization for each bidder type. For the DNNs, we optimized the architecture[9], the L2-regularization parameter for the affine mappings, the dropout rate per layer, and the learning rate of ADAM. For SVRs with a quadratic kernel $k(x, y) := x^T y + \gamma (x^T y)^2$, we optimized γ (i.e., the influence of the quadratic term), the regularization parameter C, and the loss function parameter ϵ. In what follows, we present the winner models resulting from this hyperparameter optimization.

In Table 1, we present prediction performance results for the GSVM domain. Consider the last column of the table, which shows the MAE on the test set. We observe the very good prediction performance of the SVRs and in particular that the test error converges to 0 when increasing $|T|$. This is due to the fact that in GSVM, bidders' value functions can be perfectly captured by quadratic kernels. In this sense, GSVM represents a "worst case" auction domain w.r.t. our comparison of DNNs against quadratically-kernelized SVRs. Looking at the performance of the DNNs, we observe that the test error also decreases with $|T|$, but, not surprisingly, is always larger than for the SVRs with quadratic kernels. Furthermore, we observe that the optimal architectures are always a 1-hidden layer network.

In Table 2, we present the results for the more complex LSVM domain. We observe that for $|T| = 50$, the DNNs and

[7] Experiments were conducted on machines with Intel Xeon E5-2650 v4 2.20GHz processors with 20 cores.

[8] For training the DNNs, we use the MAE as the loss function.

[9] We considered the following architectures: for the national bidders: [10], [100], [10,10], . . ., [100,100,100,100], and for the regional bidders: [32], [100], [32,32], . . ., [100,100,100,100].

ML Algorithm	$\|T\|$	Bidder Type	DNN Architecture	MAE_{train}	MAE_{test}
DNNs	50	National	[100]	1.99	**5.25** (0.11)
		Regional	[100]	2.06	**6.20** (0.19)
	100	National	[100]	2.10	**3.66** (0.07)
		Regional	[100]	2.71	**4.64** (0.13)
	200	National	[100]	1.61	**2.22** (0.05)
		Regional	[100]	2.11	**2.89** (0.09)
			Kernel		
SVRs	50	National	quadratic	0.03	**4.38** (0.11)
		Regional	quadratic	0.03	**4.98** (0.20)
	100	National	quadratic	0.03	**1.71** (0.04)
		Regional	quadratic	0.03	**2.07** (0.07)
	200	National	quadratic	0.03	**0.12** (0.00)
		Regional	quadratic	0.03	**0.13** (0.00)

Table 1: Prediction performance in GSVM. All results are averaged over 100 auction instances. For MAE_{test}, standard errors are shown in parentheses.

ML Algorithm	$\|T\|$	Bidder Type	DNN Architecture	MAE_{train}	MAE_{test}
DNNs	50	National	[10]	24.68	**29.90** (0.23)
		Regional	[100]	4.22	**16.58** (0.39)
	100	National	[10, 10, 10]	9.01	**25.62** (0.36)
		Regional	[100]	5.01	**13.74** (0.26)
	200	National	[10, 10]	10.52	**20.58** (0.21)
		Regional	[100, 100, 100]	3.64	**11.27** (0.23)
			Kernel		
SVRs	50	National	quadratic	18.51	**32.61** (0.59)
		Regional	quadratic	3.11	**15.30** (0.34)
	100	National	quadratic	20.03	**27.86** (0.28)
		Regional	quadratic	3.21	**14.21** (0.28)
	200	National	quadratic	20.03	**25.44** (0.16)
		Regional	quadratic	8.23	**12.67** (0.26)

Table 2: Prediction performance in LSVM. All results are averaged over 100 auction instances. For MAE_{test}, standard errors are shown in parentheses.

SVRs with quadratic kernels have similar test error. But for $|T| = 100$ and $|T| = 200$, the DNNs significantly outperform the SVRs with quadratic kernels. Specifically, DNNs better capture the national bidder in LSVM, which is important, since this bidder is interested in all items and usually gets a large portion of the items in the final allocation, which matters a lot for efficiency. In contrast to GSVM, we observe that for $|T| \geq 100$, multi-hidden-layer networks were found to be best. This suggests that DNNs may indeed be advantageous for capturing more complex preference structures.[10]

5.3 Efficiency Results

Finally, we compare the economic efficiency of our DNN-powered ICA against the SVR-powered ICA. When conducting the efficiency experiments, we follow Brero, Lubin, and Seuken (2018) and assume that bidders answer all value queries truthfully (i.e., $\hat{v}_i = v_i$). Furthermore, we also use their experiment setup and define a cap c_e on the total number of value queries in Algorithm 1 and set $c_e := 50$. The initial set of reported bundle-value pairs B_i^0 per bidder i is drawn uniformly at random. We denote the number of initial reports by $c_0 := |B_i^0|, \forall i \in N$, resulting in a maximum of $c_0 + n \cdot (c_e - c_0)$ queries per bidder.

[10]We also evaluated the prediction performance of other kernels (linear, gaussian and exponential) and observed that DNNs were as good or better for almost all combinations of bidder types and $|T|$.

DNN Architectures[11]	c_0	Efficiency %	Revenue %[12]
R:[16, 16] | N:[10, 10]	40	98.53%	69.26%
R:[16, 16] | N:[10, 10]	30	98.41%	68.29%
R:[16, 16] | N:[10, 10, 10]	40	98.51%	68.91%
R:[16, 16] | N:[10, 10, 10]	30	98.32%	68.59%
R:[32, 32] | N:[10, 10]	40	98.75%	71.14%
R:[32,32] | N:[10,10]	**30**	**98.94%**	**68.47%**
R:[32, 32] | N:[10, 10, 10]	40	98.69%	71.63%
R:[32, 32] | N:[10, 10, 10]	30	98.92%	68.88%
R:[100] | N:[100]	30	98.27%	66.73%

Table 3: Efficiency results for 9 configurations of a DNN-powered ICA on a training set of 100 GSVM auction instances. The selected winner model is marked in bold. All results are averaged over the 100 auction instances.

Auction Mechanism	#Queries	Max #Queries	% Efficiency	% Revenue	t-test on Efficiency[13]
VCG	2^{18}	2^{18}	100.00 (0.00)	80.4	-
SVR-ICA	41.9	42.8	**98.85** (0.13)	77.80	0.337
DNN-ICA	53	78	**98.63** (0.18)	67.81	

Table 4: A comparison of the DNN-powered ICA against the SVR-powered ICA and VCG (as reported in Brero, Lubin, and Seuken (2018)) on a test set of 100 GSVM instances. All results are averaged over the 100 instances. For efficiency, standard errors are shown in parentheses.

GSVM. In Table 3, we first present the results from comparing nine different network architectures on a training set of 100 GSVM instances. As we can see, the winning model is among the largest multi-layer networks we tested. It is noteworthy that the one-hidden-layer network (R:[100]|N:[100]), which performed best in terms of prediction performance, did not perform best in terms of efficiency.

In Table 4, we compare the performance of the winner model from Table 3 (see Appendix A for configuration details) against the SVR-based approach on a test set of 100 GSVM instances. Even though GSVM can be perfectly captured by quadratic kernels, our DNN-based approach achieves a similar result w.r.t. efficiency, where the difference in means is not statistically significant ($p = 0.337$).

[11]We denote by R and N the architectures used for the regional- and national bidders, respectively.

[12]Revenue is calculated as $(\sum_{i \in N} p_i^{pvm})/V(a^*)$.

[13]We performed a two-sided unpaired *Welch Two Sample t-test* with $\mathcal{H}_0 : \mu_1 = \mu_2$ against $\mathcal{H}_A : \mu_1 \neq \mu_2$. We thank Brero, Lubin, and Seuken (2018) for providing us with the detailed results of their experiments to enable all t-tests reported in this paper.

DNN Architectures	c_0	Efficiency %	Revenue %
R:[16, 16] | N:[10, 10]	40	97.40	60.51
R:[16, 16] | N:[10, 10]	30	96.87	56.85
R:[16, 16] | N:[10, 10, 10]	40	97.45	61.15
R:[16, 16] | N:[10, 10, 10]	30	97.12	59.31
R:[32, 32] | N:[10, 10]	40	97.40	62.01
R:[32, 32] | N:[10, 10]	30	96.83	59.07
R:[32,32] | N:[10,10,10]	**40**	**97.74**	**61.95**
R:[32, 32] | N:[10, 10, 10]	30	97.12	59.56
R:[100] | N:[10]	40	96.78	58.71

Table 5: Efficiency results for 9 configurations of a DNN-powered ICA on a training set of 100 LSVM auction instances. The selected winner model is marked in bold. All results are averaged over the 100 auction instances.

Auction Mechanism	#Queries	Max #Queries	% Efficiency	% Revenue	t-test on Efficiency
VCG	2^{18}	2^{18}	100.00 (0.00)	83.4	-
SVR-ICA	48.2	52.8	**96.03** (0.33)	65.60	4e−5
DNN-ICA	65	77	**97.74** (0.24)	62.45	

Table 6: A comparison of the DNN-powered ICA against the SVR-powered ICA and VCG (as reported in Brero, Lubin, and Seuken (2018)) on a test set of 100 LSVM auction instances. All results are averaged over the 100 instances. For efficiency, standard errors are shown in parentheses.

LSVM. We now turn to the more complex LSVM domain. We first select a winner model based on a training set of 100 LSVM instances (Table 5). As in GSVM, the best model is among the largest architectures and the best model w.r.t. prediction performance does not yield the highest efficiency.

In Table 6, we compare the performance of the selected winner model from Table 5 (see Appendix A for configuration details) against the SVR-based approach on a test set of 100 new auction instances. Here, we see that our DNN-powered ICA substantially outperforms the SVR-powered ICA by 1.71% points, and that the difference in means is highly statistically significant ($p = 4\text{e}{-5}$). This demonstrates the advantage of DNNs over SVRs with quadratic kernels in complex domains like LSVM.

In Figure 2, we present a histogram of the efficiency obtained by the selected winner model on the test set. We see that for 29 auction instances, our approach (impressively) obtains an economic efficiency of 100%. However, for two instances, the efficiency is less than 90%. Thus, it is a promising avenue for future work to investigate these outliers to further increase the average efficiency.

Remark 2. *In Tables 4 and 6, we see that our DNN-based approach achieves lower revenue than the SVR-based approach. This may be explained as follows. A bidder's payment in PVM, depends on the difference between the social welfare in the marginal and the main economy. However, PVM has one oddity: a bidder's payment may be negative. This happens more frequently with DNNs than with SVRs: consider an auction where bidder i is not allocated in the main economy. Then, ideally, the allocation (and thus the welfare) should be exactly the same in the marginal economy where bidder i is excluded, resulting in a zero payment. When using SVRs, this is guaranteed if the same set of bundle-value pairs is used in the main and marginal economy, because SVRs use a deterministic algorithm for estimation. In contrast, DNNs use a non-deterministic algorithm, sometimes resulting in different allocations in the main and marginal economies. However, this is more a limitation of PVM itself. In practice, one should lower bound the payments as also suggested by Brero, Lubin, and Seuken (2019). Lower-bounding all payments by zero increases the revenue in GSVM by 7.9% points and in LSVM by 8.3% points.*

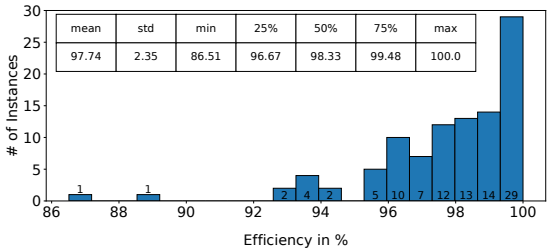


Figure 2: Histogram of efficiency results in LSVM of the selected DNN winner model on the test set.

5.4 Scaling to Larger Domains

We now present results for the MRVM domain (with 98 items and 10 bidders) to show that our DNN-powered ICA also scales well to larger domains (we present detailed runtime results in Section 5.5). We use the experiment setup of Brero, Lubin, and Seuken (2018) and set $c_e := 100$.

In Table 7, we present the results for different DNN architectures and different values of c_0, evaluated on a training set of MRVM instances. First, we observe that the efficiency increases as we decrease c_0. This can be explained by the fact that a smaller c_0 tends to lead to more iterations of the preference elicitation algorithm, resulting in a larger number of elicited bundle-value pairs. In terms of which DNN architectures performed better or worse, no clear pattern emerged.

In Table 8, we compare the performance of the selected winner model from Table 7 (see Appendix A for config-

DNN Architectures	c_0	Efficiency %	Revenue %
L:[10, 10] | R:[32, 32] | N:[32, 32]	70	93.54	31.02
L:[10, 10] | R:[32, 32] | N:[32, 32]	50	94.07	33.51
L:[16, 16] | R:[16, 16] | N:[16, 16]	30	94.46	31.39
L:[10, 10] | R:[32, 32] | N:[32, 32]	30	94.73	31.88
L:[16, 16] | R:[16, 16] | N:[16, 16]	20	94.88	30.31
L:[10, 10] | R:[32, 32] | N:[32, 32]	20	94.42	34.23
L:[16,16]| R:[16,16] | N:[16,16]	**10**	**95.00**	**31.97**
L:[10, 10] | R:[32, 32] | N:[32, 32]	10	94.54	34.78
L:[10, 10] | R:[16, 16, 16] | N:[16, 16, 16]	10	94.74	31.12

Table 7: Efficiency results for 9 configurations of a DNN-powered ICA on a training set of 19 MRVM auction instances. The selected winner model is marked in bold. All results are averaged over the 19 auction instances.

Auction Mechanism	#Queries	Max #Queries	% Efficiency	% Revenue	t-test on Efficiency
VCG	2^{98}	2^{98}	100.00 (0.00)	44.3	-
SVR-ICA	265	630	**94.58** (0.14)	35.20	0.0268
DNN-ICA	334	908	**95.04** (0.14)	30.59	

Table 8: A comparison of the DNN-powered ICA against the SVR-powered ICA and VCG (as reported in Brero, Lubin, and Seuken (2018)) on a test set of 50 MRVM auction instances. All results are averaged over the 50 instances. For efficiency, standard errors are shown in parentheses.

uration details) against the SVR-based approach on a test set of 50 MRVM instances. We see that our DNN-powered ICA outperforms the SVR-powered ICA by 0.46% points. While this is a relatively modest increase in efficiency, a t-test shows that the difference in means is statistically significant ($p = 0.0268$). We also observe that our DNN-based approach (while obeying all caps) asks a larger number of queries than the SVR-based approach. It is unclear how much of the efficiency increase is driven by the DNN or by the larger number of queries. Future work should compare the two approaches by holding the total number of queries constant.

5.5 Runtime Analysis

In Table 9, we present runtime results for our DNN-powered ICA for the winner models from Tables 4, 6 and 8. Specifically, we show average runtime results of the MIP (OP2), of an iteration of Algorithm 1, and of a whole auction (PVM). We observe that the average runtime of a whole auction takes approximately 1 hour in GSVM and LSVM and 8 hours in the larger MRVM domain. The increase in total runtime in MRVM can be explained by the fact that we use a smaller number of initial queries ($c_0 := 10$) and a larger total query cap ($c_e := 100$) compared to LSVM and GSVM. This results in a larger number of iterations of Algorithm 1. Additionally, MRVM consists of 10 bidders resulting in 11 calls of Algorithm 1 in contrast to 7 calls in LSVM and 8 in GSVM. Even though in MRVM the average MIP runtime is smaller, the larger number of iterations and bidders lead to this increase in total runtime. Overall, these results show that our DNN-based approach is practically feasible and scales well to the larger MRVM domain. Brero, Lubin, and Seuken (2018) do not provide runtime information such that we cannot provide a runtime comparison with SVRs.[14]

In Figure 3, we present additional MIP runtime results for selected DNN architectures in LSVM (results in GSVM and MRVM are qualitatively similar). We observe two effects: First, increasing the number of nodes per layer slightly increases the average runtime. Second, adding an additional layer (for the national bidder) significantly increases the average runtime. Not surprisingly, the largest DNN architectures lead to the highest runtime.

[14]In conversations with the authors, they told us that for gaussian and exponential kernels, their MIPs always timed out (1h cap) in GSVM, LSVM and MRVM. The average MIP runtime for the quadratic kernel was a few seconds in GSVM and LSVM. In MRVM, the quadratic kernel also regularly timed out resulting in an average MIP runtime of 30 min and of 36 h for a whole auction.

Domain	∅ MIP Runtime	∅ Iteration Runtime	∅ Auction Runtime
GSVM	15.90 sec	30.51 sec	44 min
LSVM	39.75 sec	51.69 sec	65 min
MRVM	3.67 sec	26.75 sec	457 min

Table 9: Average runtime results of the selected DNN winner models in different domains. All values are averaged over 100 (GSVM and LSVM) and 50 (MRVM) auction instances.

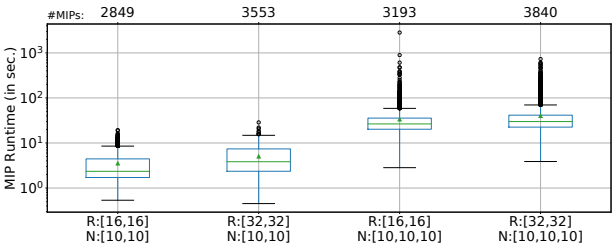


Figure 3: MIP runtimes defined in (OP2) based on 50 different LSVM instances. Results are shown for a selection of various DNN architectures and for $c_0 := 40, c_e := 50$.

The runtime of our MIPs heavily depends on the size of the "big-M" variable L. In practice, L should be chosen as small as possible to obtain a MIP formulation that is as tight as possible. We initialized $L := 3000$ and tightened this bound further by using interval arithmetic (see, e.g., Tjeng, Xiao, and Tedrake (2019)). Recently, Singh et al. (2018) proposed a novel technique for tightening such MIP formulations. Evaluating this in more detail is subject to future work.

6 Conclusion

In this paper, we have designed a deep learning-powered ICA. We have compared our approach against prior work using SVRs with quadratic kernels. Our experimental results have shown that our DNN-based approach leads to significantly higher economic efficiency in complex auction domains and scales well to large domains.

On a technical level, our main contribution was to reformulate the DNN-based WDP into a MIP. The main insight to achieve this was to use ReLU activation functions, which can be re-written as multiple linear constraints. From an experimental point of view, we were pleasantly surprised to see that even DNNs with a small number of layers and nodes and with a small number of training samples (i.e., bids) were able to achieve high prediction performance and ultimately high economic efficiency in the overall auction mechanism.

Future work should investigate the trade-off between larger DNN architectures and the resulting MIP runtime, to further increase efficiency.

7 Follow-Up Work After Publication

After the publication of the present paper, there has been a stream of follow-up papers, which have (a) further improved the ML capability of the mechanism, and (b) applied DNN-based preference elicitation to other combinatorial assignment domains.

Weissteiner et al. (2022b) designed a *Fourier analysis-based ICA* that leverages different notions of Fourier sparsity. This facilitates the intricate learning task in ICAs where only a few bids (i.e., training samples) can be elicited.

Weissteiner et al. (2022a) designed *monotone-value neural networks (MVNNs)*, a novel class of DNNs, which by design incorporate *free disposal*. They experimentally showed that MVNNs lead to better generalization performance (especially in settings with only a few bids) and also to higher efficiency (when integrated into an ICA).

Weissteiner et al. (2023) implemented uncertainty-based exploration for ICAs using a novel uncertainty quantification method for DNNs (Heiss et al. 2022). With their *Bayesian optimization-based combinatorial assignment (BOCA)* mechanism they could even further increase efficiency.

Soumalias et al. (2023) applied DNN-based preference elicitation in a combinatorial assignment domain different from that of combinatorial auctions. Concretely, Soumalias et al. (2023) designed an MVNN-based preference elicitation mechanism for course allocation.

Acknowledgments

We thank Gianluca Brero, Nils Olberg, and Stefania Ionescu for insightful discussions and the anonymous reviewers for helpful comments. This paper is part of a project that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No. 805542).

References

[2017] Ausubel, L. M., and Baranov, O. 2017. A practical guide to the combinatorial clock auction. *Economic Journal* 127(605):F334–F350. 1

[2011] Ausubel, L., and Cramton, P. 2011. Auction design for wind rights. *Report to Bureau of Ocean Energy Management, Regulation and Enforcement*. 1

[2006] Ausubel, L. M.; Cramton, P.; and Milgrom, P. 2006. *Combinatorial Auctions*. MIT Press. chapter The clock-proxy auction: A practical combinatorial auction design, 115–138. 1

[2006] Bichler, M.; Davenport, A.; Hohner, G.; and Kalagnanam, J. 2006. *Combinatorial Auctions*. MIT Press. chapter Industrial procurement auctions, 593–612. 1

[2004] Blum, A.; Jackson, J.; Sandholm, T.; and Zinkevich, M. 2004. Preference elicitation and query learning. *Journal of Machine Learning Research* 5:649–667. 1

[2018] Brero, G., and Lahaie, S. 2018. A bayesian clearing mechanism for combinatorial auctions. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*. 1

[2019] Brero, G.; Lahaie, S.; and Seuken, S. 2019. Fast iterative combinatorial auctions via bayesian learning. In *Proceedings of the 33rd AAAI Conference of Artificial Intelligence*. 1

[2017] Brero, G.; Lubin, B.; and Seuken, S. 2017. Probably approximately efficient combinatorial auctions via machine learning. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*. 1, 3

[2018] Brero, G.; Lubin, B.; and Seuken, S. 2018. Combinatorial auctions via machine learning-based preference elicitation. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence and the 23rd European Conference on Artificial Intelligence*. 1, 2, 3, 4, 6, 7, 8, 9

[2019] Brero, G.; Lubin, B.; and Seuken, S. 2019. Machine learning-powered iterative combinatorial auctions. *arXiv preprint arXiv:1911.08042*. 1, 2, 3, 4, 6, 8

[2017] Cheng, C.-H.; Nührenberg, G.; and Ruess, H. 2017. *Automated Technology for Verification and Analysis. Lecture Notes in Computer Science*, volume 10482. Springer, Cham. chapter Maximum resilience of artificial neural networks. 2

[2013] Cramton, P. 2013. Spectrum auction design. *Review of Industrial Organization* 42(2):161–190. 1

[2015] Dütting, P.; Fischer, F.; Jirapinyo, P.; Lai, J. K.; Lubin, B.; and Parkes, D. C. 2015. Payment rules through discriminant-based classifiers. *ACM Transactions on Economics and Computation* 3(1):5:1–5:41. 1

[2019] Dütting, P.; Feng, Z.; Narasimhan, H.; Parkes, D. C.; and Ravindranath, S. S. 2019. Optimal auctions through deep learning. In *Proceedings of the 36th International Conference on Machine Learning*. 1

[2018] Fischetti, M., and Jo, J. 2018. Deep neural networks and mixed integer linear optimization. *Constraints* 23(3):296–309. 2

[2010] Goeree, J. K., and Holt, C. A. 2010. Hierarchical package bidding: A paper & pencil combinatorial auction. *Games and Economic Behavior* 70(1):146–169. 6

[2018] Golowich, N.; Narasimhan, H.; and Parkes, D. C. 2018. Deep learning for multi-facility location mechanism design. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence and the 23rd European Conference on Artificial Intelligence*. 1

[2022] Heiss, J. M.; Weissteiner, J.; Wutte, H. S.; Seuken, S.; and Teichmann, J. 2022. NOMU: Neural optimization-based model uncertainty. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, 8708–8758. PMLR. 10

[2015] Kingma, D., and Ba, J. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*. 4

[2004] Lahaie, S. M., and Parkes, D. C. 2004. Applying learning algorithms to preference elicitation. In *Proceedings of the 5th ACM Conference on Electronic Commerce*. 1

[2016] Narasimhan, H.; Agarwal, S. B.; and Parkes, D. C. 2016. Automated mechanism design without money via

SATS		DNN								MIP				PVM	
Domain	Bidder	Epochs	Batch Size	L1&L2 Regularization	Learning Rate	Architecture	Dropout	Dropout Rate	MinMaxScaler[15]	Bounds Tightening	L	Time Limit (sec)	Relative Gap	c_0	c_e
GSVM	Regional	512	32	0.00001	0.01	[32,32]	True	0.05	False	IA	3000	3600	0.0001	30	50
	National	512	32	0.00001	0.01	[10,10]	True	0.05	False	IA	3000	3600	0.0001	30	50
LSVM	Regional	512	32	0.00001	0.01	[32,32]	True	0.05	False	IA	3000	3600	0.0001	40	50
	National	512	32	0.00001	0.01	[10,10,10]	True	0.05	False	IA	3000	3600	0.0001	40	50
MRVM	Local	300	32	0.00001	0.01	[16,16]	True	0.05	[0, 500]	IA	3000	3600	0.0001	10	100
	Regional	300	32	0.00001	0.01	[16,16]	True	0.05	[0, 500]	IA	3000	3600	0.0001	10	100
	National	300	32	0.00001	0.01	[16,16]	True	0.05	[0, 500]	IA	3000	3600	0.0001	10	100

Table 10: Detailed configurations of all DNN winner models for all SATS domains.

machine learning. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*. 1

[2006] Nisan, N., and Segal, I. 2006. The communication requirements of efficient allocations and supporting prices. *Journal of Economic Theory* 129(1):192–224. 1

[1982] Rassenti, S. J.; Smith, V. L.; and Bulfin, R. L. 1982. A combinatorial auction mechanism for airport time slot allocation. *The Bell Journal of Economics* 402–417. 1

[2012] Scheffel, T.; Ziegler, G.; and Bichler, M. 2012. On the impact of package selection in combinatorial auctions: an experimental study in the context of spectrum auction design. *Experimental Economics* 15(4):667–692. 6

[2018] Singh, G.; Gehr, T.; Mirman, M.; Püschel, M.; and Vechev, M. 2018. Fast and effective robustness certification. In *Proceedings of the 32nd Conference on Neural Information Processing Systems*. 9

[2004] Smola, A. J., and Schölkopf, B. 2004. A tutorial on support vector regression. *Statistics and computing* 14(3):199–222. 3

[2023] Soumalias, E.; Zamanlooy, B.; Weissteiner, J.; and Seuken, S. 2023. Machine learning-powered course allocation. *arXiv preprint arXiv:2210.00954*. 10

[2019] Tjeng, V.; Xiao, K. Y.; and Tedrake, R. 2019. Evaluating robustness of neural networks with mixed integer programming. In *Proceedings of the 7th International Conference on Learning Representations*. 9

[2017] Weiss, M.; Lubin, B.; and Seuken, S. 2017. Sats: A universal spectrum auction test suite. In *Proceedings of the 16th Conference on Autonomous Agents and Multi-Agent Systems*. 2, 6

[2020] Weissteiner, J., and Seuken, S. 2020. Deep learning—powered iterative combinatorial auctions. *Proceedings of the AAAI Conference on Artificial Intelligence* 34(02):2284–2293. 1

[2022a] Weissteiner, J.; Heiss, J.; Siems, J.; and Seuken, S. 2022a. Monotone-value neural networks: Exploiting preference monotonicity in combinatorial assignment. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, 541–548. International Joint Conferences on Artificial Intelligence Organization. Main Track. 10

[2022b] Weissteiner, J.; Wendler, C.; Seuken, S.; Lubin, B.; and Püschel, M. 2022b. Fourier analysis-based iterative combinatorial auctions. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, 549–556. International Joint Conferences on Artificial Intelligence Organization. Main Track. 10

[2023] Weissteiner, J.; Heiss, J.; Siems, J.; and Seuken, S. 2023. Bayesian optimization-based combinatorial assignment. *Proceedings of the AAAI Conference on Artificial Intelligence* 37. 10

Appendix

A Detailed Winner Configurations

In Table 10, we provide the detailed configurations (i.e., DNN, MIP, and PVM parameters) of the DNN winner models (i.e., DNN-ICA) from Table 4, Table 6 and Table 8. Other parameters not listed in Table 10 were set to their default values.

[15] We use SCIKIT LEARN'S *MinMaxScaler* to simultaneously scale bidders' value reports (i.e., $v_i(x^{(k)})$) in the generated (initial) training sets $\cup_{i=1}^{n} B_i^0$ to an interval $[0, u]$ with $u > 0$.

3 Fourier Analysis-based Iterative Combinatorial Auctions

The content of this chapter has previously appeared in

Fourier Analysis-based Iterative Combinatorial Auctions.
Jakob Weissteiner*, Chris Wendler*, Sven Seuken, Ben Lubin, and Markus Püschel.
In *Proceedings of the Thirty-first International joint Conference on Artificial Intelligence* (IJCAI'22), Vienna, AUT, July 2022.

For its full updated version including appendix, please see

Fourier Analysis-based Iterative Combinatorial Auctions.
Jakob Weissteiner*, Chris Wendler*, Sven Seuken, Ben Lubin, and Markus Püschel.
Working paper, March 2023.
URL: arxiv.org/pdf/2009.10749.pdf

*These authors contributed equally.

Fourier Analysis-based Iterative Combinatorial Auctions*

Jakob Weissteiner[1,4†], **Chris Wendler**[2,4†], **Sven Seuken**[1,4], **Ben Lubin**[3] and **Markus Püschel**[2,4]
[1]University of Zurich
[2]ETH Zurich
[3]Boston University
[4] ETH AI Center
weissteiner@ifi.uzh.ch, chris.wendler@inf.ethz.ch, seuken@ifi.uzh.ch, blubin@bu.edu, pueschel@inf.ethz.ch

Abstract

Recent advances in Fourier analysis have brought new tools to efficiently represent and learn set functions. In this paper, we bring the power of Fourier analysis to the design of combinatorial auctions (CAs). The key idea is to approximate bidders' value functions using Fourier-sparse set functions, which can be computed using a relatively small number of queries. Since this number is still too large for practical CAs, we propose a new hybrid design: we first use neural networks (NNs) to learn bidders' values and then apply Fourier analysis to the learned representations. On a technical level, we formulate a Fourier transform-based winner determination problem and derive its mixed integer program formulation. Based on this, we devise an iterative CA that asks Fourier-based queries. We experimentally show that our hybrid ICA achieves higher efficiency than prior auction designs, leads to a fairer distribution of social welfare, and significantly reduces runtime. With this paper, we are the first to leverage Fourier analysis in CA design and lay the foundation for future work in this area. Our code is available on GitHub: https://github.com/marketdesignresearch/FA-based-ICAs.

1 Introduction

Combinatorial auctions (CAs) are used to allocate multiple heterogeneous items to bidders. CAs are particularly useful in domains where bidders' preferences exhibit *complementarities* and *substitutabilities* as they allow bidders to submit bids on *bundles* of items rather than on individual items.

Since the bundle space grows exponentially in the number of items, it is impossible for bidders to report values for all bundles in settings with more than a modest number of items. Thus, parsimonious preference elicitation is key for the practical design of CAs. For general value functions, Nisan and Segal [2006] have shown that to guarantee full efficiency, exponential communication in the number of items is needed. Thus, practical CAs cannot provide efficiency guarantees in large domains. Instead, recent proposals have focused on *iterative combinatorial auctions (ICAs)*, where the auctioneer interacts with bidders over rounds, eliciting a *limited* amount of information, aiming to find a highly efficient allocation.

ICAs have found widespread application; most recently, for the sale of licenses to build offshore wind farms [Ausubel and Cramton, 2011]. For the sale of spectrum licenses, the combinatorial clock auction (CCA) [Ausubel *et al.*, 2006] has generated more than $20 billion in total revenue [Ausubel and Baranov, 2017]. Thus, increasing the efficiency by only 1–2% points translates into monetary gains of millions of dollars.

1.1 Machine Learning-based Auction Design

Recently, researchers have used machine learning (ML) to improve the performance of CAs. Early work by Blum *et al.* [2004] and Lahaie and Parkes [2004] first studied the relationship between learning theory and preference elicitation in CAs. Dütting *et al.* [2019], Shen *et al.* [2019] and Rahme *et al.* [2021] used neural networks (NNs) to learn whole auction mechanisms from data. Brero *et al.* [2019] introduced a Bayesian ICA using probabilistic price updates to achieve faster convergence. Shen *et al.* [2020] use reinforcement learning for dynamic pricing in sponsored search auctions. Most related to the present paper is the work by Brero *et al.* [2018; 2021], who developed a value-query-based ML-powered ICA using support vector regressions (SVRs) that achieves even higher efficiency than the CCA. In follow-up work, Weissteiner and Seuken [2020] extended their ICA to NNs, further increasing the efficiency. In work subsequent to the first version of this paper, Weissteiner *et al.* [2022a] proposed Monotone-Value Neural Networks (MVNNs), which are particularly well suited to learning value functions in combinatorial assignment domains.[1]

However, especially in large domains, it remains a challenge to find the efficient allocation while keeping the elicitation cost low. Thus, even state-of-the-art approaches suffer from significant efficiency losses and often result in unfair allocations, highlighting the need for better preference elicitation algorithms.

*This paper is the slightly updated version of Weissteiner *et al.* [2022b] published at IJCAI'22 including the appendix.

†These authors contributed equally to this paper.

[1]After the publication of the present paper, Weissteiner *et al.* [2023] integrated a notion of uncertainty [Heiss *et al.*, 2022] into an ML-powered ICA to balance the explore-exploit dilemma.

1.2 Combining Fourier Analysis and CAs

The goal of preference elicitation in CAs is to learn bidders' value functions using a small number of queries. Mathematically, value functions are *set functions*, which are in general exponentially large and notoriously hard to represent or learn. To address this complexity, we leverage Fourier analysis for set functions [Bernasconi *et al.*, 1996; O'Donnell, 2014; Püschel and Wendler, 2020]. In particular, we consider *Fourier-sparse approximations*, which are represented by few parameters. These parameters are the non-zero Fourier coefficients (FCs) obtained by a base change with the Fourier transform (FT). We use the framework by Püschel and Wendler [2020], which contains new FTs beyond the classical Walsh-Hadamard transform (WHT) [Bernasconi *et al.*, 1996], providing more flexibility. Until recently, methods for learning Fourier-sparse set functions focused on the WHT, and they placed assumptions on bidders' value functions that are too restrictive for CAs [Stobbe and Krause, 2012]. However, recently, Amrollahi *et al.* [2019] proposed a new algorithm that can approximate general set functions by WHT-sparse ones, which is suitable for large CAs and we use it in this work.

1.3 Our Contribution

Our main contribution in this paper is to bring the power of Fourier analysis to CA design (Section 3). In particular, we formulate *FT-based winner determination problems (WDPs)* and derive corresponding mixed integer programs (MIPs) for several FTs (Section 4). Our MIPs allow for the efficient solution of the FT-based WDP and provide the foundation for using Fourier-sparse approximations in auction design.

We first experimentally show that the WHT performs best among the FTs in terms of induced level of sparsity (Section 5.1) and reconstruction error (Section 5.2). As an initial approach, we develop a WHT-based allocation rule (Section 5.3). However, this requires too many queries for direct use in CAs. To overcome this, we propose a practical hybrid ICA based on NNs *and* Fourier analysis (Section 6.1). The key idea is to compute Fourier-sparse approximations of NN-based bidder representations, enabling us to keep the number of queries small. The advantage of the NN-based representations is that they capture key aspects of the bidders' value functions and can be queried arbitrarily often (Section 6.2).

Our efficiency experiments show that our hybrid ICA achieves higher efficiency than state-of-the-art mechanisms, leads to a significant computational speedup, and yields fairer allocations (Section 6.3). This shows that leveraging Fourier analysis in CA design is a promising new research direction.

2 Preliminaries

In this section, we present our formal model and review the MLCA mechanism, which our hybrid ICA builds upon.

2.1 Formal Model for ICAs

We consider a CA with n bidders and m indivisible items. Let $N = \{1, \dots, n\}$ and $M = \{1, \dots, m\}$ denote the set of bidders and items, respectively. We denote with $x \in \mathcal{X} = \{0,1\}^m$ a bundle of items represented as an indicator vector, where $x_j = 1$ iff item $j \in M$ is contained in x. Bidders' true preferences over bundles are represented by their (private) value functions $v_i : \mathcal{X} \to \mathbb{R}_+,\ i \in N$, i.e., $v_i(x)$ represents bidder i's true value for bundle x. By $a = (a_1, \dots, a_n) \in \mathcal{X}^n$ we denote an allocation of bundles to bidders, where a_i is the bundle bidder i obtains. We denote the set of *feasible* allocations by $\mathcal{F} = \left\{a \in \mathcal{X}^n : \sum_{i \in N} a_{ij} \leq 1,\ \forall j \in M\right\}$. The (true) *social welfare* of an allocation a is defined as $V(a) = \sum_{i \in N} v_i(a_i)$. We let $a^* \in \operatorname{argmax}_{a \in \mathcal{F}} V(a)$ be a social-welfare maximizing, i.e., *efficient*, allocation. The efficiency of any $a \in \mathcal{F}$ is measured by $V(a)/V(a^*)$. We assume that bidders' have quasilinear utilities u_i, i.e, for a payments $p \in \mathbb{R}^n_+$ it holds that $u_i(a, p) = v_i(a_i) - p_i$.

An ICA *mechanism* defines how the bidders interact with the auctioneer and how the final allocation and payments are determined. We denote a bidder's (possibly untruthful) reported value function by $\hat{v}_i : \mathcal{X} \to \mathbb{R}_+$. In this paper, we consider ICAs that ask bidders iteratively to report their value $\hat{v}_i(x)$ for particular bundles x selected by the mechanism (for early work on value queries see [Hudson and Sandholm, 2003]). A finite set of such reported bundle-value pairs of bidder i is denoted as $R_i = \left\{\left(x^{(l)}, \hat{v}_i(x^{(l)})\right)\right\},\ x^{(l)} \in \mathcal{X}$. Let $R = (R_1, \dots, R_n)$ denote the tuple of reported bundle-value pairs obtained from all bidders. We define the *reported social welfare* of an allocation a given R as $\widehat{V}(a|R) = \sum_{i \in N:\, (a_i, \hat{v}_i(a_i)) \in R_i} \hat{v}_i(a_i)$, where $(a_i, \hat{v}_i(a_i)) \in R_i$ ensures that only values for reported bundles contribute. Finally, the optimal allocation $a^*_R \in \mathcal{F}$ given the reports R is defined as

$$a^*_R \in \operatorname*{argmax}_{a \in \mathcal{F}} \widehat{V}(a|R). \tag{1}$$

The final allocation $a^*_R \in \mathcal{F}$ and payments $p(R) \in \mathbb{R}^n_+$ are computed based on the elicited reports R *only*.

As the auctioneer can only ask each bidder i a limited number of queries $|R_i| \leq Q^{\text{max}}$, the ICA needs a smart preference elicitation algorithm, with the goal of finding a highly efficient a^*_R with a limited number of value queries.

2.2 A Machine Learning-powered ICA

We now review the *machine learning-powered combinatorial auction (MLCA)* by Brero *et al.* [2021]. Interested readers are referred to Appendix A.1, where we present MLCA in detail.

MLCA starts by asking each bidder value queries for Q^{init} randomly sampled initial bundles. Next, MLCA proceeds in rounds until a maximum number of value queries per bidder Q^{max} is reached. In each round, for each bidder $i \in N$, it trains an ML algorithm $\mathcal{A}_i$ on the bidder's reports R_i. Next, MLCA generates new value queries $q^{\text{new}} = (q_i^{\text{new}})_{i=1}^n$ with $q_i^{\text{new}} \in \mathcal{X} \setminus R_i$ by solving a ML-based WDP $q^{\text{new}} \in \operatorname*{argmax}_{a \in \mathcal{F}} \sum_{i \in N} \mathcal{A}_i(a_i)$. The idea is the following: if $\mathcal{A}_i$ are good surrogate models of the bidders' true value functions then q^{new} should be a good proxy of the efficient allocation a^* and thus provide valuable information.

At the end of each round, MLCA receives reports R^{new} from all bidders for the newly generated q^{new} and updates R. When Q^{max} is reached, MLCA computes an allocation a^*_R maximizing the *reported* social welfare (eq. (1)) and determines VCG payments $p(R)$ (see Appendix A.2).

2.3 Incentives of MLCA and Hybrid ICA

A key concern in the design of ICAs are bidders' incentives. However, the seminal result by Nisan and Segal [2006] discussed above implies that practical ICAs cannot simply use VCG to achieve strategyproofness. And in fact, no ICA deployed in practice is *strategyproof* – including the famous SMRA and CCA auctions used to conduct spectrum auctions. Instead, auction designers have designed mechanisms that, while being manipulable, have "good incentives in practice" (see [Cramton, 2013; Milgrom, 2007]).

Naturally, the MLCA mechanism is also not strategyproof, and Brero *et al.* [2021] provide a simple example of a possible manipulation. The idea behind the example is straightforward: if the ML algorithm does not learn a bidder's preferences perfectly, a sub-optimal allocation may result. Thus, a bidder may (in theory) benefit from misreporting their preferences with the goal of "correcting" the ML algorithm, so that, with the misreported preferences, the mechanism actually finds a preferable allocation.

However, MLCA has two features that mitigate manipulations. First, MLCA explicitly queries each bidder's marginal economy, which implies that the marginal economy term of the final VCG payment is practically independent of bidder i's bid (for experimental support see [Brero *et al.*, 2021]). Second, MLCA enables bidders to "push" information to the auction which they deem useful. This mitigates certain manipulations of the main economy term in the VCG payment rule, as it allows bidders to increase the social welfare directly by pushing (useful) truthful information, rather than attempting to manipulate the ML algorithm. Brero *et al.* [2021] argued that with these two design features, MLCA exhibits very good incentives in practice. They performed a computational experiment, testing whether an individual bidder (equipped with more information than he would have in a real auction) can benefit from deviating from truthful bidding, while all other bidders are truthful. In their experiments, they could not identify a beneficial manipulation strategy. While this does not rule out that some (potentially more sophisticated) beneficial manipulations do exist, it provides evidence to support the claim that MLCA has good incentives in practice.

With two additional assumptions, one also obtains a theoretical incentive guarantee for MLCA. Assumption 1 requires that, if all bidders bid truthfully, then MLCA finds an efficient allocation (we show in Appendix D.3 that in two of our domains, we indeed find the efficient allocation in the majority of cases). Assumption 2 requires that, for all bidders i, if all other bidders report truthfully, then the social welfare of bidder i's marginal economy is independent of his value reports. If both assumptions hold, then bidding truthfully is an ex-post Nash equilibrium in MLCA.

Our hybrid ICA (Algorithm 1 in Section 6.1) is built upon MLCA, leaving the general framework in place, and only changing the algorithm that generates new queries each round. Given this design, the incentive properties of MLCA extend to the hybrid ICA. Specifically, our hybrid ICA is also not strategyproof, but it also has the same design features (including push-bids) to mitigate manipulations.

In future work, it would be interesting to evaluate experimentally whether the improved performance of the hybrid ICA translates into better manipulation mitigation compared to MLCA. However, such an analysis is beyond the scope of the present paper, which focuses on the ML algorithm that is integrated into the auction mechanism.

3 Fourier Analysis of Value Functions

We now show how to apply Fourier analysis to value functions providing the theoretical foundation of FT-based WDPs.

Classic Fourier analysis decomposes an audio signal or image into an orthogonal set of sinusoids of different frequencies. Similarly, the classical Fourier analysis for *set functions* (i.e., functions mapping subsets of a discrete set to a scalar) decomposes a set function into an orthogonal set of Walsh functions [Bernasconi *et al.*, 1996], which are piecewise constant with values 1 and -1 only. Recent work by Püschel and Wendler [2020] extends the Fourier analysis for set functions with several novel forms of set Fourier transforms (FTs). Importantly, because bidders' value functions are set functions, they are amenable to this type of Fourier analysis, and it is this connection that we will leverage in our auction design.

Sparsity. The motivation behind our approach is that we expect bidders' value functions to be *sparse*, i.e., they can be described with much less data than is contained in the exponentially-sized full value function. While this sparsity may be difficult to uncover when looking at bidders' value reports directly, it may reveal itself in the Fourier domain (where then most FCs are zero). As all FTs are changes of basis, each FT provides us with a new *lens on the bidder's value function*, revealing structure and thus potentially reducing dimensionality.

Set function Fourier transform. We now provide a formal description of FTs for reported value functions $\hat{v}_i$. To do so, we represent $\hat{v}_i$ as a vector $(\hat{v}_i(x))_{x\in\mathcal{X}}$. Each known FT is a change of basis and thus can be represented by a certain matrix $F \in \{-1, 0, 1\}^{2^m \times 2^m}$ with the form:

$$\phi_{\hat{v}_i}(y) = (F\hat{v}_i)(y) = \sum_{x\in\mathcal{X}} F_{y,x}\hat{v}_i(x). \tag{2}$$

There is exactly one Fourier coefficient per bundle, this follows from the theory presented by Püschel and Wendler [2020]. The corresponding inverse transform F^{-1} is thus:

$$\hat{v}_i(x) = (F^{-1}\phi_{\hat{v}_i})(x) = \sum_{y\in\mathcal{X}} F^{-1}_{x,y}\phi_{\hat{v}_i}(y). \tag{3}$$

$\phi_{\hat{v}_i}$ is again a set function and we call $\phi_{\hat{v}_i}(y)$ the *Fourier coefficient* at frequency y. A value function is *Fourier-sparse* if $|\operatorname{supp}(\phi_{\hat{v}_i})| = |\{y : \phi_{\hat{v}_i}(y) \neq 0\}| \ll 2^m$. We call $\operatorname{supp}(\phi_{\hat{v}_i})$ the *Fourier support* of $\hat{v}_i$.

Classically, the WHT is used as F [Bernasconi *et al.*, 1996; O'Donnell, 2014], but we also consider two recently introduced FTs (FT3, FT4) due to their information-theoretic interpretation given in [Püschel and Wendler, 2020]:

$$\textbf{FT3:} \quad F_{y,x} = (-1)^{|y|-|x|}\mathbb{I}_{\min(x,y)=x}, \tag{4}$$

$$\textbf{FT4:} \quad F_{y,x} = (-1)^{|\min(x,y)|}\mathbb{I}_{\max(x,y)=1_m}, \tag{5}$$

$$\textbf{WHT:} \quad F_{y,x} = \frac{1}{2^m}(-1)^{|\min(x,y)|}. \tag{6}$$

Here, $\min$ is the elementwise minimum (intersection of sets), $\max$ analogously, $|\cdot|$ is the set size, 1_m denotes the m-dimensional vector of 1s, and the indicator function $\mathbb{I}_P$ is equal to 1 if the predicate P is true and 0 otherwise.

Notions of Fourier-sparsity. In recent years, the notion of Fourier-sparsity has gained considerable attention, leading to highly efficient algorithms to compute FTs [Stobbe and Krause, 2012; Amrollahi *et al.*, 2019; Wendler *et al.*, 2021]. Many classes of set functions are Fourier-sparse (e.g., graph cuts, hypergraph valuations and decision trees [Abraham *et al.*, 2012]) and can thus be learned efficiently. The benefit of considering multiple FTs is that they offer different, non-equivalent notions of sparsity as illustrated by the following example.

Example 1. *Consider the set of items $M = \{1, 2, 3\}$ and the associated reported value function $\hat{v}_i$ shown in Table 1 (where we use 001 as a shorthand notation for $(0, 0, 1)$), together with the corresponding FCs $\phi_{\hat{v}_i}$: This bidder exhibits complementary effects for each bundle containing more than one item, as can be seen, e.g., from $3 = \hat{v}_i(110) > \hat{v}_i(100) + \hat{v}_i(010) = 2$ and $5 = \hat{v}_i(111) > \hat{v}_i(100) + \hat{v}_i(010) + \hat{v}_i(001) = 3$. Observe that while this value function is sparse in FT4, i.e., $\phi_{\hat{v}_i}(110) = \phi_{\hat{v}_i}(101) = \phi_{\hat{v}_i}(011) = 0$, it is neither sparse in FT3 nor WHT. Note that the coefficients $\phi_{\hat{v}_i}(100)$, $\phi_{\hat{v}_i}(010)$, and $\phi_{\hat{v}_i}(001)$ capture the value of single items and thus cannot be zero.*

	000	100	010	001	110	101	011	111
$\hat{v}_i$	0	1	1	1	3	3	3	5
FT3	0	1	1	1	1	1	1	-1
FT4	5	-2	-2	-2	0	0	0	1
WHT	$^{17}/_8$	$-^7/_8$	$-^7/_8$	$-^7/_8$	$^1/_8$	$^1/_8$	$^1/_8$	$^1/_8$

Table 1: Example with different induced notions of sparsity of all considered FTs.

The induced spectral energy *distributions for each FT, i.e., for each cardinality (i.e., number of items) d from 0 to $m = 3$, we compute $\sum_{y \in \mathcal{X}:|y|=d} \phi_{\hat{v}_i}(y)^2 / \sum_{y \in \mathcal{X}} \phi_{\hat{v}_i}(y)^2$, are shown in Table 2.*

	$d = 0$	$d = 1$	$d = 2$	$d = 3$
FT3	0.00	42.86	42.86	14.28
FT4	65.79	31.58	0.00	2.63
WHT	65.69	33.41	0.68	0.22

Table 2: Spectral energy in % for each cardinality (i.e., number of items) d from 0 to $m = 3$ of all considered FTs.

Fourier-sparse approximations. In practice, $\hat{v}_i$ may only be approximately sparse. Meaning that while not being sparse, it can be approximated well by a Fourier-sparse function $\tilde{v}_i$. Formally, let $\mathcal{S}_i = \text{supp}(\phi_{\tilde{v}_i})$ with $|\mathcal{S}_i| = k$, we call

$$\tilde{v}_i(x) = \sum_{y \in \mathcal{S}_i} F^{-1}_{x,y} \phi_{\tilde{v}_i}(y) \text{ for all } x \in \mathcal{X} \tag{7}$$

such that $\|\tilde{v}_i - \hat{v}_i\|_2$ is small a *k-Fourier-sparse approximation* of $\hat{v}_i$. We denote the vector of FCs by $\phi_{\tilde{v}_i|\mathcal{S}_i} = (\phi_{\tilde{v}_i}(y))_{y \in \mathcal{S}_i}$.

4 Fourier Transform-based WDPs

To leverage Fourier analysis for CA design, we represent bidders' value functions using Fourier-sparse approximations. A key step in most auction designs is to find the social welfare-maximizing allocation given bidder's reports, which is known as the *Winner Determination Problem* (WDP). To apply FTs, we need to be able to solve the WDP efficiently. Accordingly, we next derive MIPs for each of the FTs.

For each bidder $i \in N$, let $\tilde{v}_i : \mathcal{X} \to \mathbb{R}_+$ be a Fourier-sparse approximation of the bidders' reported value function $\hat{v}_i$. Next, we define the *Fourier transform-based WDP*.

Definition 1. *(*FOURIER TRANSFORM-BASED WDP*)*

$$\operatorname*{argmax}_{a \in \mathcal{F}} \sum_{i \in N} \tilde{v}_i(a_i). \tag{FT-WDP}$$

For $x, y \in \mathbb{R}^d$, let $x \leq y$, $\max(x, y)$ and $(-1)^x$ be defined component-wise, and let $\langle \cdot, \cdot \rangle$ denote the Euclidean scalar product. First, we formulate succinct representations of $\tilde{v}_i$.

Lemma 1. *For $i \in N$ let $\mathcal{S}_i = \{y^{(1)}, \ldots, y^{(k)}\}$ be the support of a k-Fourier-sparse approximation $\tilde{v}_i$ and $W_i \in \{0, 1\}^{k \times m}$ be defined as $(W_i)_{l,j} = \mathbb{I}_{y_j^{(l)}=1}$. Then it holds that*

$$\textbf{\textit{FT3:}} \quad \tilde{v}_i(x) = \left\langle \phi_{\tilde{v}_i|\mathcal{S}_i}, \max\left(0_k, 1_k - W_i(1_m - x)\right) \right\rangle \tag{8}$$

$$\textbf{\textit{FT4:}} \quad \tilde{v}_i(x) = \left\langle \phi_{\tilde{v}_i|\mathcal{S}_i}, \max\left(0_k, 1_k - W_i x\right) \right\rangle \tag{9}$$

$$\textbf{\textit{WHT:}} \quad \tilde{v}_i(x) = \left\langle \phi_{\tilde{v}_i|\mathcal{S}_i}, (-1)^{W_i x} \right\rangle. \tag{10}$$

See Appendix B.1 for the proof. With Lemma 1 and rewriting $\max(\cdot, \cdot)$ and $(-1)^{\cdot}$ as linear constraints, we next encode (FT-WDP) as a MIP (see Appendix B.2 for the proof).

Theorem 1. (FT-BASED MIPS) *Let $\tilde{v}_i : \mathcal{X} \to \mathbb{R}$ be a k-Fourier-sparse approximation from (8), (9), or (10). Then there exists a $C > 0$ s.t. the MIP defined by the objective*

$$\operatorname*{argmax}_{a \in \mathcal{F}, \beta_i \in \{0,1\}^k} \sum_{i \in N} \langle \phi_{\tilde{v}_i|\mathcal{S}_i}, \alpha_i \rangle, \tag{11}$$

and for $i \in N$ one set of transform specific constraints (12)–(14), or (15)–(17), or (18)–(20), is equivalent to (FT-WDP).

$$\textbf{\textit{FT3:}} \quad \text{s.t. } \alpha_i \geq 1_k - W_i(1_m - a_i) \tag{12}$$
$$\alpha_i \leq 1_k - W_i(1_m - a_i) + C\beta_i \tag{13}$$
$$0_k \leq \alpha_i \leq C(1_k - \beta_i) \tag{14}$$
$$\textbf{\textit{FT4:}} \quad \text{s.t. } \alpha_i \geq 1_k - W_i a_i \tag{15}$$
$$\alpha_i \leq 1_k - W_i a_i + C\beta_i \tag{16}$$
$$0_k \leq \alpha_i \leq C(1_k - \beta_i) \tag{17}$$
$$\textbf{\textit{WHT:}} \quad \text{s.t. } \alpha_i = -2\beta_i + 1_k \tag{18}$$
$$\beta_i = W_i a_i - 2\gamma_i \tag{19}$$
$$\gamma_i \in \mathbb{Z}^k \tag{20}$$

5 Analyzing the Potential of a FT-based CA

In this section, we experimentally evaluate the FTs and propose an FT-based allocation rule that motivates our practical hybrid ICA mechanism presented later in Section 6.

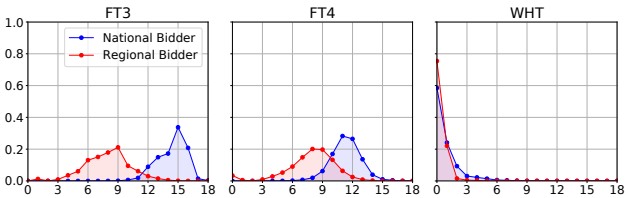


Figure 1: Spectral energy distribution in LSVM for all FTs. For each cardinality (x-axis), we plot the spectral energy of all frequencies of that cardinality normalized by the total spectral energy (y-axis).

For our experiments, we use the spectrum auction test suite (SATS) [Weiss *et al.*, 2017].[2] SATS enables us to generate synthetic CA instances in different domains. We have access to each bidder's *true* full value function v_i and the efficient allocation a^*. When simulating bidders, we assume truthful bidding (i.e., $\hat{v}_i = v_i$). We consider three domains:

The Global Synergy Value Model (GSVM) [Goeree and Holt, 2010] has 18 items, 6 *regional* and 1 *national bidder*.

The Local Synergy Value Model (LSVM) [Scheffel *et al.*, 2012] consists of 18 items, 5 *regional* and 1 *national bidder*. Complementarities arise from spatial proximity of items.

The Multi-Region Value Model (MRVM) [Weiss *et al.*, 2017] has 98 items and 10 bidders (categorized as *local*, *regional*, or *national*) and models large US spectrum auctions.

5.1 Notions of Fourier Sparsity

We first experimentally show that different notions of FT lead to different types of sparsity in LSVM (for other domains see Appendix C.1). For this we first compute the FTs of all bidders and then calculate their *spectral energy* distribution. That is, for each cardinality d (#items) from 0 to m, we compute $\sum_{y\in\mathcal{X}:|y|=d} \phi_{\hat{v}_i}(y)^2 / \sum_{y\in\mathcal{X}} \phi_{\hat{v}_i}(y)^2$. In Figure 1, we present the mean over 30 LSVM instances and bidder types.

Figure 1 shows that while the energy is spread among FCs of various degrees in FT3 and FT4, in WHT the low degree (≤ 2) FCs contain most of the energy, i.e., the WHT has much fewer dominant FCs that accurately describe each value function. As the WHT is orthogonal, learning low degree WHT-sparse approximations leads to low reconstruction error. Low degree WHT-sparse approximations can be learnt efficiently and accurately from a small number of queries using compressive sensing [Stobbe and Krause, 2012].

Note that the FT3 is identical to the classical polynomial value function representation [Lahaie, 2010] defined as

$$\hat{v}_i^{\text{poly}}(x) = \sum_{l=1}^{m} \sum_{\boldsymbol{j}=\{j_1,\dots,j_l\}\subseteq M} x_{j_1} \cdot \ldots \cdot x_{j_l} \cdot c_{\boldsymbol{j}}^{(i)}. \quad (21)$$

where the coefficient $c_{\boldsymbol{j}}^{(i)}$ is equal to the FT3 FC at frequency y with $y_i = 1$ for $i \in \{j_1, \dots, j_l\}$ and $y_i = 0$ else.[3] E.g. for $M = \{1, 2\}$, $\hat{v}_i^{\text{poly}}(x) = x_1 c_{\{1\}}^{(i)} + x_2 c_{\{2\}}^{(i)} + x_1 x_2 c_{\{1,2\}}^{(i)}$. Thus, converting $\hat{v}_i^{\text{poly}}$ into another FT basis (here WHT) can indeed be very helpful for the design of ML-based CAs.

[2] We used SATS version 0.6.4 for our experiments. The implementations of GSVM and LSVM have changed slightly in newer SATS versions. This must be considered when comparing the performance of different mechanisms in those domains.

[3] This can be seen by calculating the inverse in (4), i.e., $F_{y,x}^{-1} = \mathbb{I}_{\min(x,y)=x}$, and plugin $F_{y,x}^{-1}$ into (3).

Domain	k	Bidder	FT3	FT4	WHT	NN
GSVM	100	Nat.	11.3 ± 0.7	14.2 ± 0.8	1.8 ± 0.1	9.0 ± 1.8
		Reg.	0.0	1.4 ± 0.2	0.4 ± 0.1	7.2 ± 0.9
	200	Nat.	0.0	0.0	0.0	5.7 ± 0.4
		Reg.	0.0	0.0	0.0	5.2 ± 0.8
LSVM	100	Nat.	78.4 ± 1.0	580.2 ± 7.9	31.2 ± 0.4	48.7 ± 1.2
		Reg.	28.2 ± 2.3	48.5 ± 2.7	6.8 ± 0.3	17.8 ± 0.7
	200	Nat.	95.8 ± 1.2	639.0 ± 10.0	26.2 ± 0.3	40.6 ± 0.7
		Reg.	25.8 ± 2.0	43.1 ± 2.4	5.3 ± 0.3	15.3 ± 0.9

Table 3: Reconstruction error with a 95%-CI of k-Fourier-sparse approximations $\tilde{v}_i$ and NNs trained on k randomly selected bundles. Winners are marked in grey.

5.2 Reconstruction Error of Fourier Transforms

Next we validate the FT approach by comparing the reconstruction error of the FTs in the medium-sized GSVM and LSVM, where we can still compute the full FT (in contrast to MRVM). For now, we assume that we have access to bidders' full $\hat{v}_i$. In Procedure 1, we determine the best k-Fourier-sparse approximation $\tilde{v}_i$ (see Appendix C.2 for details).

Procedure 1. (Best FCs Given Full Access to $\hat{v}_i$)
Compute $\tilde{v}_i$ using the k absolutely largest FCs $\phi_{\hat{v}_i|\mathcal{S}_i}$ from the full FT for each bidders' reported value function $\phi_{\hat{v}_i} = F\hat{v}_i$.

Remark 1. *Since the WHT is orthogonal and the simulated auction data is noise-free, its approximation error is exactly equal to the residual of the FCs. Thus, Procedure 1 is optimal for the WHT. This is not the case for FT3 and FT4 because they are not orthogonal.*

We then calculate the RMSE $(\frac{1}{2^m} \sum_{x\in\mathcal{X}} (\hat{v}_i(x) - \tilde{v}_i(x))^2)^{1/2}$ averaged over 100 instances and bidder types. In Table 3, we present the RMSEs for the three FTs and for NNs, where we used the architectures from Weissteiner and Seuken [2020].

For GSVM, we observe that we can perfectly reconstruct $\hat{v}_i$ with the 200 best FCs, which shows that GSVM is 200-sparse. In contrast, LSVM is non-sparse, and we do not achieve perfect reconstruction with 200 FCs. Overall, we observe that the WHT outperforms FT3 and FT4. Moreover, we see that, if we could compute the k best FCs of the WHT from k training points, the WHT would outperform the NNs.

However, in practice, we do not have access to full value functions. Instead, we must use an algorithm that computes the best FCs using a reasonable number of value queries.

Remark 2. *Thanks to its orthogonality the WHT has strong theoretical guarantees for sparse recovery from few samples using compressive sensing (see [Stobbe and Krause, 2012]). Thus, we focus on the WHT in the remainder of this paper.*

5.3 A Fourier Transform-based Allocation Rule

We now present an FT-based allocation rule using the *robust sparse WHT algorithm (RWHT)* by Amrollahi *et al.* [2019]. RWHT learns a Fourier-sparse approximation $\tilde{v}_i$ of $\hat{v}_i$ from *value queries*. Procedure 2 finds the allocation $\tilde{a}$.

Procedure 2. (WHT-BASED ALLOCATION RULE)
i. *Use RWHT to compute k-sparse approximations $\tilde{v}_i$, $i \in N$.*
ii. *Solve $\tilde{a} \in \operatorname{argmax}_{a \in \mathcal{F}} \sum_{i \in N} \tilde{v}_i(a_i)$ using Theorem 1.*

In Figure 2, we present the efficiency of $\tilde{a}$ on 50 GSVM instances for various values of k. We see that RWHT achieves a median efficiency of 100% for 90 or more FCs. Nevertheless, the main practical issue with this approach is the number of value queries required. As we can see, RWHT needs $102,000$ value queries (39% of all bundles) to find the 90 best FCs. For a practical ICA mechanism this is far too many.

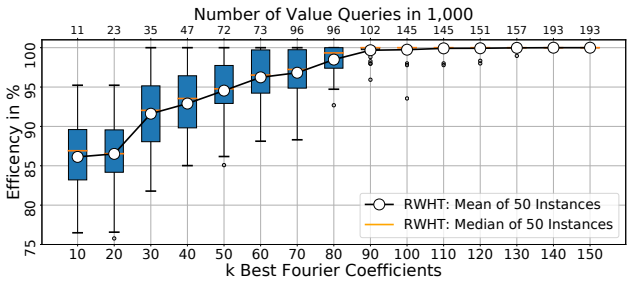


Figure 2: Efficiency of Procedure 2 in GSVM.

6 A Practical Hybrid ICA Mechanism

In this section, we introduce and experimentally evaluate a practical hybrid ICA mechanism, based on FTs *and* NNs.

6.1 The Hybrid ICA Mechanism

The main issue of the FT-based allocation rule in Section 5.3 is the large number of queries, which we now address. The idea is the following: instead of directly applying a sparse FT algorithm (like RWHT) to bidders, we apply it to a NN-based representation. In this way, we query NNs instead of bidders. Based on the FCs of the NNs, we determine a Fourier-sparse approximation $\tilde{v}_i$ with only *few value queries*, where the idea is that the FCs of each NN concentrate on the most dominant FCs of its respective value function. Indeed, recent evidence suggests that a NN trained by SGD can learn the Fourier-support [Rahaman *et al.*, 2019]. We analyze our NN support discovery rule in Section 6.2. We now present HYBRID ICA, leaving details of the sub-procedures to Appendix D.1.

HYBRID ICA (Algorithm 1) consists of 3 phases: the *MLCA*, the *Fourier reconstruction*, and the *Fourier allocation phase.* It is parameterized by an FT F and the numbers $\ell_1, \ell_2, \ell_3, \ell_4$ of different query types. In total, it asks each bidder $\sum_{i=1}^{4} \ell_i$ queries: ℓ_1 random initial, ℓ_2 MLCA, ℓ_3 Fourier reconstruction, and ℓ_4 Fourier allocation queries.

1. MLCA Phase. We first run MLCA such that the NNs can then be trained on "meaningfully" elicited reports. In MLCA, we request reports for ℓ_1 random initial bundles and for ℓ_2 *MLCA queries* (Lines 1-2).

2. Fourier Reconstruction Phase. Next, we compute a Fourier-sparse approximation $\tilde{v}_i$. For this, we first fit a NN $\mathcal{N}_i$ on the reports R_i (Line 4). Then we compute the best FCs of the fitted NNs (Line 5, **Procedure 3**) in order to discover

Algorithm 1: HYBRID ICA

Params: F Fourier transform; $\ell_1, \ell_2, \ell_3, \ell_4$ query split
1 Set $Q^{\text{init}} = \ell_1$ and $Q^{\max} = \ell_1 + \ell_2$ ▷ MLCA phase
2 Run MLCA($Q^{\text{init}}, Q^{\max}$); get $\ell_1 + \ell_2$ reports R
3 **foreach** *bidder* $i \in N$ **do** ▷ Fourier reconstr. phase
4 Fit NN $\mathcal{N}_i$ to R_i
5 Determine the best FCs of $\mathcal{N}_i$ ▷ **Proc. 3**
6 Compute ℓ_3 reconstruction queries $\tilde{\mathcal{S}}_i \subseteq \mathcal{X}$ ▷ **Proc. 4**
7 Ask $\tilde{\mathcal{S}}_i$, add reports to R_i, and fit $\tilde{v}_i$ to R_i ▷ **Proc. 5**
8 **for** $l = 1, \ldots, \ell_4$ **do** ▷ Fourier alloc. phase
9 Solve $q \in \operatorname{argmax}_{a \in \mathcal{F}} \sum_{i \in N} \tilde{v}_i(a_i)$ (FT-WDP)
10 **foreach** *bidder* $i \in N$ **do**
11 **if** $q_i \in R_i$ **then** ▷ Bundle already queried
12 Define $\mathcal{F}' = \{a \in \mathcal{F} : a_i \neq x, \forall x \in R_i\}$
13 Resolve $q' \in \operatorname{argmax}_{a \in \mathcal{F}'} \sum_{i \in N} \tilde{v}_i(a_i)$
14 Update $q_i = q'_i$
15 Query bidder i's value for q_i and add report to R_i
16 Fit $\tilde{v}_i$ to R_i ▷ **Proc. 5**
17 From R compute: a^*_R as in eq. (1), VCG payments $p(R)$
18 **return** *Final allocation a^*_R and VCG payments $p(R)$*

which FCs are important to represent the bidders. Based on these FCs, we determine ℓ_3 *Fourier reconstruction queries* $\tilde{\mathcal{S}}_i$ (Line 6, **Procedure 4**), send them to the bidders and fit $\tilde{v}_i$ to the reports R_i received so far (Line 7, **Procedure 5**).

3. Fourier Allocation Phase. We use the fitted $\tilde{v}_i$ to generate ℓ_4 *Fourier allocation queries*. Here, we solve the FT-based WDP (Line 9) to get candidate queries q, ensure that all queries are new (Lines 11–14), add the received reports to R_i (Line 15) and refit $\tilde{v}_i$ (Line 16). Finally in Line 17, HYBRID ICA computes based on all reports R a welfare-maximizing allocation a^*_R and VCG payments $p(R)$ (see Appendix A.2).

Experiment Setup. For HYBRID ICA and MLCA, we use the NN architectures from Weissteiner and Seuken [2020] and set a total query budget of 100 (GSVM, LSVM) and 500 (MRVM). For HYBRID ICA, we optimized the FTs and query parameters ℓ_i on a training set of CA instances. Table 4 shows the best configurations.

	NN ARCHITECTURES	FT	ℓ_1	ℓ_2	ℓ_3	ℓ_4
GSVM	R:[32, 32] | N:[10, 10]	WHT	30	21	20	29
LSVM	R:[32, 32] | N:[10, 10, 10]	WHT	30	30	10	30
MRVM	L,R,N:[16, 16]	WHT	30	220	0	250

Table 4: Best configuration of HYBRID ICA. R:$[d_1, d_2]$ denotes a 3-hidden-layer NN for the regional bidder with d_1, and d_2 nodes.

6.2 NNs Support Discovery Experiments

In HYBRID ICA we use the NNs for support discovery where it is key that the FCs of these NNs concentrate on the dominant FCs of its value function, i.e. $\text{supp}(\phi_{\mathcal{N}_i}) \approx \text{supp}(\phi_{\tilde{v}_i})$.

To evaluate the NN-based support discovery (Line 5), we consider the spectral *energy ratio* obtained by dividing the spectral energies of the k frequencies selected from the NN and the k best frequencies (for the WHT the best FCs are the ones with the largest absolute value). Formally, for each bidder i, let the k frequencies selected from the NN be $\tilde{\mathcal{S}}_i = \{\tilde{y}^{(1)}, \ldots, \tilde{y}^{(k)}\}$ and the best ones be $\mathcal{S}^*_i =$

	GSVM					LSVM					MRVM					
Mechanism	Efficiency in %	Regional in %	National in %	Rev in %	hrs/ Inst.	Efficiency in %	Regional in %	National in %	Rev in %	hrs/ Inst.	Efficiency in %	Local in %	Regional in %	National in %	Rev in %	hrs/ Inst.
Hybrid ICA	99.97 ± 0.03	94.72	5.25	81	0.81	98.74 ± 0.43	89.09	9.65	78	1.95	96.63 ± 0.31	0.00	1.19	95.44	36	23.88
MLCA	99.17 ± 0.37	98.11	1.06	79	4.65	99.14 ± 0.42	93.40	5.75	77	6.09	95.32 ± 0.32	0.00	0.53	94.79	41	43.26
Hybrid ICA (no FR)	98.30 ± 0.49	97.94	0.36	75	0.93	98.16 ± 0.60	93.83	4.33	72	2.03	96.63 ± 0.31	0.00	1.19	95.44	36	23.88
Hybrid ICA (no FR/FA)	98.16 ± 0.50	97.47	0.69	75	0.71	97.75 ± 0.63	92.78	5.27	72	1.86	93.91 ± 0.36	0.01	0.42	93.48	42	14.68
Efficient Allocation		94.75	5.25				84.03	15.97				0.00	2.11	97.89		

Table 5: Hybrid ICA vs. MLCA, Hybrid ICA (no FR), and Hybrid ICA (no FR/FA). All results are averages over a test set of 100 (GSVM and LSVM) and 30 (MRVM) CA instances. For efficiency we give a 95% confidence interval and mark the best mechanisms in grey.

$\{\overset{*}{y}^{(1)}, \ldots, \overset{*}{y}^{(k)}\}$. Then, bidder i's *energy ratio* is given by $\sum_{\tilde{y} \in \tilde{S}_i} \phi_{\hat{v}_i}(\tilde{y})^2 / \sum_{\overset{*}{y} \in \mathcal{S}_i^*} \phi_{\hat{v}_i}(\overset{*}{y})^2 \in [0, 1]$ (see Appendix D.2 for details). This ratio is equal to one if $\tilde{S}_i = \mathcal{S}_i^*$. Figure 3 shows that the NN-based supports are almost on par with the best supports given a fixed budget of k frequencies.

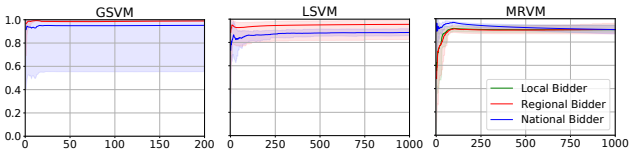


Figure 3: Average energy ratio (y-axis) with 97.5% and 2.5% empirical quantiles for a number of selected frequencies k (x-axis) over 30 instances in GSVM and LSVM and over 5 instances in MRVM.

6.3 Efficiency Experiments

We now evaluate the efficiency of Hybrid ICA vs MLCA.

Results. Table 5 contains our main results in all domains.[4] We show efficiency, distribution of efficiency to bidder types, revenue $(\sum_{i \in N} p(R)_i)/V(a^*)$, and runtime. First, we see that Hybrid ICA *statistically significantly outperforms* MLCA *w.r.t. efficiency* in GSVM and MRVM and performs on par in LSVM. Second, it also leads to a computational speedup ($\times 6$ GSVM, $\times 3$ LSVM, $\times 2$ MRVM). The reason for this computational speedup is that the generation of the $\ell_3 + \ell_4$ Fourier queries (estimating the superset of the support using RWHT on the NNs, fitting the FT models using compressive sensing and solving our new FT-based MIPs) is faster than the generation of the NN-based MLCA allocation queries (training NNs and solving the NN-based MIP). Third, it distributes the welfare more evenly (= fairer) to bidder types.[5] This also leads to a distribution that more closely resembles that of the efficient allocation (see *Efficient Allocation*). We present full efficiency path plots for the different phases of Hybrid ICA in Appendix D.3.

Fourier queries. To verify the importance of the ℓ_3 *Fourier reconstruction* and ℓ_4 *Fourier allocation queries*, we also present Hybrid ICA (no FR) and Hybrid ICA (no FR/FA), which use random queries in place of the ℓ_3 Fourier reconstruction and the $\ell_3 + \ell_4$ Fourier-based queries. As we see in Table 5, using the Fourier queries leads to significantly better efficiency and Hybrid ICA (no FR/FA) does not achieve a fairer efficiency distribution. A comparison of Hybrid ICA to Hybrid ICA (no FR) reveals that, in GSVM and LSVM, the Fourier reconstruction queries cause the fairer distribution. We empirically verified that these queries are composed of larger bundles (on avg. 17 items vs. 4 in MLCA queries) and thus allocate large bundles to bidders that would have been overlooked. In MRVM, the optimal query split for Hybrid ICA uses $\ell_3 = 0$ Fourier reconstruction queries such that Hybrid ICA is equal to Hybrid ICA (no FR). Thus, in MRVM, Hybrid ICA's increased efficiency and fairer distribution results from the Fourier allocation queries.

Overall, we see that our Fourier-based auction is especially powerful in sparse domains. In practice, bidders are often limited by their cognitive abilities [Scheffel *et al.*, 2012] or use a low-dimensional computational model to represent their value function. Thus, their reported preferences typically exhibit only a limited degree of substitutability and complementarity, which is captured well by Fourier-sparsity.

7 Conclusion

We have introduced Fourier analysis for the design of CAs. The main idea was to represent value functions using Fourier-sparse approximations, providing us with a new lens on bidder's values in the Fourier domain.

On a technical level, we have derived succinct MIPs for the Fourier transform-based WDPs, which makes computing Fourier-based allocation queries practically feasible. We have leveraged this to design a new hybrid ICA that uses NN and Fourier-queries. Our experiments have shown that our approach leads to higher efficiency, a computational speedup and a fairer distribution of social welfare than state-of-the-art.

With this paper, we have laid the foundations for future work leveraging Fourier analysis for representing and eliciting preferences in combinatorial settings.

Acknowledgments

We thank the anonymous reviewers for helpful comments. This paper is part of a project that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No. 805542). Furthermore, this material is based upon work supported by the National Science Foundation under grant no. CMMI-1761163.

[4] All experiments were conducted on machines with Intel Xeon E5 v4 2.20GHz processors with 24 cores and 128GB RAM or with Intel E5 v2 2.80GHz processors with 20 cores and 128GB RAM.

[5] We consider an allocation to be "fairer" if its social welfare is more evenly distributed among bidder types. This is similar (but not identical) to the standard notion of egalitarian social welfare.

References

[Abraham *et al.*, 2012] I. Abraham, M. Babaioff, S. Dughmi, and T. Roughgarden. Combinatorial auctions with restricted complements. In *Proceedings of the 13th ACM Conference on Electronic Commerce*, 2012. 4

[Amrollahi *et al.*, 2019] A. Amrollahi, A. Zandieh, M. Kapralov, and A. Krause. Efficiently learning Fourier sparse set functions. In *Advances in Neural Information Processing Systems*, 2019. 2, 4, 5, 12, 13, 14

[Ausubel and Baranov, 2017] L. M. Ausubel and O. Baranov. A practical guide to the combinatorial clock auction. *Economic Journal*, 127(605):F334–F350, 2017. 1

[Ausubel and Cramton, 2011] L. Ausubel and P. Cramton. Auction design for wind rights. *Report to Bureau of Ocean Energy Management, Regulation and Enforcement*, 2011. 1

[Ausubel *et al.*, 2006] L. Ausubel, P. Cramton, and P. Milgrom. The clock-proxy auction: A practical combinatorial auction design. In *Combinatorial Auctions*, pages 115–138. MIT Press, 2006. 1

[Bernasconi *et al.*, 1996] A. Bernasconi, B. Codenotti, and J. Simon. On the Fourier analysis of Boolean functions. *preprint*, pages 1–24, 1996. 2, 3

[Blum *et al.*, 2004] A. Blum, J. Jackson, T. Sandholm, and M. Zinkevich. Preference elicitation and query learning. *Journal of Machine Learning Research*, 5:649–667, 2004. 1

[Brero *et al.*, 2018] G. Brero, B. Lubin, and S. Seuken. Combinatorial auctions via machine learning-based preference elicitation. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 2018. 1

[Brero *et al.*, 2019] G. Brero, S. Lahaie, and S. Seuken. Fast iterative combinatorial auctions via bayesian learning. In *Proceedings of the 33rd AAAI Conference of Artificial Intelligence*, 2019. 1

[Brero *et al.*, 2021] G. Brero, B. Lubin, and S. Seuken. Machine learning-powered iterative combinatorial auctions. *arXiv preprint arXiv:1911.08042*, Jan 2021. 1, 2, 3, 10

[Cramton, 2013] P. Cramton. Spectrum auction design. *Review of Industrial Organization*, 42(2):161–190, 2013. 3

[Dütting *et al.*, 2019] P. Dütting, Z. Feng, H. Narasimhan, D. Parkes, and S. Ravindranath. Optimal auctions through deep learning. In *Proceedings of the 36th International Conference on Machine Learning*, 2019. 1

[Efron *et al.*, 2004] B. Efron, T. Hastie, I. Johnstone, R. Tibshirani, et al. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004. 14

[Goeree and Holt, 2010] J. K. Goeree and C. A. Holt. Hierarchical package bidding: A paper & pencil combinatorial auction. *Games and Economic Behavior*, 2010. 5

[Heiss *et al.*, 2022] Jakob M Heiss, Jakob Weissteiner, Hanna S Wutte, Sven Seuken, and Josef Teichmann. NOMU: Neural optimization-based model uncertainty. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 8708–8758. PMLR, 17–23 Jul 2022. 1

[Hudson and Sandholm, 2003] B. Hudson and T. Sandholm. Using value queries in combinatorial auctions. In *Proceedings of the 4th ACM Conference on Electronic commerce*, pages 226–227, 2003. 2

[Lahaie and Parkes, 2004] S. Lahaie and C. Parkes. Applying learning algorithms to preference elicitation. In *Proceedings of the 5th ACM Conference on Electronic Commerce*, 2004. 1

[Lahaie, 2010] S. Lahaie. Kernel methods for revealed preference analysis. In *ECAI*, pages 439–444, 2010. 5

[Milgrom, 2007] P. Milgrom. Package auctions and exchanges. *Econometrica*, 75(4):935–965, 2007. 3

[Nisan and Segal, 2006] N. Nisan and I. Segal. The communication requirements of efficient allocations and supporting prices. *Journal of Economic Theory*, 2006. 1, 3

[O'Donnell, 2014] R. O'Donnell. *Analysis of boolean functions*. Cambridge University Press, 2014. 2, 3

[Püschel and Wendler, 2020] M. Püschel and C. Wendler. Discrete signal processing with set functions. *IEEE Transactions on Signal Processing*, 69:1039–1053, 2020. 2, 3, 10, 13, 14

[Rahaman *et al.*, 2019] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. Hamprecht, Y. Bengio, and A. Courville. On the spectral bias of neural networks. In *International Conference on Machine Learning*, 2019. 6

[Rahme *et al.*, 2021] J. Rahme, S. Jelassi, J. Bruna, and M. Weinberg. A permutation-equivariant neural network architecture for auction design. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*, 2021. 1

[Scheffel *et al.*, 2012] T. Scheffel, G. Ziegler, and M. Bichler. On the impact of package selection in combinatorial auctions: an experimental study in the context of spectrum auction design. *Experimental Economics*, 2012. 5, 7

[Shen *et al.*, 2019] W. Shen, P. Tang, and S. Zuo. Automated mechanism design via neural networks. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, 2019. 1

[Shen *et al.*, 2020] W. Shen, B. Peng, H. Liu, M. Zhang, R. Qian, Y. Hong, Z. Guo, Z. Ding, P. Lu, and P. Tang. Reinforcement mechanism design: With applications to dynamic pricing in sponsored search auctions. In *Proceedings of the 34th AAAI Conference*, 2020. 1

[Stobbe and Krause, 2012] P. Stobbe and A. Krause. Learning Fourier sparse set functions. In *Artificial Intelligence and Statistics*, 2012. 2, 4, 5, 13, 14

[Weiss *et al.*, 2017] M. Weiss, B. Lubin, and S. Seuken. Sats: A universal spectrum auction test suite. In *Proceedings of the 16th Conference on Autonomous Agents and Multi-Agent Systems*, 2017. 5

[Weissteiner and Seuken, 2020] Jakob Weissteiner and Sven Seuken. Deep learning—powered iterative combinatorial auctions. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(02):2284–2293, Apr. 2020. 1, 5, 6, 10

[Weissteiner *et al.*, 2022a] Jakob Weissteiner, Jakob Heiss, Julien Siems, and Sven Seuken. Monotone-value neural networks: Exploiting preference monotonicity in combinatorial assignment. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 541–548. International Joint Conferences on Artificial Intelligence Organization, 7 2022. Main Track. 1

[Weissteiner *et al.*, 2022b] Jakob Weissteiner, Chris Wendler, Sven Seuken, Ben Lubin, and Markus Püschel. Fourier analysis-based iterative combinatorial auctions. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 549–556. International Joint Conferences on Artificial Intelligence Organization, 7 2022. Main Track. 1

[Weissteiner *et al.*, 2023] Jakob Weissteiner, Jakob Heiss, Julien Siems, and Sven Seuken. Bayesian optimization-based combinatorial assignment. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37, 2023. 1

[Wendler *et al.*, 2021] C. Wendler, A. Amrollahi, B. Seifert, A. Krause, and M. Püschel. Learning set functions that are sparse in non-orthogonal Fourier bases. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*, 2021. 4, 12, 13

Appendix

A Preliminaries

A.1 A Machine Learning powered ICA

In this section, we provide a detailed review of the *machine learning-powered combinatorial auction (MLCA)* introduced by Brero *et al.* [2021]. We describe MLCA using NNs $\mathcal{N}_i$ as the generic ML algorithm $\mathcal{A}_i$ (see [Weissteiner and Seuken, 2020]).

At the core of MLCA is a *query module* (Algorithm 2), which, for each bidder $i \in I$, determines a new value query q_i. First, in the *estimation step* (Line 1), NNs are used to learn bidders' valuations from reports R_i. Next, in the *optimization step* (Line 2), a *NN-based WDP* is solved to find a candidate q of value queries (see [Weissteiner and Seuken, 2020] for details on the NN-based *estimation* and *optimization step*). Finally, if q_i has already been queried before (Line 4), another, more restricted NN-based WDP (Line 6) is solved and q_i is updated correspondingly. This ensures that all final queries q are new.

Algorithm 2: NN-QUERY MODULE (Brero et al. 2021)

Function : *NextQueries*(I, R)
Inputs : Index set of bidders I and reported values R
Parameters: Neural networks $\mathcal{N}_i : \mathcal{X} \to \mathbb{R}_+$, $i \in N$
1 **foreach** $i \in I$ **do** Fit $\mathcal{N}_i$ on R_i: $\mathcal{N}_i[R_i]$ ▷ `Estimation step`
2 Solve $q \in \operatorname{argmax}_{a \in \mathcal{F}} \sum_{i \in I} \mathcal{N}_i[R_i](a_i)$ ▷ `Optimization step`
3 **foreach** $i \in I$ **do**
4 | **if** $q_i \in R_i$ **then** ▷ `Bundle already queried`
5 | | Define $\mathcal{F}' = \{a \in \mathcal{F} : a_i \neq x, \forall x \in R_i\}$
6 | | Re-solve $q' \in \operatorname{argmax}_{a \in \mathcal{F}'} \sum_{l \in I} \mathcal{N}_l[R_l](a_l)$
7 | | Update $q_i = q'_i$
8 **return** *profile of new queries* $q = (q_1, \dots, q_n)$

In Algorithm 3, we present MLCA in a slightly abbreviated form. Let $R_{-i} = (R_1, \dots, R_{i-1}, R_{i+1}, \dots, R_n)$. MLCA proceeds in rounds until a maximum number of queries per bidder $Q^{\max}$ is reached. In each round, it calls Algorithm 2 $n+1$ times: once including all bidders (Line 4, *main economy*) and n times excluding one bidder (Lines 5–6, *marginal economies*). At the end of each round, the mechanism receives reports R^{new} from all bidders for the newly generated queries q^{new}, and updates the overall elicited reports R (Lines 7–8). In Lines 10–11, MLCA computes an allocation a^*_R that maximizes the reported social welfare and determines VCG payments $p(R)$ (see Appendix A.2).

A.2 VCG Payments from Reports

In this section, we provide a recap on how to compute VCG payments from bidder's reports. Let $R = (R_1, \dots, R_n)$ denote an elicited set of reported bundle-value pairs from each bidder obtained from MLCA (Algorithm 3) or HYBRID ICA (Algorithm 1) and let $R_{-i} := (R_1, \dots, R_{i-1}, R_{i+1}, \dots, R_n)$. We then calculate the VCG payments $p(R) = (p(R)_1 \dots, p(R)_n) \in \mathbb{R}^n_+$ as follows:

Algorithm 3: MLCA (Brero et al. 2021)

Params: $t = 1, Q^{\text{init}}, Q^{\max}$ init. and max #queries per bidder
1 **foreach** $i \in N$ **do**
2 | Receive reports R_i for Q^{init} randomly drawn bundles
3 **while** $t \leq \lfloor (Q^{max} - Q^{init})/n \rfloor$ **do** ▷ `Round iterator`
4 | $q^{\text{new}} = NextQueries(N, R)$ ▷ `Main economy queries`
5 | **foreach** *bidder* $i \in N$ **do** ▷ `Marginal economy queries`
6 | | $q^{\text{new}} = q^{\text{new}} \cup NextQueries(N \setminus \{i\}, R_{-i})$
7 | **foreach** *bidder* $i \in N$ **do**
8 | | Receive reports R_i^{new} for q_i^{new}, set $R_i = R_i \cup R_i^{\text{new}}$
9 | $t = t + 1$
10 From elicited reports R compute a^*_R as in Equation (1)
11 From elicited reports R compute VCG-payments $p(R)$
12 **return** *Final allocation* a^*_R *and payments* $p(R)$

Definition 2. (VCG PAYMENTS FROM REPORTS)

$$p(R)_i := \sum_{j \in N \setminus \{i\}} \hat{v}_j \left(\left(a^*_{R_{-i}} \right)_j \right) - \sum_{j \in N \setminus \{i\}} \hat{v}_j \left((a^*_R)_j \right). \tag{22}$$

where $a^*_{R_{-i}}$ maximizes the reported social welfare when excluding bidder i, i.e.,

$$a^*_{R_{-i}} \in \operatorname*{argmax}_{a \in \mathcal{F}} \widehat{V}(a|R_{-i}) = \operatorname*{argmax}_{a \in \mathcal{F}} \sum_{\substack{j \in N \setminus \{i\}: \\ (a_j, \hat{v}_j(a_j)) \in R_j}} \hat{v}_j(a_j), \tag{23}$$

and a^*_R is a reported-social-welfare-maximizing allocation (including all bidders), i.e,

$$a^*_R \in \operatorname*{argmax}_{a \in \mathcal{F}} \widehat{V}(a|R) = \operatorname*{argmax}_{a \in \mathcal{F}} \sum_{i \in N: (a_i, \hat{v}_i(a_i)) \in R_i} \hat{v}_i(a_i). \tag{24}$$

As argued by Brero *et al.* [2021], using such payments are key for MLCA to induce "good" incentives for bidders to report truthfully. As their incentive analysis also applies to HYBRID ICA, we use them in our design too.

B Fourier Transform-based WDPs

In this section, we present the proofs of Lemma 1 and Theorem 1.

Let 1_d and 0_d for $d \in \mathbb{N}$ denote the d-dimensional vector of ones and zeros, respectively. For all $x, y \in \mathbb{R}^d$, let $x \leq y$, $\max(x, y)$, $\min(x, y)$ and $(-1)^x$ be defined componentwise, and let $\langle \cdot, \cdot \rangle$ denote the Euclidean scalar product.

We consider the matrix representation F and F^{-1} of the considered FTs from Püschel and Wendler [2020] given by:

$$\textbf{FT3:} \quad F_{y,x} = (-1)^{|y|-|x|} \mathbb{I}_{\min(x,y)=x}, \tag{25}$$

$$F^{-1}_{x,y} = \mathbb{I}_{\min(x,y)=y}, \tag{26}$$

$$\textbf{FT4:} \quad F_{y,x} = (-1)^{|\min(x,y)|} \mathbb{I}_{\max(x,y)=1_m}, \tag{27}$$

$$F^{-1}_{x,y} = \mathbb{I}_{\min(x,y)=0_m}, \tag{28}$$

$$\textbf{WHT:} \quad F_{y,x} = \frac{1}{2^m} (-1)^{|\min(x,y)|}, \tag{29}$$

$$F^{-1}_{x,y} = (-1)^{|\min(x,y)|}. \tag{30}$$

The Fourier-sparse approximations used in the WDPs are defined in terms of F^{-1}.

B.1 Proof of Lemma 1

Lemma 1. (SUCCINCT REPRESENTATIONS) *For $i \in N$ let $\mathcal{S}_i = \{y^{(1)}, \dots, y^{(k)}\}$ be the support of a k-Fourier-sparse approximation $\tilde{v}_i$ and $W_i \in \{0,1\}^{k \times m}$ be defined as $(W_i)_{l,j} = \mathbb{I}_{y_j^{(l)}=1}$. Then $\tilde{v}_i$ can be rewritten as:*

$$\textbf{\textit{FT3:}}\ \tilde{v}_i(x) = \left\langle \phi_{\tilde{v}_i|\mathcal{S}_i}, \max\left(0_k, 1_k - W_i(1_m - x)\right)\right\rangle \tag{31}$$

$$\textbf{\textit{FT4:}}\ \tilde{v}_i(x) = \left\langle \phi_{\tilde{v}_i|\mathcal{S}_i}, \max\left(0_k, 1_k - W_i x\right)\right\rangle \tag{32}$$

$$\textbf{\textit{WHT:}}\ \tilde{v}_i(x) = \left\langle \phi_{\tilde{v}_i|\mathcal{S}_i}, (-1)^{W_i x}\right\rangle. \tag{33}$$

Proof. In this proof we are going to make use of the equivalence between bundles (= indicator vectors) and sets. Recall that $x, y \in \mathcal{X} = \{0,1\}^m$ are bundles and bundles correspond to subsets of items. We can translate set operations such as intersection, union and complement to indicator vectors as follows:

$$x \cap y = \min(x, y), \tag{34}$$

$$x \cup y = \max(x, y), \tag{35}$$

$$(x)^c = 1_m - x, \tag{36}$$

where we slightly abused notation by identifying the indicator vectors with their corresponding subsets on the left hand sides of (34)–(36). Furthermore, it holds that:

$$|\min(x, y)| = y^T x = \sum_{i=1}^{m} x_i y_i \tag{37}$$

FT3. Let F denote the matrix representation of FT3 in (25). By definition, we have

$$\tilde{v}_i(x) = \sum_{y \in \mathcal{S}_i} F^{-1}_{x,y} \phi_{\tilde{v}_i}(y) \tag{38}$$

$$= \sum_{y \in \mathcal{S}_i} \mathbb{I}_{\min(x,y)=y} \phi_{\tilde{v}_i}(y) \tag{39}$$

$$= \sum_{y \in \mathcal{S}_i} \mathbb{I}_{\min(1_m - x, y)=0_m} \phi_{\tilde{v}_i}(y) \tag{40}$$

$$= \langle (\mathbb{I}_{\min(1_m - x, y)=0_m})_{y \in \mathcal{S}_i}, \phi_{\tilde{v}_i|\mathcal{S}_i} \rangle, \tag{41}$$

where we used $x \cap y = y \Leftrightarrow x^c \cap y = \emptyset$. Now, the claim follows by observing that

$$\mathbb{I}_{\min(1_m - x, y)=0_m} = \max\left(0, 1 - y^T(1_m - x)\right), \tag{42}$$

which is a direct consequence of $|\min(1_m - x, y)| = y^T(1_m - x)$, and recalling that for each $y \in \mathcal{S}_i$ there is a respective row y^T in W_i.

FT4. Let F be the matrix representation of FT4 in (27). By definition, we have

$$\tilde{v}_i(x) = \sum_{y \in \mathcal{S}_i} F^{-1}_{x,y} \phi_{\tilde{v}_i}(y) \tag{43}$$

$$= \sum_{y \in \mathcal{S}_i} \mathbb{I}_{\min(x,y)=0_m} \phi_{\tilde{v}_i}(y) \tag{44}$$

$$= \langle (\mathbb{I}_{\min(x,y)=0_m})_{y \in \mathcal{S}_i}, \phi_{\tilde{v}_i|\mathcal{S}_i} \rangle. \tag{45}$$

Now, the claim follows by observing that

$$\mathbb{I}_{\min(x,y)=0_m} = \max\left(0, 1 - y^T x\right), \tag{46}$$

which is a direct consequence of $|\min(x, y)| = y^T x$, and recalling that for each $y \in \mathcal{S}_i$ there is a respective row y^T in W_i.

WHT. Let F be the matrix representation of WHT in (29). By definition, we have

$$\tilde{v}_i(x) = \sum_{y \in \mathcal{S}_i} F^{-1}_{x,y} \phi_{\tilde{v}_i}(y) \tag{47}$$

$$= \sum_{y \in \mathcal{S}_i} (-1)^{y^T x} \phi_{\tilde{v}_i}(y) \tag{48}$$

$$= \langle ((-1)^{y^T x})_{y \in \mathcal{S}_i}, \phi_{\tilde{v}_i|\mathcal{S}_i} \rangle. \tag{49}$$

Now, the claim follows by recalling that for each $y \in \mathcal{S}_i$ there is a respective row y^T in W_i. □

B.2 Proof of Theorem 1

We first state and proof two elementary lemmata.

For $x \in \mathbb{R}^d$ let $x \pmod 2$ be defined component-wise.

Lemma 2. (MAX REPRESENTATION) *Let $\zeta, \eta \in \mathbb{R}^d$ for $d \in \mathbb{N}$ and $C > 0$ such that for all $l \in \{1, \dots, d\}$ it holds that $|\zeta_l - \eta_l| \leq C$. Let $\alpha := \max(\zeta, \eta)$ and consider the polytope $\mathcal{P}$ in $(\tilde{\alpha}, \beta)$ defined by (50)–(54)*

$$\tilde{\alpha} \geq \eta \tag{50}$$

$$\tilde{\alpha} \leq \eta + C\beta \tag{51}$$

$$\tilde{\alpha} \geq \zeta \tag{52}$$

$$\tilde{\alpha} \leq \zeta + C(1_d - \beta) \tag{53}$$

$$\beta \in \{0,1\}^d. \tag{54}$$

Then it holds that $\mathcal{P} \neq \emptyset$ and every element $(\tilde{\alpha}, \beta) \in \mathcal{P}$ satisfies $\tilde{\alpha} = \alpha$.

Proof. Non-emptiness follows from the assumption that $|\zeta_l - \eta_l| \leq C$ for all $l \in \{1, \dots, d\}$. For any $l \in \{1, \dots, d\}$ we have to distinguish the following cases:

$$\zeta_l < \eta_l \implies \beta_l = 0,\ \tilde{\alpha}_l = \eta_l = \max(\zeta_l, \eta_l) = \alpha_l$$

$$\zeta_l > \eta_l \implies \beta_l = 1,\ \tilde{\alpha}_l = \zeta_l = \max(\zeta_l, \eta_l) = \alpha_l$$

$$\zeta_l = \eta_l \implies \tilde{\alpha}_l = \zeta_l = \eta_l = \max(\zeta_l, \eta_l) = \alpha_l$$

This yields that $\tilde{\alpha} = \alpha$. □

Lemma 3. (ODD-EVEN REPRESENTATION) *Let $\zeta \in \mathbb{Z}^d$ for $d \in \mathbb{N}$. Let $\beta := \zeta \pmod 2 \in \{0,1\}^d$ and consider the polytope $\mathcal{P}$ in $(\tilde{\beta}, \gamma)$ defined by (55) and (56)*

$$\tilde{\beta} = \zeta - 2\gamma \tag{55}$$

$$\tilde{\beta} \in \{0,1\}^d, \gamma \in \mathbb{Z}^d. \tag{56}$$

Then it holds that $\mathcal{P} \neq \emptyset$ and every element $(\tilde{\beta}, \gamma) \in \mathcal{P}$ satisfies $\tilde{\beta} = \beta$.

Proof. Non-emptiness follows since $\zeta \in \mathbb{Z}^d$ per assumption. For any $l \in \{1, \dots, d\}$ we have to distinguish the following cases:

$$\zeta_l \pmod 2 = 0 \implies \gamma_l = \frac{\zeta}{2}, \tilde{\beta}_l = 0 = \beta_l$$

$$\zeta_l \pmod 2 = 1 \implies \gamma_l = \frac{\zeta - 1}{2}, \tilde{\beta}_l = 1 = \beta_l$$

Thus $\tilde{\beta} = \beta$. □

For each bidder $i \in N$ let $\tilde{v}_i : \mathcal{X} \to \mathbb{R}_+$ be a Fourier-sparse approximation of the bidder's reported value function $\hat{v}_i$. Then the *Fourier transform-based WDP* was defined as follows:

Definition 3. *(*FOURIER TRANSFORM-BASED WDP*)*

$$\operatorname*{argmax}_{a \in \mathcal{F}} \sum_{i \in N} \tilde{v}_i(a_i). \tag{FT-WDP}$$

Next, we proof Theorem 1.

Theorem 1. (FOURIER TRANSFORM-BASED MIPS) *Let $\tilde{v}_i : \mathcal{X} \to \mathbb{R}$ be a k-Fourier-sparse approximation as defined in (31), (32), or (33). Then there exists a constant $C > 0$ such that the MIP defined by the following objective*

$$\operatorname*{argmax}_{a \in \mathcal{F}, \beta_i \in \{0,1\}^k} \sum_{i \in N} \langle \phi_{\tilde{v}_i | \mathcal{S}_i}, \alpha_i \rangle, \tag{57}$$

and for $i \in N$ one set of transform specific constraints (58)–(60), or (61)–(63), or (64)–(66), is equivalent to (FT-WDP).

$$\begin{aligned}
\textbf{\textit{FT3:}} \quad & \text{s.t. } \alpha_i \geq 1_k - W_i(1_m - a_i) && (58)\\
& \alpha_i \leq 1_k - W_i(1_m - a_i) + C\beta_i && (59)\\
& 0_k \leq \alpha_i \leq C(1_k - \beta_i) && (60)\\
\textbf{\textit{FT4:}} \quad & \text{s.t. } \alpha_i \geq 1_k - W_i a_i && (61)\\
& \alpha_i \leq 1_k - W_i a_i + C\beta_i && (62)\\
& 0_k \leq \alpha_i \leq C(1_k - \beta_i) && (63)\\
\textbf{\textit{WHT:}} \quad & \text{s.t. } \alpha_i = -2\beta_i + 1_k && (64)\\
& \beta_i = W_i a_i - 2\gamma_i && (65)\\
& \gamma_i \in \mathbb{Z}^k && (66)
\end{aligned}$$

Proof. We proof the equivalence for each of the FTs and corresponding MIPs separately.

- **FT3**
  Let $C > 0$ such that $|\,[1_k - W_i(1_m - a_i)]_l\,| \leq C$ for all $l \in \{1, \dots, k\}, i \in N$, and $a \in \mathcal{F}$. We then define

  $$\alpha_i := \max\left(0_k, 1_k - W_i(1_m - a_i)\right) \text{ for all } i \in N,$$

  and apply for each α_i Lemma 2 to $\zeta := 0_k$ and $\eta := 1_k - W_i(1_m - a_i)$, which concludes the proof for FT3.

- **FT4**:
  This follows analogously as for the FT3 when defining $\alpha_i := \max(0_k, 1_k - W_i a_i)$, $\zeta := 0_k$ and $\eta := 1_k - W_i a_i$.

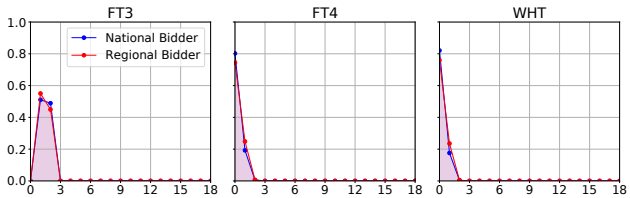


Figure 4: Spectral energy distribution in GSVM for different notions of FTs. For each cardinality (x-axis), we collect the spectral energy (y-axis) of all frequencies of that cardinality and normalize by the total spectral energy.

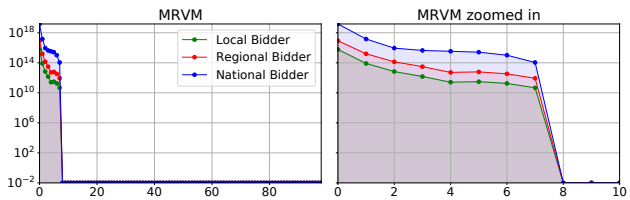


Figure 5: Approximate spectral energy distribution in MRVM for WHT computed with the robust WHT algorithm from [Amrollahi *et al.*, 2019]. For each cardinality (x-axis), we collect the spectral energy (y-axis) of all frequencies of that cardinality.

- **WHT**:
  Since for all $i \in N$ and $a \in \mathcal{F}$ it holds that $W_i a_i \in \mathbb{Z}^k$, an immediate consequence from Lemma 3 is that $\beta_i = W_i a_i \pmod 2$ and $(-1)^{W_i a_i} = -2\beta_i + 1_k$, which concludes the proof for the WHT.

□

C Analyzing the Potential of a FT-based CA

C.1 Fourier Sparsity - GSVM and MRVM

In this section, we provide the Fourier sparsity results for GSVM and MRVM. In Figure 4, we present the mean over 30 GSVM instances and bidder types. Figure 4 shows that GSVM is low degree ≤ 2 in all considered FTs and thus can be represented using less or equal to $172 = \binom{18}{0} + \binom{18}{1} + \binom{18}{2}$ FCs with any FT.

To the best of our knowledge it is not possible to compute the exact FCs with respect to any of the considered FTs for all MRVM bidders (see [Wendler *et al.*, 2021]). In Figure 5, we thus depict the *approximate* WHT-energy-distribution for MRVM. In order to do so, we used the RWHT algorithm by Amrollahi *et al.* [2019], which recovers the largest WHT-FCs with high probability. Note that here we do not normalize by the total energy since the total energy is an unknown quantity in the space of 2^{98} bundles.

Figure 5 shows the *approximate* WHT-energy-distribution for MRVM. We see that while most of the energy is contained in the low degree FCs ≤ 2, there is still some non negligible energy in the FCs of degree 3–7.

C.2 Reconstruction Error of Fourier Transforms

Using Procedure 1, we determine the best k-Fourier-sparse approximation $\tilde{v}_i$ for all considered FTs.

Procedure 1. (BEST FCS GIVEN FULL ACCESS TO $\hat{v}_i$)
Compute all FCs by using the full FT for each bidders' reported value function $\phi_{\hat{v}_i} = F\hat{v}_i$. Next, compute $\tilde{v}_i$ by determining the k best FCs $\phi_{\tilde{v}_i|\mathcal{S}_i}$ in terms of L2 error $\|\hat{v}_i - \tilde{v}_i\|_2^2$ as follows:

i. ***WHT:*** *use the FCs of $\phi_{\hat{v}_i}$ with the k largest absolute values.*

ii. ***FT3 and FT4:*** *use the FCs with the k largest coefficients $|\phi_{\hat{v}_i}(y)|\|F^{-1}_{\cdot,y}\|_2$, where $F^{-1}_{\cdot,y}$ denotes the yth column.*

In the following, we present the technical details for *i.* and *ii.* in Procedure 1.

Let F denote the corresponding matrix representation of the FT3, FT4, or the WHT. Furthermore, fix bidder $i \in N$ and for $\mathcal{S}_i \subseteq \text{supp}(\phi_{\hat{v}_i})$, $|\mathcal{S}_i| = k \ll 2^m$ let

$$\tilde{v}_i = \sum_{y \in \mathcal{S}_i} F^{-1}_{x,y}\phi_{\hat{v}_i}(y) = F^{-1}_{\cdot,\mathcal{S}_i}\phi_{\hat{v}_i|\mathcal{S}_i},$$

be a k-Fourier-sparse approximation, where we denote by $F^{-1}_{\cdot,\mathcal{S}_i}$ the sub matrix of F^{-1} obtained by selecting the columns indexed by the bundles in $\mathcal{S}_i$.

Best Fourier Coefficients WHT

For the WHT, we consider the following optimization problem of selecting the k-best FCs with respect to the quadratic error:

$$\mathcal{S}_i^* \in \underset{\mathcal{S}_i \subseteq \text{supp}(\phi_{\hat{v}_i}),\, |\mathcal{S}_i| = k}{\text{argmin}} \|\hat{v}_i - F^{-1}_{\cdot,\mathcal{S}_i}\phi_{\hat{v}_i|\mathcal{S}_i}\|_2^2. \tag{67}$$

Then, it follows that $\mathcal{S}_i^*$ consists out of those bundles $\{y^{(l)}\}_{l=1}^k$ with the largest absolute value of the corresponding FCs $\{\phi_{\hat{v}_i}(y^{(l)})\}_{l=1}^k$. This can be seen as follows.

$$\|\hat{v}_i - \underbrace{F^{-1}_{\cdot,\mathcal{S}_i}\phi_{\hat{v}_i|\mathcal{S}_i}}_{\tilde{v}_i}\|_2^2 = \|\sum_{y \notin \mathcal{S}_i} F^{-1}_{\cdot,y}\phi_{\hat{v}_i}(y)\|_2^2 = \tag{68}$$

$$= \sum_{x,y \notin \mathcal{S}_i} \langle F^{-1}_{\cdot,x}, F^{-1}_{\cdot,y}\rangle \phi_{\hat{v}_i}(x)\phi_{\hat{v}_i}(y) = \tag{69}$$

$$= \sum_{y \notin \mathcal{S}_i} \phi_{\hat{v}_i}(y)^2, \tag{70}$$

where in the last equality we used that F^{-1} is an orthogonal matrix and thus its columns fulfill $\langle F^{-1}_{\cdot,x}, F^{-1}_{\cdot,y}\rangle = \mathbb{I}_{x=y}$.

For the other non-orthogonal transforms FT3 and FT4, we use a heuristic based on the triangular inequality to select the k "best" FCs, which we present next.

Best Fourier Coefficients FT3 and FT4

For FT3 and FT4, we use the triangular inequality

$$\|\hat{v}_i - \underbrace{F^{-1}_{\cdot,\mathcal{S}_i}\phi_{\hat{v}_i|\mathcal{S}_i}}_{\tilde{v}_i}\|_2 \leq \sum_{y \notin \mathcal{S}_i} |\phi_{\hat{v}_i}(y)|\|F^{-1}_{\cdot,y}\|_2$$

to get an upper bound of the quadratic error $\|\hat{v}_i - \tilde{v}_i\|_2^2$ and select the FCs with the k largest coefficients $|\phi_{\hat{v}_i}(y)|\|F^{-1}_{\cdot,y}\|_2$, where $F^{-1}_{\cdot,y}$ denotes the yth column of F^{-1}.

D A Practical Hybrid ICA Mechanism

D.1 Technical Details of HYBRID ICA

In this section, we provide the technical details for the implementation of HYBRID ICA. In particular, we explain in more detail the Fourier Transform-based procedures 3–5.

Determining the best FCs of NN

Procedure 3. (BEST FCS OF NEURAL NETWORKS)

i. If $m \leq 29$: calculate the full FT of the NNs and select the best FCs.

ii. If $m > 29$: use sparse FT algorithms, i.e., RWHT for WHT and SSFT [Wendler et al., 2021] for FT3 and FT4 to obtain the best FCs of the NNs.

Using **Procedure 3** we determine the best FCs of the NN estimates of each bidder. We distinguish the two cases:

- **Small number of items:** if the number of items is small enough, i.e., $m \leq 29$, we determine the best FCs (and associated locations $\mathcal{S}_i \subset \mathcal{X}$) of $\mathcal{N}_i$ for all FTs by computing its full Fourier transform $F\mathcal{N}_i = \phi_{\mathcal{N}_i}$ using the respective algorithms from Püschel and Wendler [2020].
- **Large number of items:** if the number of items is too large, i.e., $m > 29$, we cannot compute the full Fourier transform and thus require sparse FT algorithms to compute the locations of the best FCs. For the WHT, we can do so by using the *robust sparse WHT algorithm (RWHT)* by Amrollahi *et al.* [2019]. For FT3 and FT4, we use the *sparse set function Fourier transform (SSFT)* algorithm by Wendler *et al.* [2021]. Both algorithms compute Fourier-sparse approximations by only using queries from the corresponding set function (= a bidder's value function), and thus do not require a representation of the exponentially large full set function.
- **Best WHT FCs:** for the WHT the best FCs are the ones with the largest absolute values (see Section C.2). We select the locations of the ℓ_{superset} best NN FCs, i.e., $|\mathcal{S}_i| = \ell_{\text{superset}}$. We do so because we are going to use the *compressive sensing method* by Stobbe and Krause [2012], which only requires a superset of the support $\text{supp}(\phi_{\hat{v}_i})$, in **Procedure 5** to fit a Fourier-sparse approximations $\tilde{v}_i$ to the bidders' reports R_i. In our experiments, we set $\ell_{\text{superset}} = 2,000$ for all domains. Ideally, we would like to choose ℓ_{superset} as large as possible to ensure that the best NN FCs actually overlap with the best FCs of $\hat{v}_i$, however, there is a trade-off between the number of samples (and also running time) required by the *compressive sensing method* and the size of the support superset. We did not optimize this hyperparameter.
- **Best FT3 and FT4 FCs:** for the FT3 and FT4 we use a heuristic based on the triangular inequality to select the best FCs and their locations (see Section C.2). As neither FT3 nor FT4 are orthogonal or satisfy the restricted isometry property required by the *compressive sensing method*, we only take the best ℓ_3 locations of NN FCs, i.e., $|\mathcal{S}_i| = \ell_3$.

Notice that the goal of **Procedure 3** is solely to determine a set of bundles $\mathcal{S}_i \subseteq \mathcal{X}$ that is likely to contain many of the dominant FCs of $\hat{v}_i$. This set $\mathcal{S}_i$ is then used to determine reconstructing queries $\tilde{\mathcal{S}}_i$.

Determining reconstruction queries

Procedure 4. (FOURIER RECONSTRUCTION QUERIES)

i. ***FT3 and FT4:*** *use the sampling theorems by Püschel and Wendler [2020] to determine queries for the bidders, i.e., bundles $\tilde{\mathcal{S}}_i \subseteq \mathcal{X}$, that enable a reconstruction of $\tilde{v}_i$.*

ii. ***WHT:*** *use the same queries as for FT4.*

We make use of the sampling theorems for FT3 and FT4 presented by Püschel and Wendler [2020] to obtain reconstruction queries, i.e., queries that help obtaining a small reconstruction error $\|\hat{v}_i - \tilde{v}_i\|_2$. Our rationale here is that the queries given by the sampling theorem $\tilde{\mathcal{S}}_i$ would lead to $\|\hat{v}_i - \tilde{v}_i\|_2 = 0$ for $\text{supp}(\phi_{\hat{v}_i}) = \mathcal{S}_i$. The sampling theorem by Püschel and Wendler [2020] selects rows $\tilde{\mathcal{S}}_i$ such that $F^{-1}_{\tilde{\mathcal{S}}_i,\mathcal{S}_i}$ is of full rank, because then (iff $\text{supp}(\phi_{\hat{v}_i}) = \mathcal{S}_i$) the Fourier coefficients $\phi_{\tilde{v}_i|\mathcal{S}_i}$ are the solution of the linear system of equations $\hat{v}_{i|\tilde{\mathcal{S}}_i} = F^{-1}_{\tilde{\mathcal{S}}_i,\mathcal{S}_i}\phi_{\tilde{v}_i|\mathcal{S}_i}$. In particular, the theorem yields

- **FT3:** $\tilde{\mathcal{S}}_i = \mathcal{S}_i$,
- **FT4:** $\tilde{\mathcal{S}}_i = \{1_m - y : y \in \mathcal{S}_i\}$.

For the WHT there is no sampling theorem and we use

$$\tilde{\mathcal{S}}_i = \{1_m - y : |\phi_{\mathcal{N}_i}(y)| \text{ is in the } \ell_3 \text{ largest in } \mathcal{S}_i\}$$

as a heuristic to obtain reconstruction queries. Choosing $\tilde{\mathcal{S}}_i$ in that way empirically often leads to a full-rank submatrix $F^{-1}_{\tilde{\mathcal{S}}_i,\mathcal{Z}}$ of the inverse WHT obtained by selecting the rows indexed by $\tilde{\mathcal{S}}_i$ and the columns indexed by $\mathcal{Z}$, where $\mathcal{Z} \subseteq \mathcal{S}_i$ are the locations of the ℓ_3 in absolute value largest FCs in $\mathcal{S}_i$, i.e., $\mathcal{Z} := \{y : |\phi_{\mathcal{N}_i}(y)| \text{ is in the } \ell_3 \text{ largest in } \mathcal{S}_i\}$. If $F^{-1}_{\tilde{\mathcal{S}}_i,\mathcal{Z}}$ is full rank and $\text{supp}(\phi_{\hat{v}_i}) = \mathcal{Z}$, the Fourier coefficients $\phi_{\hat{v}_i|\mathcal{Z}}$ are the solution of the linear system of equations $\hat{v}_{i|\tilde{\mathcal{S}}_i} = F^{-1}_{\tilde{\mathcal{S}}_i,\mathcal{Z}}\phi_{\hat{v}_i|\mathcal{Z}}$.

Fitting Fourier-sparse approximations

Procedure 5. (FIT FOURIER-SPARSE $\tilde{v}_i$ TO REPORTS)

i. ***FT3 and FT4:*** *solve the least squares problem defined by the best FCs and reports R_i.*

ii. ***WHT:*** *use the compressive sensing method by Stobbe and Krause [2012] defined by the best FCs and reports R_i.*

Lastly, we need to fit our Fourier-sparse approximations $\tilde{v}_i$ with $\text{supp}(\phi_{\tilde{v}_i}) = \mathcal{S}_i$ to the elicited reports $R_i := \{(x^{(l)}, \hat{v}_i(x^{(l)}))\}$. That is, we determine values $w \in \mathbb{R}^{|\mathcal{S}_i|}$ for the FCs of the Fourier-sparse approximation $\tilde{v}_i = F^{-1}_{\cdot,\mathcal{S}_i} w$.

- **WHT:** as already mentioned, for the WHT this is done using *the compressive sensing method* by Stobbe and Krause [2012]. We calculate the full regularization path for the L1-regularization parameter λ using the LARS method from Efron *et al.* [2004], and select the λ that yields ℓ_{support} non zero FCs. In our experiments we set $\ell_{\text{support}} = 100$ in GSVM and LSVM and $\ell_{\text{support}} = 500$ in MRVM. We did not optimize this hyperparameter.
- **FT3 and FT4:** for FT3 and FT4 we cannot use *the compressive sensing method* and instead solve the following least squares problem

$$\min_{w \in \mathbb{R}^{\ell_3}} \sum_{(x,\hat{v}_i(x)) \in R_i} \left(\hat{v}_i(x) - \underbrace{\left(F^{-1}_{\cdot,\mathcal{S}_i} w\right)(x)}_{=\tilde{v}_i(x)} \right)^2 .$$

In Figure 6, we present a flow diagram of the different algorithms we use in HYBRID ICA.

D.2 Details of NNs Support Discovery Experiments

In this section, we discuss the energy ratio plot (see Figure 3 in the main paper) for the large domain MRVM with $m = 98$ items. Recall, that in MRVM one cannot compute the full FT analytically and one has to use sparse FT algorithms to obtain an approximation of the true k-best possible frequencies $\mathcal{S}_i^* = \{\overset{*}{y}^{(1)}, \ldots, \overset{*}{y}^{(k)}\}$. For this we use the recently developed RWHT algorithm from Amrollahi *et al.* [2019], which computes the largest WHT-FCs with high probability and thus gives us an approximation $\mathcal{S}_i^{\text{rwht}} \approx \mathcal{S}_i^*$. As in the main paper, we then calculate for each bidder $i \in N$ the *energy ratio* of the support found by the corresponding NNs $\tilde{\mathcal{S}}_i$ and $\mathcal{S}_i^{\text{rwht}}$.

D.3 Details of Efficiency Experiments

In Figures 7–9, we present detailed efficiency results of HYBRID ICA in all three SATS domains, where we used the best found configuration of hyperparameters from Table 4 in the main paper.

In the upper plot we show in each figure the efficiency of the different phases of HYBRID ICA.[6] In the lower plot, we present a histogram of the final efficiency distribution over 100 (in GSVM and LSVM) and 30 (in MRVM) new CA instances. To enable a head-to-head comparison, we use for the test set of CA instances the seeds 1–100 in GSVM and LSVM, and seeds 51–80 in MRVM.

All experiments were conducted on machines with Intel Xeon E5 v4 2.20GHz processors with 24 cores and 128GB RAM or with Intel E5 v2 2.80GHz processors with 20 cores and 128GB RAM.

GSVM (Figure 7)

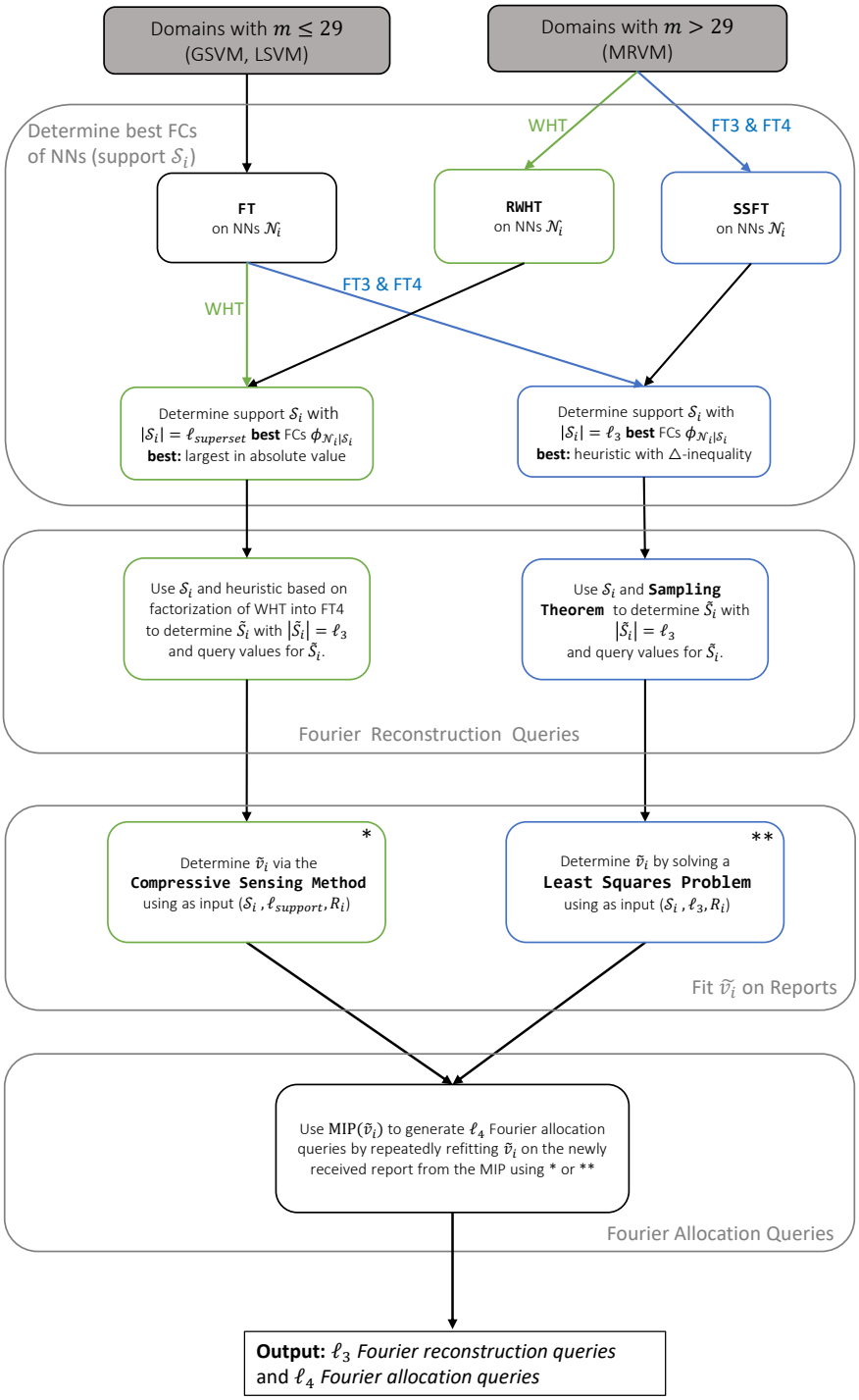
After 30 random queries, HYBRID ICA achieves an (avg.) efficiency of 66%. Next, after three MLCA iterations, i.e., 21 *MLCA queries*, 99 instances have an efficiency $\geq 90\%$.[7] Finally, the Fourier-based queries significantly increase the (avg.) efficiency by 1.87 percentage points from 98.1% to 99.97%. Furthermore, we present in the lower plot a histogram of the final efficiency distribution. We see, that for 94 out of 100 instances HYBRID ICA impressively achieves an economic efficiency of 100% using in total only 100 value queries per bidder.

[6] In MRVM we abbreviate the different query types as follows: RI=random initial queries, 1-55=MLCA iterations, FR=Fourier reconstruction queries, FA=Fourier allocation queries.

[7] Recall that *MLCA* asks each bidder n queries per iteration.

LSVM (Figure 8)
Starting with $\ell_1 = 30$ random initial queries, HYBRID ICA achieves an average efficiency of approximately 62%. Next, HYBRID ICA performs 5 MLCA iterations. After these 5 MLCA iterations, HYBRID ICA already found for each of the 100 instances an allocation with an efficiency of at least 80% with an average efficiency of 97.80%. Here, in the non-sparse LSVM domain, *the Fourier reconstruction and allocation queries* can increase the efficiency of some outliers arriving at an average efficiency of 98.74%. In the histogram, we see, that for 66 instances, HYBRID ICA was able to achieve full efficiency. Overall, we observe, that in the non-sparse LSVM the Fourier-based approach is not as effective as in the sparse GSVM, but still leads to results, that statistically match the efficiency of MLCA (see Table 5 in the main paper).

MRVM (Figure 9)
Starting with $\ell_1 = 30$ random initial queries, HYBRID ICA achieves an average efficiency of approximately 50%. Next, HYBRID ICA performs 55 MLCA iterations and asks in total 220 *MLCA allocation queries*. Here, we observe a steep increase in efficiency at the beginning. In later iterations the increase in efficiency gets smaller resulting in an average efficiency of 94.20%. In MRVM, the best query split we found uses $\ell_3 = 0$ *Fourier-based reconstruction queries*, thus the Fourier reconstruction queries (FR) do not change the efficiency distribution in Figure 9. However, the $\ell_4 = 250$ *Fourier-based allocation queries* significantly increase the efficiency further by 2.43% resulting in a final average efficiency of 96.63%.



Algorithms:
FT: (Full) Fourier transform, Püschel and Wendler (2020).
RWHT: Robust sparse WHT algorithm, Amrollahi et al. (2019).
SSFT: Sparse set function Fourier transform, Wendler et al. (2020)
Sampling Theorem: Püschel and Wendler (2020).
Compressive Sensing Method: Stobbe and Krause (2012).

Figure 6: Overview of Fourier Transform-based Procedures in HYBRID ICA.

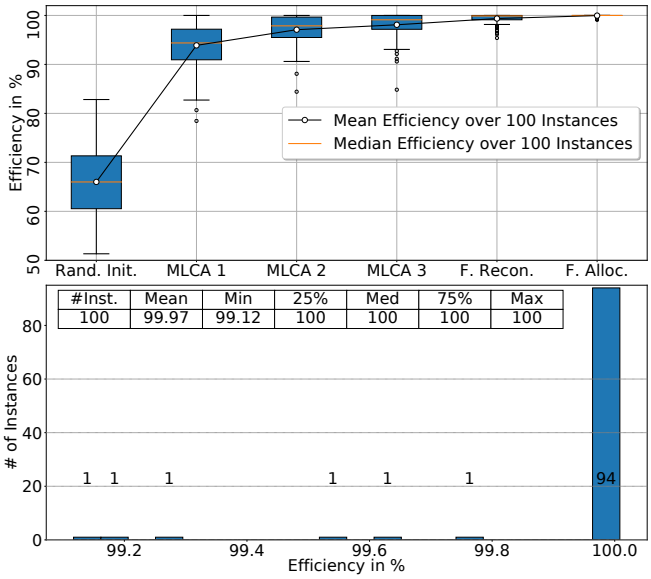


Figure 7: Details of HYBRID ICA in GSVM.

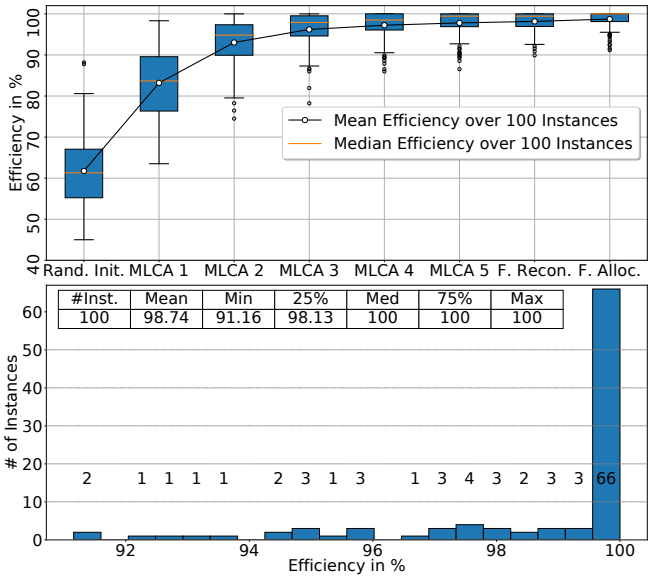


Figure 8: Details of HYBRID ICA in LSVM.

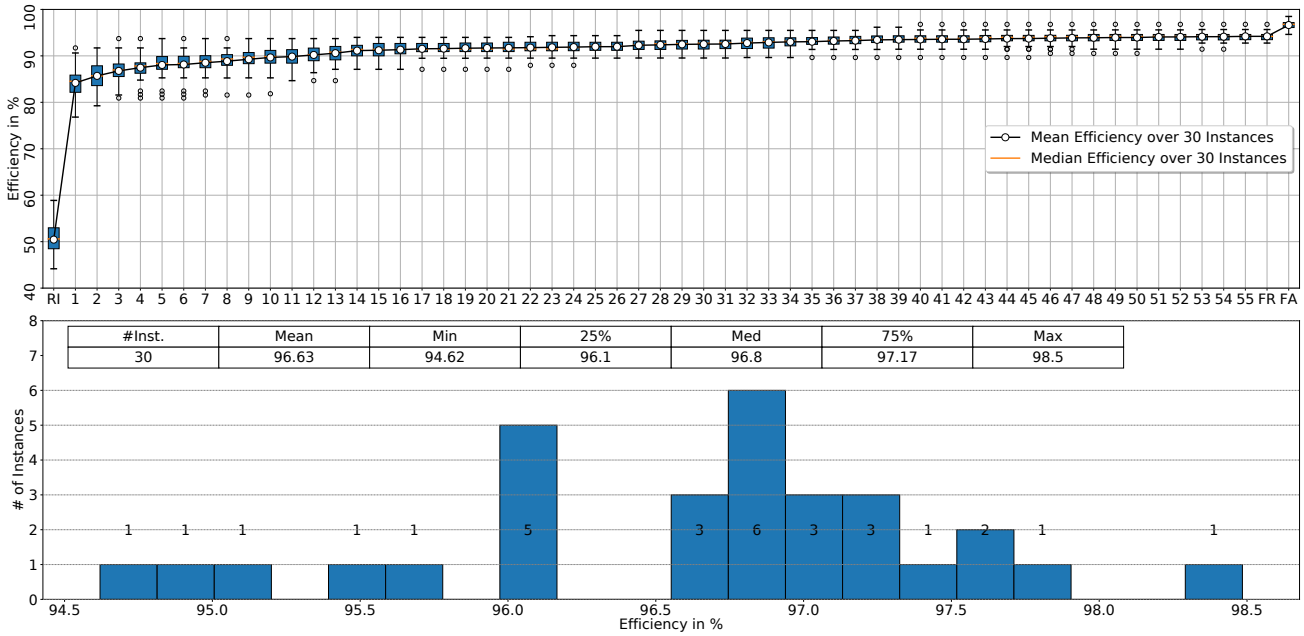


Figure 9: Details of HYBRID ICA in MRVM.

4 NOMU: Neural Optimization-based Model Uncertainty

The content of this chapter has previously appeared in

NOMU: Neural Optimization-based Model Uncertainty.
Jakob Heiss*, Jakob Weissteiner*, Hanna Wutte*, Sven Seuken, and Josef Teichmann.
In *Proceedings of the Thirty-ninth International Conference on Machine Learning* (ICML'22), Baltimore, USA, July 2022.

For its full updated version including appendix, please see

NOMU: Neural Optimization-based Model Uncertainty.
Jakob Heiss*, Jakob Weissteiner*, Hanna Wutte*, Sven Seuken, and Josef Teichmann.
Working paper, March 2023.
URL: arxiv.org/pdf/2102.13640.pdf

*These authors contributed equally.

NOMU: Neural Optimization-based Model Uncertainty

Jakob Heiss [*1 2] **Jakob Weissteiner** [*2 3] **Hanna Wutte** [*1 2] **Sven Seuken** [2 3] **Josef Teichmann** [1 2]

Abstract

We study methods for estimating model uncertainty for neural networks (NNs) in regression. To isolate the effect of model uncertainty, we focus on a noiseless setting with scarce training data. We introduce five important desiderata regarding model uncertainty that any method should satisfy. However, we find that established benchmarks often fail to reliably capture some of these desiderata, even those that are required by Bayesian theory. To address this, we introduce a new approach for capturing model uncertainty for NNs, which we call *Neural Optimization-based Model Uncertainty (NOMU)*. The main idea of NOMU is to design a network architecture consisting of two connected sub-NNs, one for model prediction and one for model uncertainty, and to train it using a carefully-designed loss function. Importantly, our design enforces that NOMU satisfies our five desiderata. Due to its modular architecture, NOMU can provide model uncertainty for any given (previously trained) NN if given access to its training data. We evaluate NOMU in various regressions tasks and noiseless Bayesian optimization (BO) with costly evaluations. In regression, NOMU performs at least as well as state-of-the-art methods. In BO, NOMU even outperforms all considered benchmarks.

1. Introduction

Neural networks (NNs) are becoming increasingly important in machine learning applications (LeCun et al., 2015). In many domains, it is essential to be able to quantify the *model uncertainty (epistemic uncertainty)* of NNs (Neal, 2012; Ghahramani, 2015). Good estimates of model uncertainty are indispensable in Bayesian optimization (BO) and active learning, where exploration is steered by (functions of) these uncertainty estimates. In recent years, BO has been successfully applied in practice to a wide range of problems, including robotics (Martinez-Cantin et al., 2009), sensor networks (Srinivas et al., 2012), and drug development (Gómez-Bombarelli et al., 2018). Better model uncertainty estimates for BO directly translate to improvements in these applications.

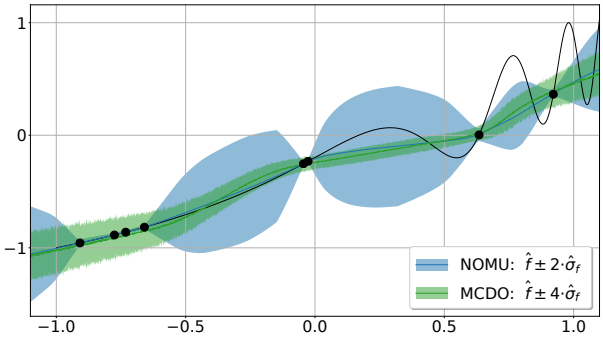


Figure 1. Visualization of estimated model uncertainty $\hat{\sigma}_f$. The unknown true function is depicted as a black solid line with training points as black dots. NOMU's model prediction $\hat{f}$ is shown as a solid blue line and its uncertainty bounds are shown as a blue shaded area. As a benchmark, MC Dropout is shown in green.

However, estimating model uncertainty well for NNs is still an open research problem. For settings with scarce training data and negligible data noise, where model uncertainty is the main source of uncertainty, we uncover deficiencies of widely used state-of-the-art methods for estimating model uncertainty for NNs. Prior work often only measures the performance in data noise dominant settings, and thus does not adequately isolate the pure model uncertainty, thereby overlooking the algorithms' deficiencies. However, in tasks such as BO with costly evaluations, where accurate estimates of model uncertainty are of utmost importance, these deficiencies can drastically decrease performance.

In this paper, we study the problem of estimating model uncertainty for NNs to obtain *uncertainty bounds (UBs)* that estimate Bayesian credible bounds in a setting with negligible data noise and scarce training data. For this, we propose a novel algorithm (NOMU) that is specialized to such a setting. Figure 1 shows UBs for NOMU and the benchmark method MC Dropout.

[*]Equal contribution [1]ETH Zurich [2]ETH AI Center [3]University of Zurich. Correspondence to: Jakob Weissteiner <weissteiner@ifi.uzh.ch>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

1.1. Prior Work on Model Uncertainty for NNs

Over the last decade, researchers have developed various methods to quantify model uncertainty for NNs. One strand of research considers Bayesian Neural Networks (BNNs), where distributions are placed over the NN's parameters (Graves, 2011; Blundell et al., 2015; Hernández-Lobato & Adams, 2015). However, variational methods approximating BNNs are usually computationally prohibitive and require careful hyperparameter tuning. Thus, BNNs are rarely used in practice (Wenzel et al., 2020a).

In practice, *ensemble methods* are more established:

- Gal & Ghahramani (2016) proposed *Monte Carlo dropout (MCDO)* to estimate model uncertainty via stochastic forward passes. Interestingly, they could show that training a NN with dropout can also be interpreted as variational inference approximating a BNN.
- Lakshminarayanan et al. (2017) experimentally evaluated ensembles of NNs and showed that they perform as well as or better than BNNs. They proposed using *deep ensembles (DE)*, which use NNs with two outputs for model prediction and data noise, and they estimate model uncertainty via the empirical standard deviation of the ensemble. DE is the most established state-of-the art ensemble method and has been shown to consistently outperform other ensemble methods (Ovadia et al., 2019; Fort et al., 2019; Gustafsson et al., 2020; Ashukha et al., 2020).
- Recently, Wenzel et al. (2020b) proposed *hyper deep ensembles (HDE)*, an extension of DE where additional diversity is created via different hyperparameters, and they showed that HDE outperforms DE.

Despite the popularity of MCDO, DE and HDE, our empirical results suggest that none of them reliably capture all essential features of model uncertainty: MCDO yields tubular bounds that do not narrow at observed data points (which can already be observed in Figure 1); DE and HDE can produce UBs that are sometimes unreasonably narrow in regions far from observed data or unreasonably wide at training points (as we will show in Section 4.1).

1.2. Overview of our Contribution

We present a new approach for estimating model uncertainty for NNs, which we call *neural optimization-based model uncertainty (NOMU)*. In contrast to a fully Bayesian approach (e.g., BNNs), where approximating the posterior for a realistic prior is in general very challenging, we estimate posterior credible bounds by directly enforcing essential properties of model uncertainty. Specifically, we make the following contributions:

1. We first introduce five desiderata that we argue model UBs should satisfy (Section 3.1).
2. We then introduce NOMU, which consists of a network architecture (Section 3.2) and a carefully-designed loss function (Section 3.3), such that the estimated UBs fulfill these five desiderata. NOMU is easy to implement, scales well to large NNs, and can be represented as a *single* NN without the need for further ensemble distillation (in contrast to MCDO, DE and HDE). Because of its modular architecture, it can easily be used to obtain UBs for already trained NNs.
3. We experimentally evaluate NOMU in various regression settings: in scarce and noiseless settings to isolate model uncertainty (Sections 4.1.1 and 4.1.2) and on real-word data-sets (Sections 4.1.3 and 4.1.4). We show that NOMU performs well across all these settings while state-of-the-art methods (MCDO, DE, and HDE) exhibit several deficiencies.[1]
4. Finally, we evaluate the performance of NOMU in high-dimensional Bayesian optimization (BO) and show that NOMU performs as well or better than all considered benchmarks (Section 4.2).

Our source code is available on GitHub: https://github.com/marketdesignresearch/NOMU.

1.3. Further Related Work

Nix & Weigend (1994) were among the first to introduce NNs with two outputs: one for model prediction and one for *data noise (aleatoric uncertainty)*, using the Gaussian negative log-likelihood as loss function. However, such a data noise output cannot be used as an estimator for model uncertainty (epistemic uncertainty); see Appendix G for details. To additionally capture model uncertainty, Kendall & Gal (2017) combined the idea of Nix & Weigend (1994) with MCDO.

Similarly, NNs with two outputs for lower and upper UBs, trained on specifically-designed loss functions, were previously considered by Khosravi et al. (2010) and Pearce et al. (2018). However, the method by Khosravi et al. (2010) again only accounts for data noise and does not consider model uncertainty. The method by Pearce et al. (2018) also does not take model uncertainty into account in the design of their loss function and only incorporates it via ensembles (as in DE).

Besides the state-of-the art ensemble methods HDE and DE, there exist several other papers on ensemble methods that,

[1] We also conducted experiments with (Blundell et al., 2015). However, we found that this method did not perform as well as the other considered benchmarks. Moreover, it was shown in (Gal & Ghahramani, 2016; Lakshminarayanan et al., 2017) that *deep ensembles* and *MC dropout* outperform the methods by (Hernández-Lobato & Adams, 2015) and (Graves, 2011), respectively. Therefore, we do not include (Graves, 2011; Blundell et al., 2015; Hernández-Lobato & Adams, 2015) in our experiments.

for example, promote their diversity on the function space (Wang et al., 2019) or reduce their computational cost (Wen et al., 2020; Havasi et al., 2021).

For classification, Malinin & Gales (2018) introduced prior networks, which explicitly model in-sample and out-of-distribution uncertainty, where the latter is realized by minimizing the reverse KL-distance to a selected flat point-wise defined prior. In a recent working paper (and concurrent to our work), Malinin et al. (2020a) report on progress extending their idea to regression. While the idea of introducing a separate loss for learning model uncertainty is related to NOMU, there are several important differences (loss, architecture, behavior of the model prediction, theoretical motivation; see Appendix E for details). Furthermore, their experiments suggest that DE still performs weakly better than their proposed method.

In contrast to BNNs, which perform approximate inference over the entire set of weights, Neural Linear Models (NLMs) perform *exact* inference on only the last layer. NLMs have been extensively benchmarked in (Ober & Rasmussen, 2019) against MCDO and the method from (Blundell et al., 2015). Their results suggest that MCDO and (Blundell et al., 2015) perform competitive, even to carefully-tuned NLMs.

Neural processes, introduced by Garnelo et al. (2018a;b), have been used to express model uncertainty for image completion tasks, where one has access to thousands of different images interpreted as functions f_i instead of input points x_i. See Appendix F for a detailed comparison of their setting to the setting we consider in this paper.

2. Preliminaries

In this section, we briefly review the classical Bayesian uncertainty framework for regression.

Let $X \subset \mathbb{R}^d, Y \subset \mathbb{R}$ and let $f\colon X \to Y$ denote the unknown ground truth function. Let $D^{\text{train}} := \{(x_i^{\text{train}}, y_i^{\text{train}}) \in X \times Y, i \in \{1, \dots, n^{\text{train}}\}\}$, with $n^{\text{train}} \in \mathbb{N}$ be i.i.d samples from the data generating process $y = f(x) + \varepsilon$, where $\varepsilon | x \sim \mathcal{N}(0, \sigma_{\text{n}}^2(x))$. We use σ_{n} to refer to the *data noise (aleatoric uncertainty)*. We refer to $(x_i^{\text{train}}, y_i^{\text{train}})$ as a *training point* and to x_i^{train} as an *input training point*.

In the remainder of this paper, we follow the classic Bayesian uncertainty framework by modelling the unknown ground truth function f as a random variable. Hence, with a slight abuse of notation, we use the symbol f to denote *both* the unknown ground truth function as well as the corresponding random variable. In Appendix I, we provide a mathematically rigorous formulation of the considered Bayesian uncertainty framework.

Given a prior distribution for f and known data noise σ_{n}, the posterior of f and y are well defined. The *model uncertainty (epistemic uncertainty)* $\sigma_f(x)$ is the posterior standard deviation of $f(x)$, i.e.,

$$\sigma_f(x) := \sqrt{\mathbb{V}[f(x)|D^{\text{train}}, x]}, \quad x \in X. \tag{1}$$

Assuming independence between f and ε, the variance of the predictive distribution of y can be decomposed as $\mathbb{V}[y|D^{\text{train}}, x] = \sigma_f^2(x) + \sigma_{\text{n}}^2(x)$. We present our algorithm for estimating model uncertainty $\hat{\sigma}_f$ for the case of zero or negligible data noise, i.e., $\sigma_{\text{n}} \approx 0$ (see Appendix C for an extension to $\sigma_{\text{n}} \gg 0$). For a given model prediction $\hat{f}$, the induced uncertainty bounds (UBs) are then given by $\left(\underline{UB}_c(x), \overline{UB}_c(x)\right) := \left(\hat{f}(x) \mp c\, \hat{\sigma}_f(x)\right)$, for $x \in X$ and a calibration parameter $c \geq 0$.[2]

3. The NOMU Algorithm

We now present NOMU. We design NOMU to yield a model prediction $\hat{f}$ and a model uncertainty prediction $\hat{\sigma}_f$, such that the resulting UBs $(\underline{UB}_c, \overline{UB}_c)$ fulfill five desiderata.

3.1. Desiderata

D1 (Non-Negativity) *The upper/lower UB between two training points lies above/below the model prediction $\hat{f}$, i.e., $\underline{UB}_c(x) \leq \hat{f}(x) \leq \overline{UB}_c(x)$ for all $x \in X$ and for $c \geq 0$. Thus, $\hat{\sigma}_f \geq 0$.*

By definition, for any given prior, the exact posterior model uncertainty σ_f is positive, and therefore Desideratum D1 should also hold for any estimate $\hat{\sigma}_f$.

D2 (In-Sample) *In the noiseless case ($\sigma_n \equiv 0$), there is zero model uncertainty at each input training point x^{train}, i.e., $\hat{\sigma}_f(x^{\text{train}}) = 0$. Thus, $\overline{UB}_c(x^{\text{train}}) = \underline{UB}_c(x^{\text{train}}) = \hat{f}(x^{\text{train}})$ for $c \geq 0$.*

In Appendix D.2, we prove that, for any prior that does not contradict the training data, the exact σ_f satisfies D2. Thus, D2 should also hold for any estimate $\hat{\sigma}_f$, and we argue that even in the case of non-zero small data noise, model uncertainty should be small at input training data points.

D3 (Out-of-Sample) *The larger the distance of a point $x \in X$ to the input training points in D^{train}, the wider the UBs at x, i.e., model uncertainty $\hat{\sigma}_f$ increases out-of-sample.*[3]

For D3 it is often not obvious which metric to choose to measure distances. Some aspects of this metric can be speci-

[2] Note that our UBs estimate *credible bounds (CBs)* $\underline{CB}$ and $\overline{CB}$, which, for $\alpha \in [0, 1]$, fulfill that $\mathbb{P}[f(x) \in [\underline{CB}, \overline{CB}] | D^{\text{train}}, x] = \alpha$. For $\sigma_{\text{n}} \equiv 0$, CBs are equal to *predictive bounds* $\underline{PB}, \overline{PB}$ with $\mathbb{P}[y \in [\underline{PB}, \overline{PB}] | D^{\text{train}}, x] = \alpha$. See Appendix A for an explanation.

[3] Importantly, D3 also promotes model uncertainty in gaps *between* input training points.

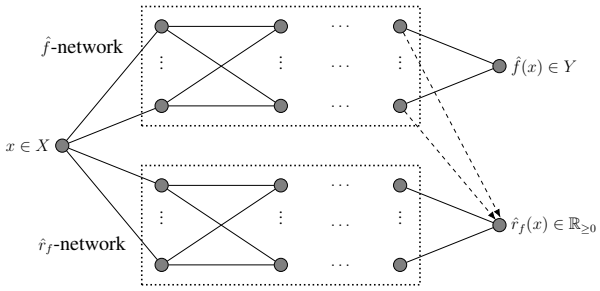


Figure 2. NOMU's network architecture

fied via the architecture (e.g., shift invariance for CNNs). In many applications, it is best to *learn* further aspects of this metric from training data, motivating our next desideratum.

D4 (Metric Learning) *Changes in those features of x that have high predictive power on the training set have a large effect on the distance metric used in D3.*[4]

D4 is not required for any application. However, specifically in *deep* learning applications, where it is a priori not clear which features are important, D4 is particularly desirable.

D5 (Vanishing) *As the number n^{train} of training points (with $x_i^{train} \overset{i.i.d}{\sim} \mathbb{P}_X$) tends to infinity, model uncertainty vanishes for each x in the support of the input data distribution $\mathbb{P}_X$, i.e., $\lim_{n^{train}\to\infty} \hat{\sigma}_f(x) = 0$ for a fixed $c \geq 0$. Thus, for a fixed c, $\lim_{n^{train}\to\infty} |\overline{UB}_c(x) - \underline{UB}_c(x)| = 0$.*

In Appendix D, we discuss all desiderata in more detail (see Appendix D.4 for a visualization of D4).

3.2. The Network Architecture

For NOMU, we construct a network $\mathcal{NN}_\theta$ with two outputs: *the model prediction* $\hat{f}$ (e.g., mean prediction) and a *raw* model uncertainty prediction $\hat{r}_f$. Formally: $\mathcal{NN}_\theta \colon X \to Y \times \mathbb{R}_{\geq 0}$, with $x \mapsto \mathcal{NN}_\theta(x) := (\hat{f}(x), \hat{r}_f(x))$. NOMU's architecture consists of two almost separate sub-networks: the $\hat{f}$-network and the $\hat{r}_f$-network (see Figure 2). For each sub-network, any network architecture can be used (e.g., feed-forward NNs, CNNs). This makes NOMU highly modular and we can plug in any previously trained NN for $\hat{f}$, or we can train $\hat{f}$ simultaneously with the $\hat{r}_f$-network. The $\hat{r}_f$-network learns the raw model uncertainty and is connected with the $\hat{f}$-network through the last hidden layer (dashed lines in Figure 2). This connection enables $\hat{r}_f$ to re-use features that are important for the model prediction $\hat{f}$, implementing Desideratum D4 (Metric Learning).[5]

Remark 3.1 *NOMU's network architecture can be modified to realize D4 (Metric Learning) in many different ways. For example, if low-level features were important for predicting the model uncertainty, one could additionally add connections from earlier hidden layers of the $\hat{f}$-network to layers of the $\hat{r}_f$-network. Furthermore, one can strengthen D4 (Metric Learning) by increasing the regularization of the $\hat{r}_f$-network (see Appendix D.4).*

After training $\mathcal{NN}_\theta$, we apply the readout map $\varphi(z) = \ell_{\max}(1 - \exp(-\frac{\max(0,z)+\ell_{\min}}{\ell_{\max}}))$, $\ell_{\min} \geq 0, \ell_{\max} > 0$ to the *raw* model uncertainty output $\hat{r}_f$ to obtain NOMU's *model uncertainty prediction*

$$\hat{\sigma}_f(x) := \varphi(\hat{r}_f(x)), \forall x \in X. \tag{2}$$

The readout map φ monotonically interpolates between a minimal $\approx \ell_{\min}$ and a maximal $\approx \ell_{\max}$ model uncertainty (see Figure 28 in Appendix J for a visualization). Here, $\ell_{\min}$ is used for numerical stability, and $\ell_{\max}$ defines the maximal model uncertainty far away from input training points (similarly to the prior variance for RBF-GPs). With NOMU's model prediction $\hat{f}$, its model uncertainty prediction $\hat{\sigma}_f$ defined in (2), and given a calibration parameter[6] $c \in \mathbb{R}_{\geq 0}$, we can now define for each $x \in X$ NOMU's UBs as

$$\left(\underline{UB}_c(x), \overline{UB}_c(x)\right) := \left(\hat{f}(x) \mp c\, \hat{\sigma}_f(x)\right). \tag{3}$$

It is straightforward to construct a *single* NN that directly outputs the upper/lower UB, by extending the architecture shown in Figure 2: we monotonically transform and scale the output $\hat{r}_f(x)$ and then add/subtract this to/from the other output $\hat{f}(x)$. It is also straightforward to compute NOMU's UBs for any *given, previously trained NN*, by attaching the $\hat{r}_f$-network to the trained NN, and only training the $\hat{r}_f$-network on the same training points as the original NN.

Remark 3.2 *The readout map φ can be modified depending on the subsequent use of the estimated UBs. For example, for BO over discrete domains (e.g., $X = \{0,1\}^d$) (Baptista & Poloczek, 2018), we propose the linearized readout map $\varphi(z) = \ell_{min} + \max(0, z - \ell_{min}) - \max(0, z - \ell_{max})$. With this φ and ReLU activations, one can encode NOMU's UBs as a mixed integer program (MIP) (Weissteiner & Seuken,*

[4] Consider the task of learning facial expressions from images. For this, eyes and mouth are important features, while background color is not. A CNN automatically learns which features are important for model prediction. The same features are also important for model uncertainty: Consider an image with pixel values similar to those of an image of the training data, but where mouth and eyes are very different. We should be substantially more uncertain about the model prediction for such an image than for one which is almost identical to a training image except that it has a different background color, even if this change of background color results in a huge Euclidean distance of the pixel vectors. D4 requires that a more useful metric is learned instead.

[5] To prevent that $\hat{r}_f$ impacts $\hat{f}$, the dashed lines should only make forward passes when trained.

[6] Like all other methods, NOMU outputs *relative* UBs that should be calibrated, e.g., via a parameter $c \geq 0$. See also Kuleshov et al. (2018) for a non-linear calibration method.

2020). This enables optimizing the upper UB as acquisition function without the need for further approximation via ensemble distillations (Malinin et al., 2020b).

3.3. The Loss Function

We now introduce the loss function L^π we use for training NOMU's architecture. Let X be such that $0 < \lambda_d(X) < \infty$, where λ_d denotes the d-dimensional Lebesgue measure. We train $\mathcal{NN}_\theta$ with loss L^π and L2-regularization parameter $\lambda > 0$, i.e., minimizing $L^\pi(\mathcal{NN}_\theta) + \lambda \|\theta\|_2^2$ via a gradient descent-based algorithm.

Definition 3.3 (NOMU Loss) *Let $\pi := (\pi_{sqr}, \pi_{exp}, c_{exp}) \in \mathbb{R}^3_{\geq 0}$ denote a tuple of hyperparameters. Given a training set D^{train}, the loss function L^π is defined as*

$$L^\pi(\mathcal{NN}_\theta) := \underbrace{\sum_{i=1}^{n^{train}} (\hat{f}(x_i^{train}) - y_i^{train})^2}_{(a)} + \pi_{sqr} \cdot \underbrace{\sum_{i=1}^{n^{train}} (\hat{r}_f(x_i^{train}))^2}_{(b)} + \pi_{exp} \cdot \underbrace{\frac{1}{\lambda_d(X)} \int_X e^{-c_{exp} \cdot \hat{r}_f(x)} \, dx}_{(c)} . \quad (4)$$

In the following, we explain how the three terms of L^π promote the desiderata we introduced in Section 3.1. Note that the behaviour of $\hat{r}_f$ directly translates to that of $\hat{\sigma}_f$.

- Term (a) solves the regression task, i.e., learning a smooth function $\hat{f}$ given D^{train}. If $\hat{f}$ is given as a pre-trained NN, then this term can be omitted.
- Term (b) implements D2 (In-Sample) and D5 (Vanishing) (i.e., $\hat{r}_f(x_i^{\text{train}}) \approx 0$). The hyperparameter π_{sqr} controls the amount of uncertainty at the training points.[7] The larger π_{sqr}, the narrower the UBs at training points.
- Term (c) has two purposes. First, it implements D1 (Non-Negativity) (i.e., $\hat{r}_f \geq 0$). Second, it pushes $\hat{r}_f$ towards infinity across the whole input space X. However, due to the counteracting force of (b) as well as regularization, $\hat{r}_f$ increases continuously as you move away from the training data. The interplay of (b), (c), and regularization thus promotes D3 (Out-of-Sample). The hyperparameters π_{exp} and c_{exp} control the size and shape of the UBs. Concretely, the larger π_{exp}, the wider the UBs; the larger c_{exp}, the narrower the UBs at points x with large $\hat{\sigma}_f(x)$ and the wider the UBs at points x with small $\hat{\sigma}_f(x)$.

In Appendix H.1, we provide detailed visualizations on how the loss hyperparameters π_{sqr}, π_{exp}, and c_{exp} shape NOMU's model uncertainty estimate. In the implementation of L^π, we approximate (c) via MC-integration using additional, *artificial input points* $D^{\text{art}} := \{x_i\}_{i=1}^{l} \overset{i.i.d}{\sim} \text{Unif}(X)$ by $\frac{1}{l} \cdot \sum_{x \in D^{\text{art}}} e^{-c_{\text{exp}} \cdot \hat{\sigma}_f(x)}$.

[7] In the noiseless case, in theory $\frac{\pi_{\text{sqr}}}{\lambda} = \infty$; we set $\frac{\pi_{\text{sqr}}}{\lambda} = 10^7$. For small non-zero data noise, setting $\frac{\pi_{\text{sqr}}}{\lambda} \ll \infty$ captures *data noise induced model uncertainty* $\sigma_f(x^{\text{train}}) > 0$ (Appendix D.2).

Remark 3.4 *In (c), instead of the Lebesgue-measure, one can also use a different measure ν, i.e., $\frac{1}{\nu(X)} \int_X e^{-c_{exp} \cdot \hat{r}_f(x)} \, d\nu(x)$. This can be relevant in high dimensions, where meaningful data points often lie close to a lower-dimensional manifold (Cayton, 2005); ν can then be chosen to concentrate on that region. In practice, this can be implemented by sampling from a large unlabeled data set D^{art} representing this region or learning the measure ν using GANs (Goodfellow et al., 2014).*

Theory In Appendix D, we prove that NOMU fulfills D1, D2 and D5 (Propositions D.1.a, D.2.c and D.5.a), and discuss how NOMU fulfills D3 and D4. In Appendix A.1, we show that, under certain assumptions, NOMU's UBs can be interpreted as *pointwise worst-case* UBs $\overline{UB}_{\text{pw}}(x) := \sup_{f \in \mathcal{H}_{D^{\text{train}}}} f(x)$ within a hypothesis class $\mathcal{H}_{D^{\text{train}}}$ of data-explaining functions. In Appendix A.2, we explain how $\underline{UB}_{\text{pw}}(x)$ and $\overline{UB}_{\text{pw}}(x)$ estimate posterior CBs of BNNs (with a Gaussian prior on the weights), without performing challenging variational inference. However, while exact posterior CBs of BNNs lose D4 as their width goes to infinity, NOMU's UBs are capable of retaining D4 in this limit.

4. Experimental Evaluation

In this section, we experimentally evaluate NOMU's model uncertainty estimate in multiple synthetic and real-world regression settings (Section 4.1) as well as in high-dimensional Bayesian optimization (Section 4.2).

Benchmarks We compare NOMU against four benchmarks, each of which gives a model prediction $\hat{f}$ and a model uncertainty prediction $\hat{\sigma}_f$ (see Appendix B.1 for formulas). We calculate model-specific UBs at $x \in X$ as $(\hat{f}(x) \mp c\, \hat{\sigma}_f(x))$ with calibration parameter $c \in \mathbb{R}_{\geq 0}$ and use them to evaluate all methods. We consider three algorithms that are specialized to model uncertainty for NNs: (i) MC dropout (MCDO) (ii) deep ensembles (DE) and (iii) hyper deep ensembles (HDE) and a non-NN-based benchmark: (iv) Gaussian process (GP) with RBF kernel.

4.1. Regression

To develop intuition, we first study the *model* UBs of all methods on synthetic test functions with 1D–2D scarce input training points without data noise (Section 4.1.1). We then propose a novel generative test-bed and evaluate NOMU within this setting (Section 4.1.2). Next, we analyze NOMU on a real-world time series (Section 4.1.3). Finally, we evaluate NOMU on the real-world UCI data sets (Section 4.1.4).

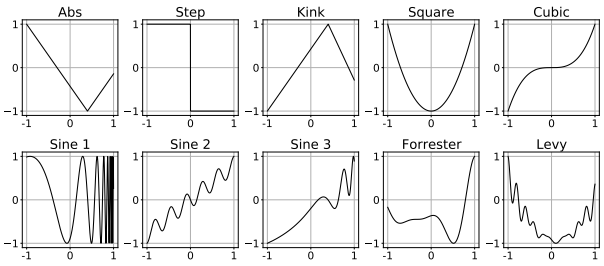


Figure 3. 1D synthetic test functions

4.1.1. Toy Regression

Setting We consider ten different 1D functions whose graphs are shown in Figure 3. Those include the popular Levy and Forrester function with multiple local optima.[8] All functions are transformed to $X := [-1, 1] =: f(X)$. For each function, we conduct 500 runs. In each run, we randomly sample eight noiseless input training points from X, such that the only source of uncertainty is model uncertainty. For each run, we also generate 100 *test points* in the same fashion to assess the quality of the UBs.

Metrics We report the *average negative log (Gaussian) likelihood (NLL)*, minimized over the calibration parameter c, which we denote as NLL_{min}. Following prior work (Khosravi et al., 2010; Kuleshov et al., 2018; Pearce et al., 2018), we further measure the quality of UBs by contrasting their *mean width* (MW) with their *coverage probability* (CP). Ideally, MW should be as small as possible, while CP should be close to 1. Since CP is counter-acting MW, we consider ROC-like curves, plotting MW against CP for a range of calibration parameters c, and report the *area under the curve (AUC)* (see Appendix B.2.1 for details).

Algorithm Setup For each of the two NOMU subnetworks, we use a feed-forward NN with three fully-connected hidden layers à 2^{10} nodes, ReLUs, and hyperparameters $\pi_{\text{exp}} = 0.01, \pi_{\text{sqr}} = 0.1, c_{\text{exp}} = 30$. In practice, the values for π_{exp}, π_{sqr}, and c_{exp} can be tuned on a validation set. However, for all synthetic experiments (Section 4.1.1 and Section 4.1.2), we use the same values, which lead to good results across all functions. Moreover, we set $\lambda = 10^{-8}$ accounting for zero data-noise, $\ell_{\text{min}} = 0.001$ and $\ell_{\text{max}} = 2$. In Appendix H.2, we provide an extensive sensitivity analysis for the hyperparameters π_{exp}, π_{sqr}, c_{exp}, ℓ_{min} and ℓ_{max}. This analysis demonstrates NOMU's robustness within a certain range of hyperparameter values.

Furthermore, to enable a fair comparison of all methods, we use generic representatives and do not optimize any of them for any test function. We also choose the architectures of all NN-based methods, such that the overall number of parameters is comparable. We provide all methods with the same prior information (specifically, this entails knowledge of zero data noise), and set the corresponding parameters accordingly. Finally, we set all remaining hyperparameters of the benchmarks to the values proposed in the literature. Details on all configurations are provided in Appendix B.2.2.

[8] See sfu.ca/ ssurjano/optimization.html.

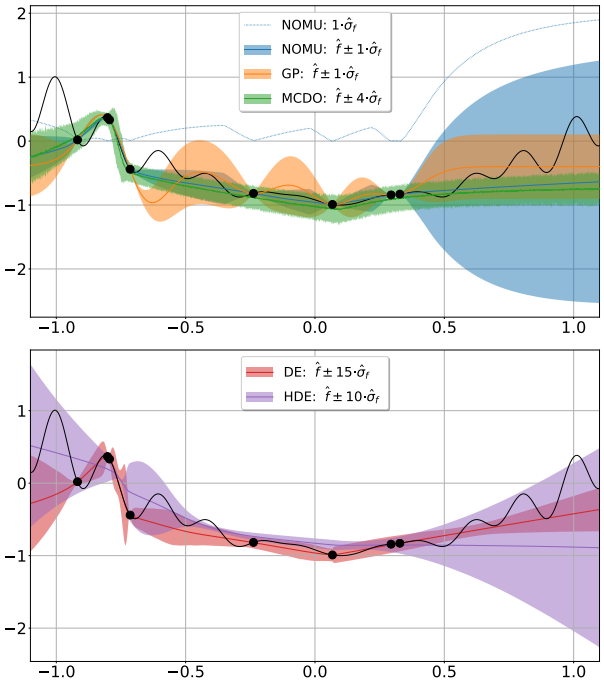


Figure 4. UBs resulting from NOMU, GP, MCDO, and DE, HDE for the Levy function (solid black line). For NOMU, we also show $\hat{\sigma}_f$ as a dotted blue line. Training points are shown as black dots.

Results Figure 4 exemplifies our findings, showing typical UBs for the Levy function as obtained in one run. In Appendix B.2.4 we provide further visualisations. We find that MCDO consistently yields tube-like UBs; in particular, its UBs do not narrow at training points, i.e., failing in D2 (In-Sample) even though MCDO's Bayesian interpretation requires D2 to hold (see Appendix D.2). Moreover, it only fulfills D3 (Out-of-Sample) to a limited degree. We frequently observe that DE leads to UBs of somewhat arbitrary shapes. This can be seen most prominently in Figure 4 around $x \approx -0.75$ and at the edges of its input range, where DE's UBs are very different in width with no clear justification. Thus, also DE is limited in D3 (Out-of-Sample). In addition, we sometimes see that also DE's UBs do not narrow sufficiently at training points, i.e., not fulfilling D2 (In-Sample). HDE's UBs are even more random, i.e., predicting large model uncertainty at training points and some-

Table 1. Ranks (1=best to 5=worst) for AUC and NLL_{min}.

	NOMU		GP		MCDO		DE		HDE	
FUNCTION	AUC	NLL_{MIN}	AUC	NLL_{MIN}	AUC	NLL_{MIN}	AUC	NLL_{MIN}	AUC	NLL_{MIN}
ABS	1	1	3	1	3	4	1	1	5	5
STEP	2	2	4	3	2	3	1	1	5	5
KINK	1	1	3	1	4	4	1	1	5	5
SQUARE	2	2	2	1	4	4	2	3	5	5
CUBIC	2	2	1	1	3	4	3	3	5	5
SINE 1	2	1	2	1	1	1	2	1	5	5
SINE 2	2	2	3	1	1	2	3	3	5	5
SINE 3	1	1	4	1	3	4	1	1	5	5
FORRESTER	1	2	1	1	3	4	3	3	5	5
LEVY	1	1	4	3	1	4	3	1	5	5

times zero model uncertainty in gaps between them (e.g., $x \approx -0.75$). In contrast, NOMU displays the behaviour it is designed to show. Its UBs nicely tighten at training points and expand in-between (D1–D3, for D4 (Metric Learning) see Appendix D.4). Like NOMU, the GP fulfills D2 (In-Sample) and D3 (Out-of-Sample) well, but cannot account for D4 (Metric Learning) (a fixed kernel does not depend on the model prediction). Table 1 provides the ranks achieved by each algorithm (see Appendix B.2.3 for corresponding metrics). We calculate the ranks based on the medians and a 95% bootstrap CI of AUC and NLL_{min}. An algorithm loses one rank to each other algorithm that significantly dominates it. Winners are marked in grey. We observe that NOMU is the only algorithm that never comes in third place or worse. Thus, while some algorithms do particularly well in capturing uncertainties of functions with certain characteristics (e.g., RBF-GPs for polynomials), NOMU is the only algorithm that consistently performs well. HDE's bad performance can be explained by its randomness and the fact that it sometimes predicts zero model uncertainty out-of-sample. For 2D, we provide results and visualizations in Appendix B.2.3 and B.2.4 highlighting similar characteristics of all the algorithms as in 1D.

4.1.2. GENERATIVE TEST-BED

Setting Instead of only relying on a *limited* number of data-sets, we also evaluate all algorithms on a generative test-bed, which provides an unlimited number of test-functions. The importance of using a test-bed (to avoid over-fitting on specific data-sets) has also been highlighted in a recent work by Osband et al. (2021). For our test-bed, we generate 200 different data-sets by randomly sampling 200 different test-functions from a BNN with i.i.d centered Gaussian weights and three fully-connected hidden layers with nodes $[2^{10}, 2^{11}, 2^{10}]$ and ReLU activations. From each of these test-functions we uniformly at random sample $n^{\text{train}} = 8 \cdot d$ input training points and $100 \cdot d$ test data points, where d refers to the input dimension. We train all algorithms on these training sets and determine the NLL on the corresponding test sets averaged over the 200 test-functions. This metric converges to the Kullback-Leibler divergence to the exact BNN-posterior (see Theorem B.7). We calibrate by choosing per dimension an optimal value of c in terms of average NLL, which does not depend on the test-function.

Table 2. Average NLL (without const. $\ln(2\pi)/2$) and a 95% CI over 200 BNN samples. Winners are marked in grey.

FUNCTION	NOMU	GP	MCDO	DE	HDE
BNN1D	-1.65±0.10	-1.08±0.22	-0.34±0.23	-0.38±0.36	8.47±1.00
BNN2D	-1.16±0.05	-0.52±0.11	-0.33±0.13	-0.77±0.07	9.11±0.39
BNN5D	-0.37±0.02	-0.33±0.02	-0.05±0.04	-0.13±0.03	8.41±1.00

Results In Table 2, we provide the results for input dimensions 1, 2 and 5. We see that NOMU outperforms all other algorithms including MCDO, which is a variational inference method to approximate this posterior (Gal & Ghahramani, 2016). See Appendix B.2.5 for a detailed explanation of the experiment setting and further results in modified settings including a discussion of higher dimensional settings.

4.1.3. SOLAR IRRADIANCE TIME SERIES

Setting Although the current version of NOMU is specifically designed for scarce settings without data noise, we are also interested to see how well it performs in settings where these assumptions are not satisfied. To this end, we now study a setting with many training points and small non-zero data noise. This allows us to analyze how well NOMU captures D5 (Vanishing). We consider the popular task of interpolating the solar irradiance data (Steinhilber et al., 2009) also studied in (Gal & Ghahramani, 2016). We scale the data to $X = [-1, 1]$ and split it into 194 training and 197 test points. As in (Gal & Ghahramani, 2016), we choose five intervals to contain only test points. Since the true function is likely of high frequency, we set $\lambda = 10^{-19}$ for NOMU and the benchmarks' regularization accordingly. To account for small non-zero data noise, we set $\pi_{\text{exp}} = 0.05$, $\ell_{\text{min}} = 0.01$ and use otherwise the same hyperparameters as in Section 4.1.1.

Results Figure 5 visualizes NOMU's UBs. We see that NOMU manages to fit the training data well while capturing model uncertainty between input training points. In particu-

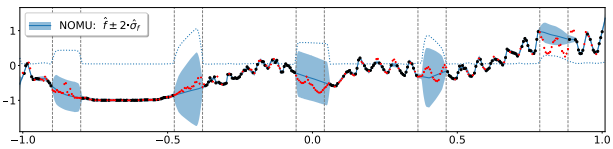


Figure 5. NOMU's model prediction (solid), model uncertainty (dotted) and UBs (shaded area) on the solar irradiance data. Training and test points are shown as black dots and red crosses.

Table 3. Average NLL and a 95% normal-CI over 20 runs for UCI data sets. Winners are marked in grey.

DATASET	NOMU	DE	MCDO	MCDO2	LL	NLM-HPO	NLM
BOSTON	2.68 ±0.11	2.41 ±0.49	2.46 ±0.11	2.40 ±0.07	2.57 ±0.09	2.58 ±0.17	3.63 ±0.39
CONCRETE	3.05 ±0.06	3.06 ±0.35	3.04 ±0.03	2.97 ±0.03	3.05 ±0.07	3.11 ±0.09	3.12 ±0.09
ENERGY	0.77 ±0.06	1.38 ±0.43	1.99 ±0.03	1.72 ±0.01	0.82 ±0.05	0.69 ±0.05	0.69 ±0.05
KIN8NM	-1.08 ±0.01	-1.20 ±0.03	-0.95 ±0.01	-0.97 ±0.00	-1.23 ±0.01	-1.12 ±0.01	-1.13 ±0.01
NAVAL	-5.63 ±0.39	-5.63 ±0.09	-3.80 ±0.01	-3.91 ±0.01	-6.40 ±0.11	-7.36 ±0.15	-7.35 ±0.01
CCPP	2.79 ±0.01	2.79 ±0.07	2.80 ±0.01	2.79 ±0.01	2.83 ±0.01	2.79 ±0.01	2.79 ±0.01
PROTEIN	2.79 ±0.01	2.83 ±0.03	2.89 ±0.00	2.87 ±0.00	2.89 ±0.00	2.78 ±0.01	2.81 ±0.00
WINE	1.08 ±0.04	0.94 ±0.23	0.93 ±0.01	0.92 ±0.01	0.97 ±0.03	0.96 ±0.01	1.48 ±0.09
YACHT	1.38 ±0.28	1.18 ±0.41	1.55 ±0.05	1.38 ±0.01	1.01 ±0.09	1.17 ±0.13	1.13 ±0.09

lar, large gaps *between* input training points are successfully modeled as regions of high model uncertainty (D3 (Out-of-Sample)). Moreover, in regions highly populated by input training points, NOMU's model uncertainty vanishes as required by D5 (Vanishing). Plots for the other algorithms are provided in Appendix B.2.6, where we observe similar patterns as in Section 4.1.1.

4.1.4. UCI DATA SETS

Setting Recall that NOMU is specifically tailored to noiseless settings with scarce input training data. However, even the current version of NOMU, which does not explicitly model data noise, already performs on par with existing benchmarks on real-world regression tasks *with* data noise. To demonstrate this, we test its performance on (a) the UCI data sets proposed in Hernández-Lobato & Adams (2015), a common benchmark for uncertainty quantification in noisy, real-world regression, and (b) the UCI gap data set extension proposed in Foong et al. (2019). We consider exactly the same experiment setup as proposed in these works, with a $70/20/10$-train-validation-test split, equip NOMU with a shallow architecture of 50 hidden nodes, and train it for 400 epochs. Validation data are used to calibrate the constant c on NLL. See Appendix B.2.7 for details on NOMU's setup.

Results Table 3 reports NLLs on test data, averaged across 20 splits, compared to the current state-of-the-art. We report the NLL for MCDO (Gal & Ghahramani, 2016) and DE (Lakshminarayanan et al., 2017) from the original papers. Moreover, we reprint the best results (for comparable network sizes) of neural linear models (NLMs) with and without hyperparameter optimization (HPO) from (Ober & Rasmussen, 2019) (NLM-HPO, NLM), linearized laplace (LL) (Foong et al., 2019) and a recent strong MCDO baseline (MCDO2) from (Mukhoti et al., 2018). It is surprising that, even though the current design of NOMU does not yet explicitly incorporate data noise, it already performs comparably to state-of-the-art results. We obtain similar results for the UCI gap data. See Appendix B.2.7 for more details on this experiment.

4.2. Bayesian Optimization

In this section, we assess the performance of NOMU in high-dimensional noiseless *Bayesian optimization (BO)*.

Setting In BO, the goal is to maximize an unknown expensive-to-evaluate function, given a budget of function queries. We use a set of test functions with different characteristics from the same library as before, but now in 5 to 20 dimensions d, transformed to $X = [-1, 1]^d$, $f(X) = [-1, 1]$.[9] Additionally, we again use Gaussian BNNs to create a generative test-bed consisting of a large variety of test functions (see Section 4.1.2). For each test function, we randomly sample 8 initial points $(x_i, f(x_i))$ and let each algorithm choose 64 further function evaluations (one by one) using its upper UB as acquisition function. This corresponds to a setting where one can only afford 72 expensive function evaluations in total. We provide details regarding the selected hyperparameters for each algorithm in Appendix B.3.1. We measure the performance of each algorithm based on its *final regret* $|\max_{x \in X} f(x) - \max_{i \in \{1,\dots,72\}} f(x_i)| / |\max_{x \in X} f(x)|$.

For each algorithm, the UBs must be calibrated by choosing appropriate values of c. We do so in the following straightforward way: First, after observing the 8 random initial points, we determine those two values of c for which the resulting mean width (MW) of the UBs is 0.05 and 0.5, respectively (MW SCALING).[10] We perform one BO run for both resulting initial values of c. Additionally, if in a BO run, an algorithm were to choose a point $x_{i'}$ very close to an already observed point x_i, we dynamically increase c to make it select a different one instead (DYNAMIC C; see Appendix B.3.2 for details). A value of 0.05 in MW SCALING corresponds to small model uncertainty, such that exploration is mainly due to DYNAMIC C. Smaller values than 0.05 thus lead to similar outcomes. In contrast, a value of 0.5 corresponds to large model uncertainties, such that DYNAMIC C is rarely used. Only for the "plain GP (pGP)" we use neither MW SCALING nor DYNAMIC C, as pGP uses its default calibration (c is determined by the built-in hyperparameter optimization in every step). However, a comparison of GP and pGP suggests that MW SCALING and DYNAMIC C surpass the built-in calibration (see Table 4). As a baseline, we also report random search (RAND).

Results In Table 4, we present the BO results in 5D, 10D and 20D. We show the average final regret per dimension across the five functions. For each algorithm and dimension, we give the results corresponding to the MW scaling parameter (0.05 or 0.5) that minimizes the average final

[9] These functions are designed for minimization. We multiply them by -1 and equivalently maximize instead.

[10] We fix MW instead of c, since the scales of the algorithms vary by orders of magnitude.

Table 4. BO results: average final regrets per dimension and ranks for each individual function (1=best to 7=worst).

FUNCTION	NOMU	GP	MCDO	DE	HDE	PGP	RAND
LEVY5D	1	1	6	3	3	4	7
ROSENBROCK5D	1	1	1	1	2	5	7
G-FUNCTION5D	2	3	1	4	2	3	7
PERM5D	3	1	1	5	7	2	4
BNN5D	1	1	4	1	4	1	7
Average Regret 5D	2.87e−2	5.03e−2	4.70e−2	5.18e−2	7.13e−2	4.14e−2	1.93e−1
LEVY10D	1	3	5	6	1	1	6
ROSENBROCK10D	1	1	2	6	3	2	7
G-FUNCTION10D	2	5	1	3	2	5	7
PERM10D	2	1	2	6	2	2	1
BNN10D	1	2	1	1	3	1	7
Average Regret 10D	8.40e−2	1.17e−1	6.96e−2	1.15e−1	9.32e−2	9.46e−2	2.35e−1
LEVY20D	1	1	5	7	1	1	6
ROSENBROCK20D	2	2	2	6	1	4	6
G-FUNCTION20D	1	4	5	1	1	3	7
PERM20D	3	5	3	2	3	3	1
BNN20D	1	2	2	2	6	1	7
Average Regret 20D	1.12e−1	1.33e−1	1.39e−1	1.71e−1	1.37e−1	1.17e−1	2.80e−1

regret across that dimension (see Appendix B.3.3 for both MWs). In practice, one often only knows the dimensionality of a given BO task, which is why we use the average final regret per dimension as the criterion for setting the optimal MW. For each individual function, we also present the ranks based on the final regret and a 95% CI over 100 (5D) and 50 (10-20D) runs. We see that NOMU performs as well or better than all benchmarks in terms of average final regret. By inspecting the ranks achieved for each individual function, we further observe that NOMU is never ranked worse than 3rd. In contrast, the performance of the benchmarks heavily depends on the test function; and each benchmark is ranked 4th and worse multiple times. For Perm10D/20D we see that RAND performs best. However, due to a flat optimum of Perm, all algorithms achieve similar (very small) final regrets. Finally, we see that NOMU is always ranked first for the BNN test functions. Figure 6 shows the regret plot for BNN20D (see Appendix B.3.4 for all regret plots).

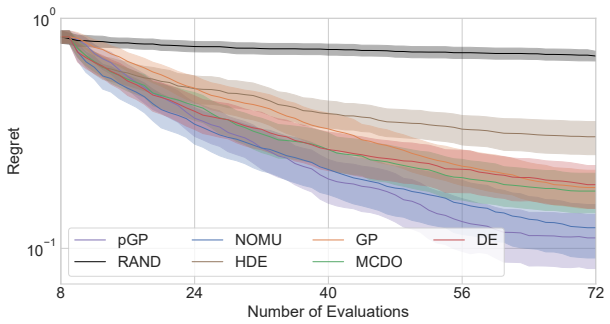


Figure 6. Regret plot for BNN20D. For each BO step, we show the regrets averaged over 50 runs (solid lines) with 95% CIs.

5. Conclusion

We have introduced NOMU, a new algorithm for estimating model uncertainty for NNs, specialized for scarce and noiseless settings. By using a specific architecture and carefully-designed loss function, we have ensured that NOMU satisfies five important desiderata regarding model uncertainty that any method should satisfy. However, when analyzing model uncertainty decoupled from data noise, we have experimentally uncovered that, perhaps surprisingly, established state-of-the-art methods fail to reliably capture some of the desiderata, even those that are required by Bayesian theory. In contrast, NOMU satisfies all desiderata, matches the performance of all benchmarks in regression tasks, and performs as well or better in noiseless BO tasks. We see great potential to further improve NOMU, for example by adapting the loss, or by modifying the connections between the two sub-NNs. We also envision several extensions of NOMU, including its application to classification, employing different architectures (CNNs, GNNs, RNNs or Transformers), and incorporating data noise.

Acknowledgements

We thank Marius Högger and Aurelio Dolfini for insightful discussions and their excellent research assistance in implementing the Bayesian optimization experiments and the UCI data set experiments, respectively. Furthermore, we thank the anonymous reviewers for helpful comments. This paper is part of a project that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No. 805542).

References

Ashukha, A., Lyzhov, A., Molchanov, D., and Vetrov, D. Pitfalls of in-domain uncertainty estimation and ensembling in deep learning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BJxI5gHKDr. 2

Baptista, R. and Poloczek, M. Bayesian optimization of combinatorial structures. In *International Conference on Machine Learning*, pp. 462–471. PMLR, 2018. URL http://proceedings.mlr.press/v80/baptista18a/baptista18a.pdf. 4

Bergstra, J. and Bengio, Y. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012. URL https://www.jmlr.org/papers/v13/bergstra12a.html. 18

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. Weight uncertainty in neural networks. In *32nd International Conference on Machine Learning (ICML)*, 2015. URL http://proceedings.mlr.press/v37/blundell15.pdf. 2, 3, 38

Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., and Vandergheynst, P. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4): 18–42, 2017. URL https://arxiv.org/abs/1611.08097. 40, 41

Caruana, R., Niculescu-Mizil, A., Crew, G., and Ksikes, A. Ensemble selection from libraries of models. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 18, 2004. URL https://www.cs.cornell.edu/~caruana/ctp/ct.papers/caruana.icml04.icdm06long.pdf. 18

Cayton, L. Algorithms for manifold learning. *Univ. of California at San Diego Tech. Rep*, 12(1-17):1, 2005. URL https://www.lcayton.com/resexam.pdf. 5, 26

Foong, A. Y., Li, Y., Hernández-Lobato, J. M., and Turner, R. E. "In-between" uncertainty in bayesian neural networks. *arXiv preprint arXiv:1906.11537*, 2019. URL https://arxiv.org/abs/1906.11537. 8, 29, 30

Fort, S., Hu, H., and Lakshminarayanan, B. Deep ensembles: A loss landscape perspective. *arXiv preprint arXiv:1912.02757*, 2019. URL https://arxiv.org/abs/1912.02757. 2

Gal, Y. and Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *33rd International Conference on Machine Learning (ICML)*, pp. 1050–1059, 2016. URL https://proceedings.mlr.press/v48/gal16.html. 2, 7, 8, 20, 27, 29, 38

Garnelo, M., Rosenbaum, D., Maddison, C., Ramalho, T., Saxton, D., Shanahan, M., Teh, Y. W., Rezende, D., and Eslami, S. A. Conditional neural processes. In *International Conference on Machine Learning*, pp. 1704–1713. PMLR, 2018a. URL https://proceedings.mlr.press/v80/garnelo18a.html. 3, 46

Garnelo, M., Schwarz, J., Rosenbaum, D., Viola, F., Rezende, D. J., Eslami, S., and Teh, Y. W. Neural processes. *arXiv preprint arXiv:1807.01622*, 2018b. URL https://arxiv.org/abs/1807.01622. 3, 46

Ghahramani, Z. Probabilistic machine learning and artificial intelligence. *Nature*, 521(7553):452–459, 2015. URL https://www.nature.com/articles/nature14541. 1

Gómez-Bombarelli, R., Wei, J. N., Duvenaud, D., Hernández-Lobato, J. M., Sánchez-Lengeling, B., Sheberla, D., Aguilera-Iparraguirre, J., Hirzel, T. D., Adams, R. P., and Aspuru-Guzik, A. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018. URL https://doi.org/10.1021/acscentsci.7b00572. 1

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014. URL https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf. 5

Graves, A. Practical variational inference for neural networks. In *Advances in neural information processing systems*, pp. 2348–2356, 2011. URL http://papers.nips.cc/paper/4329-practical-variational-inference-for-neural-networks.pdf. 2, 38

Gustafsson, F. K., Danelljan, M., and Schon, T. B. Evaluating scalable bayesian deep learning methods for robust computer vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 318–319, 2020. URL https://arxiv.org/abs/1906.01620. 2

Havasi, M., Jenatton, R., Fort, S., Liu, J. Z., Snoek, J., Lakshminarayanan, B., Dai, A. M., and Tran, D. Training independent subnetworks for robust prediction. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=OGg9XnKxFAH. 3

Heiss, J., Teichmann, J., and Wutte, H. How implicit regularization of relu neural networks characterizes the learned function – part i: the 1-d case of two layers with random first layer. *arXiv preprint arXiv:1911.02903*, 2019. URL https://doi.org/10.3929/ethz-b-000402003. 15

Heiss, J., Teichmann, J., and Wutte, H. How infinitely wide neural networks can benefit from multi-task learning - an exact macroscopic characterization. *arXiv preprint arXiv:2112.15577*, 2022. doi: 10.3929/ETHZ-B-000550890. URL https://arxiv.org/abs/2112.15577. 14, 43

Hernández-Lobato, J. M. and Adams, R. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning*, pp. 1861–1869, 2015. URL http://proceedings.mlr.press/v37/hernandez-lobatoc15.pdf. 2, 8, 27, 29, 38

Kendall, A. and Gal, Y. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in neural information processing systems*, pp. 5574–5584, 2017. URL https://papers.nips.cc/paper/2017/hash/2650d6089a6d640c5e85b2b88265dc2b-Abstract.html. 2, 44

Khosravi, A., Nahavandi, S., Creighton, D., and Atiya, A. F. Lower upper bound estimation method for construction of neural network-based prediction intervals. *IEEE transactions on neural networks*, 22(3):337–346, 2010. URL https://doi.org/10.1109/TNN.2010.2096824. 2, 6

Kuleshov, V., Fenner, N., and Ermon, S. Accurate uncertainties for deep learning using calibrated regression. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2796–2804, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/kuleshov18a.html. 4, 6, 14, 19

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in neural information processing systems*, pp. 6402–6413, 2017. URL http://papers.nips.cc/paper/7219-simple-and-scalable-predictive-uncertainty-estimation-using-deep-ensembles.pdf. 2, 8, 17, 20, 29

LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *Nature*, 521(7553):436–444, 2015. ISSN 1476-4687. doi: 10.1038/nature14539. URL https://www.nature.com/articles/nature14539. 1

Malinin, A. and Gales, M. Predictive uncertainty estimation via prior networks. In *Advances in Neural Information Processing Systems*, pp. 7047–7058, 2018. URL https://papers.nips.cc/paper/2018/file/3ea2db50e62ceefceaf70a9d9a56a6f4-Paper.pdf. 3, 36, 44

Malinin, A., Chervontsev, S., Provilkov, I., and Gales, M. Regression prior networks, 2020a. URL https://arxiv.org/pdf/2006.11590.pdf. 3, 45

Malinin, A., Mlodozeniec, B., and Gales, M. Ensemble distribution distillation. In *International Conference on Learning Representations*, 2020b. URL https://openreview.net/forum?id=BygSP6Vtvr. 5

Martinez-Cantin, R., De Freitas, N., Brochu, E., Castellanos, J., and Doucet, A. A bayesian exploration-exploitation approach for optimal online sensing and planning with a visually guided mobile robot. *Autonomous Robots*, 27(2): 93–103, 2009. URL https://doi.org/10.1007/s10514-009-9130-2. 1

Mukhoti, J., Stenetorp, P., and Gal, Y. On the importance of strong baselines in bayesian deep learning. *arXiv preprint arXiv:1811.09385*, 2018. URL https://arxiv.org/abs/1811.09385. 8, 29

Neal, R. M. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012. URL https://api.semanticscholar.org/CorpusID:60809283. 1

Nix, D. A. and Weigend, A. S. Estimating the mean and variance of the target probability distribution. In *Proceedings of 1994 ieee international conference on neural networks (ICNN'94)*, volume 1, pp. 55–60. IEEE, 1994. URL https://doi.org/10.1109/ICNN.1994.374138. 2, 46

Ober, S. W. and Rasmussen, C. E. Benchmarking the neural linear model for regression. *arXiv preprint arXiv:1912.08416*, 2019. URL https://arxiv.org/abs/1912.08416. 3, 8, 29

Osband, I., Wen, Z., Asghari, M., Ibrahimi, M., Lu, X., and Van Roy, B. Epistemic neural networks. *arXiv preprint arXiv:2107.08924*, 2021. URL https://arxiv.org/abs/2107.08924. 7, 26, 27

Ovadia, Y., Fertig, E., Ren, J., Nado, Z., Sculley, D., Nowozin, S., Dillon, J., Lakshminarayanan, B., and Snoek, J. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/8558cb408c1d76621371888657d2eb1d-Paper.pdf. 2

Pearce, T., Zaki, M., Brintrup, A., and Neely, A. High-quality prediction intervals for deep learning: A distribution-free, ensembled approach. In *35th International Conference on Machine Learning (ICML)*, pp. 4075–4084, 2018. URL https://proceedings.mlr.press/v80/pearce18a.html. 2, 6

Srinivas, N., Krause, A., Kakade, S. M., and Seeger, M. W. Information-theoretic regret bounds for gaussian process optimization in the bandit setting. *IEEE Transactions on Information Theory*, 58(5):3250–3265, May 2012. ISSN 1557-9654. doi: 10.1109/tit.2011.2182033. URL http://dx.doi.org/10.1109/TIT.2011.2182033. 1

Steinhilber, F., Beer, J., and Fröhlich, C. Total solar irradiance during the holocene. *Geophysical Research Letters*, 36(19), 2009. doi: 10.1029/2009GL040142. URL https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2009GL040142. 7

Wahba, G. Improper priors, spline smoothing and the problem of guarding against model errors in regression. *Journal of the Royal Statistical Society: Series B (Methodological)*, 40(3):364–372, 1978. URL https://doi.org/10.1111/j.2517-6161.1978.tb01050.x. 36

Wang, Z., Ren, T., Zhu, J., and Zhang, B. Function space particle optimization for bayesian neural networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=BkgtDsCcKQ. 3

Weissteiner, J. and Seuken, S. Deep learning—powered iterative combinatorial auctions. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(02):2284–2293, Apr. 2020. doi: 10.1609/aaai.v34i02.5606. URL https://ojs.aaai.org/index.php/AAAI/article/view/5606. 4

Weissteiner, J., Heiss, J., Siems, J., and Seuken, S. Bayesian optimization-based combinatorial assignment. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37, 2023. URL https://arxiv.org/abs/2208.14698. 36

Wen, Y., Tran, D., and Ba, J. Batchensemble: An alternative approach to efficient ensemble and lifelong learning, 2020. URL https://arxiv.org/pdf/2002.06715.pdf. 3

Wenzel, F., Roth, K., Veeling, B., Swiatkowski, J., Tran, L., Mandt, S., Snoek, J., Salimans, T., Jenatton, R., and Nowozin, S. How good is the Bayes posterior in deep neural networks really? In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 10248–10259. PMLR, 13–18 Jul 2020a. URL https://proceedings.mlr.press/v119/wenzel20a.html. 2, 18

Wenzel, F., Snoek, J., Tran, D., and Jenatton, R. Hyperparameter ensembles for robustness and uncertainty quantification. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020b. Curran Associates Inc. ISBN 9781713829546. URL https://papers.nips.cc/paper/2020/file/481fbfa59da2581098e841b7afc122f1-Paper.pdf. 2, 18, 20, 21

Williams, C. K. and Rasmussen, C. E. *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006. URL https://gaussianprocess.org/gpml/. 16, 37

Appendix

A. Theoretical Analysis of NOMU

In this section, we

1. first provide a theoretical motivation for the design of NOMU and establish via Theorem A.1 a connection to pointwise worst-case UBs $\underline{UB}_{\text{pw}}, \overline{UB}_{\text{pw}}$ **(Appendix A.1)**.
2. Next, we provide a Bayesian interpretation of those pointwise worst-case UBs $\underline{UB}_{\text{pw}}, \overline{UB}_{\text{pw}}$ and elaborate on *relative uncertainties* **(Appendix A.2)**.
3. Then, we discuss the case if $\mathcal{H}_{D^{\text{train}}}$ is not *upwards directed* as assumed in Theorem A.1 and further show how we deal with this challenge **(Appendix A.3)**.
4. Finally, we prove Theorem A.1 **(Appendix A.4)**.

A.1. Relating NOMU to Pointwise Worst-Case Uncertainty Bounds

In this section, we provide a theoretical motivation for the design of NOMU. To this end, we first define worst-case UBs. We then state a theorem connecting these worst case bounds and NOMU's UBs via NOMU's loss function. In what follows, we assume zero data noise $\sigma_{\text{n}} = 0$.

Consider a *hypothesis class* $\mathcal{H}$ given as a subset of a Banach space of functions $f : X \to Y$. Furthermore, let $\mathcal{H}_{D^{\text{train}}} := \{f \in \mathcal{H} : f(x_i^{\text{train}}) = y_i^{\text{train}}, i = 1, \dots, n^{\text{train}}\}$ denote the set of all functions from the hypothesis class that fit through the training points and let $\hat{f} \in \mathcal{H}_{D^{\text{train}}}$ be a prediction function (e.g., a NN trained on D^{train}). Worst-case bounds within the class $\mathcal{H}_{D^{\text{train}}}$ can be defined **p**oint**w**ise for each $x \in X$ as:

$$\underline{UB}_{\text{pw}}(x) := \inf_{f \in \mathcal{H}_{D^{\text{train}}}} f(x), \tag{5}$$

$$\overline{UB}_{\text{pw}}(x) := \sup_{f \in \mathcal{H}_{D^{\text{train}}}} f(x). \tag{6}$$

By definition, these UBs are the tightest possible bounds that cover every $f \in \mathcal{H}_{D^{\text{train}}}$ (i.e., $\underline{UB}_{\text{pw}}(x) \leq f(x) \leq \overline{UB}_{\text{pw}}(x)\ \forall x \in X$). From a Bayesian perspective, such bounds correspond to credible bounds for $\alpha = 1$ if the support of the prior is contained in $\mathcal{H}$. Interestingly, if $\mathcal{H}$ is the class of regularized NNs, these bounds can also be interpreted as an approximation of credible bounds for $\alpha < 1$ with respect to a Gaussian prior on the parameters of a NN (see Appendix A.2. for a derivation).[11]

In applications like BO, when optimizing an acquisition function based on these pointwise-defined bounds, we require the UBs for *all* $x \in X$. Thus, numerically optimizing such an acquisition function is practically infeasible, as it would require solving the optimization problems from (5) millions of times. In NOMU, we circumvent this problem by constructing the UBs for *all* $x \in X$ simultaneously. We can do so by only solving a *single* optimization problem, i.e., minimizing the NOMU loss from (4).

In the following theorem, we show that these pointwise-defined UBs can be computed by solving a single optimization problem under the following assumption.

Assumption 1 (UPWARDS DIRECTED) *For every $f_1, f_2 \in \mathcal{H}_{D^{\text{train}}}$ there exists an $f \in \mathcal{H}_{D^{\text{train}}}$ such that $f(x) \geq \max(f_1(x), f_2(x))$ for all $x \in X$.*

Theorem A.1 (SINGLE OPTIMIZATION PROBLEM) *Let $X = \prod_{i=1}^{d}[a_i, b_i] \subset \mathbb{R}^d$, $a_i < b_i$, let $Y = \mathbb{R}$, and let D^{train} be a nonempty set of training points. Furthermore, let $\mathcal{H}_{D^{\text{train}}} \subset (C(X, Y), \|\cdot\|_\infty)$ be compact and* upwards directed *and $\hat{f} \in \mathcal{H}_{D^{\text{train}}}$. Then, for every strictly-increasing and continuous $u : \mathbb{R} \to \mathbb{R}$, it holds that*

$$\overline{UB}_{\text{pw}} = \operatorname*{arg\,max}_{h \in \mathcal{H}_{D^{\text{train}}}} \int_X u(h(x) - \hat{f}(x))\, dx. \tag{7}$$

In words, $\overline{UB}_{\text{pw}}$ can be calculated via the single optimization problem (7) on $\mathcal{H}_{D^{\text{train}}}$.[12]

Proof. See Appendix A.4. □

In practice, Assumption 1 can be violated such that a straightforward calculation of the r.h.s. of (7) for an *arbitrary* u would result in unreasonable UBs. However, for a sensible choice of u, NOMU's UBs based on the r.h.s. of (7) still satisfy our Desiderata D1–D5, similar to $\overline{UB}_{\text{pw}}$ (see Appendix A.3 for a discussion).

The connection of NOMU's UBs to the pointwise worst-case bounds $\underline{UB}_{\text{pw}}, \overline{UB}_{\text{pw}}$ becomes clear by observing that minimizing NOMU's loss function L^π (Equation (4)) can be interpreted as solving the r.h.s of (7) for a specific choice of u, when $\mathcal{H}$ is the class of regularized NNs. In detail:

- Term (a) of the NOMU-loss (4) implements that $\hat{f}$ solves the regression task and thus $\hat{f} \in \mathcal{H}_{D^{\text{train}}}$ up to numerical precision (if the regularization λ is small enough).
- Term (b) enforces $\hat{r}_f(x_i^{\text{train}}) \approx 0$ and thus when defining $h := \hat{f} + \hat{r}_f$, we directly obtain $h(x_i^{\text{train}}) \approx y_i^{\text{train}}$ corresponding to the constraint $h \in \mathcal{H}_{D^{\text{train}}}$ in (7).
- While terms (a) and (b) enforce the constraints of (7), term (c) is the objective function of (7) for the specific choice of $u(z) := -e^{-c_{\exp} z}, c_{\exp} \in \mathbb{R}_{\geq 0}$.

[11] Standard BNNs also aim to approximate the posterior coming from exactly this prior.

[12] We formulate Theorem A.1 for upper UBs $\overline{UB}_{\text{pw}}$. The analogous statement also holds for lower UBs $\underline{UB}_{\text{pw}}$.

A.2. Bayesian Interpretation of Pointwise Worst-Case Uncertainty Bounds

In this section, we provide a Bayesian interpretation of the pointwise worst-case UBs $\underline{UB}_{\text{pw}}, \overline{UB}_{\text{pw}}$ and elaborate on *relative uncertainties*.

In the following, we denote by $\mathcal{NN}_\theta^f : X \to Y$ a (standard) NN for model predictions. Note that $\mathcal{NN}_\theta^f$ does not represent the whole NOMU architecture but can be used as $\hat{f}$-sub-network in the NOMU architecture $\mathcal{NN}_\theta$. Furthermore, we consider the hypothesis class of regularized NNs, i.e., $\mathcal{H} := \left\{\mathcal{NN}_\theta^f : \|\theta\|_2 \leq \gamma\right\}$. Recall that one needs to assume that the prior is fully concentrated on $\mathcal{H}$ in order to interpret the pointwise UBs $\underline{UB}_{\text{pw}}, \overline{UB}_{\text{pw}}$ as $\alpha{=}1$ CBs. In the following, we will present an alternative Bayesian interpretation of $\overline{UB}_{\text{pw}}$.

Many other approaches (MC dropout, BNNs) assume a Gaussian prior on the parameters of the NNs, i.e, $\theta \sim \mathcal{N}(0, \sigma_\theta^2 I)$, and try to approximate the corresponding posterior CBs. Interestingly, $\underline{UB}_{\text{pw}}$ and $\overline{UB}_{\text{pw}}$ can also be seen as approximations of $\alpha{<}1$ CBs in the case of such a Gaussian prior on the parameters. This can be seen as follows:

Let the data generating process be given as $y = \mathcal{NN}_\theta^f + \varepsilon$, with $\varepsilon \sim \mathcal{N}(0, \sigma_{\text{n}}^2)$.[13] For a Gaussian prior on the parameters $\theta \sim \mathcal{N}(0, \sigma_\theta^2 I)$ the negative log posterior can be written as

$$-\log(p(\theta|D^{\text{train}})) = \frac{1}{2\sigma_{\text{n}}^2} \overbrace{\sum_{i=1}^{n^{\text{train}}} \left(\mathcal{NN}_\theta^f(x_i^{\text{train}}) - y_i^{\text{train}}\right)^2}^{=:L(\theta)} + \frac{\|\theta\|_2^2}{2\sigma_\theta^2} + n^{\text{train}} \log(\sigma_{\text{n}}) + C_{n^{\text{train}},\sigma_\theta}. \tag{8}$$

for a constant $C_{n^{\text{train}},\sigma_\theta} := \frac{n^{\text{train}}}{2}\log(2\pi) + \frac{1}{2}\log(2\pi\sigma_\theta^2)$. Then the pointwise upper UBs can be reformulated to

$$\overline{UB}_{\text{pw}}(x) \overset{\text{def.}}{=} \sup_{f \in \mathcal{H}_{D^{\text{train}}}} f(x) = \lim_{\sigma_{\text{n}} \to 0} \sup_{f \in \tilde{\mathcal{H}}_{D^{\text{train}}}^{\sigma_{\text{n}}}} f(x) \tag{9}$$

with

$$\tilde{\mathcal{H}}_{D^{\text{train}}}^{\sigma_{\text{n}}} := \left\{\mathcal{NN}_\theta^f : \frac{\sigma_\theta^2}{\sigma_{\text{n}}^2} L(\theta) + \|\theta\|_2^2 \leq \gamma\right\} \tag{10a}$$

$$= \left\{\mathcal{NN}_\theta^f : \log(p(\theta|D^{\text{train}})) \geq \tilde{\gamma}_{\sigma_{\text{n}}}\right\} \tag{10b}$$

where $\tilde{\gamma}_{\sigma_{\text{n}}} := -\frac{\gamma}{2\sigma_\theta^2} - n^{\text{train}}\log(\sigma_{\text{n}}) - C_{n^{\text{train}},\sigma_\theta}$.

Therefore, for small data noise $\sigma_{\text{n}} \approx 0$ we obtain from Equations (9) and (10b) that

$$\overline{UB}_{\text{pw}}(x) \approx \sup_\theta \left(\mathcal{NN}_\theta^f(x) : p(\theta|D^{\text{train}}) > e^{\tilde{\gamma}_{\sigma_{\text{n}}}}\right). \tag{11}$$

In words, from a Bayesian point of view, we seek the highest value $\mathcal{NN}_\theta^f(x)$ for which the posterior density $p(\theta|D^{\text{train}}) > e^{\tilde{\gamma}_{\sigma_{\text{n}}}}$, which can be seen as a heuristic to approximate CBs analogously to the MAP on the parameter-space as a popular heuristic to approximate the posterior mean.

Remark A.2 *If $p(\mathcal{NN}_\theta^f(x)|D^{\text{train}}) = p(\theta|D^{\text{train}})$ and this posterior is unimodal, Equation (11) is a classical* highest posterior density interval (HPDI). *However, typically $p(\mathcal{NN}_\theta^f(x)|D^{\text{train}}) \neq p(\theta|D^{\text{train}})$. Thus, NOMU is only a heuristic to approximate Gaussian BNN posterior CBs (analogously to the MAP). Heiss et al. (2022) show that such heuristics can perform better than exact BNNs (e.g., in contrast to BNNs, NOMU fulfills D4 also for infinite-width).*

A.2.1. From Absolute to Relative Model Uncertainty

If the prior scale (e.g., γ or σ_θ in the above mentioned approaches) is known, in theory no further calibration is needed and one can interpret the resulting UBs in *absolute* terms (Kuleshov et al., 2018). However, typically, the prior scale is unknown and the resulting UBs can only be interpreted in terms of *relative* model uncertainty (i.e., how much more model uncertainty does one have at one point x compared to any other point x'?) and in a second step careful calibration of the resulting UBs is required.

Thus, there are two (almost) independent problems: First, the fundamental concept of how to estimate relative model uncertainty (such as MC dropout, deep ensembles, hyper deep ensembles or NOMU) and second, the calibration of these UBs. In this paper, we do not mix up these two challenges and only focus on the first one. Furthermore, desiderata D1–D4 and our metrics AUC and NLL_{min} do not depend on the scaling of the uncertainty. Whenever we use NLL as a metric, we make sure to calibrate the uncertainties by a scalar c to ensure that our evaluations do not depend on the scaling of the uncertainty predictions.

A.3. A Note on Theorem A.1

In practice, the set $\mathcal{H}_{D^{\text{train}}}$ often is not upwards directed for typical NN-architectures and Equation (7) of Theorem A.1 is not fulfilled in general. Indeed, $h^* := \arg\max_{h \in \mathcal{H}_{D^{\text{train}}}} \int_X u(h(x) - \hat{f}(x))\, d(x)$ can be much more overconfident than $\overline{UB}_{\text{pw}}$. However, in the following we will motivate why for a suitably chosen u the relative model uncertainty of h^* still fulfills the desiderata D1 (Non-Negativity), D2 (In-Sample) and D3 (Out-of-Sample).

Remark A.3 *In our proposed final algorithm from Section 3 NOMU, we incorporated D4 (Metric Learning) by*

[13] For simplicity we assume homoskedastic noise in this section.

modifying the network architecture as presented in Section 3.2.

Problem First, we give an example of a specific NN-architecture, where Theorem A.1 is not fulfilled due to a violation of the upwards directed assumption. Note that from a Lagrangian perspective it is equivalent to add a term $-\lambda \|\theta\|_2^2, \lambda \geq 0$ to the target function of (7) instead of bounding $\|\theta\|_2^2 \leq \gamma^2$ as a constraint in $\mathcal{H}_{D^{\text{train}}}$. Moreover, for a specific NN-architecture it is shown that regularizing $\|\theta\|_2^2$ is equivalent to regularizing the L_2-norm of the second derivative of the function $f = \mathcal{NN}_\theta^f$ (Heiss et al., 2019), i.e., regularizing $\int_X f''(x)^2 \, dx$. If we choose for example $u = id : \mathbb{R} \to \mathbb{R}$, then increasing h in-between the two close points in the middle of Figure 7, would improve the target function of (7) $\int u(h(x) - \hat{f}(x)) \, dx$ less than the additional regularization cost resulting from the additional second derivative when increasing h. Therefore, h^* will be below the mean prediction $\hat{f}$ in this region, break D1 (Non-Negativity), and $h^* \neq \overline{UB}_{\text{pw}}$.

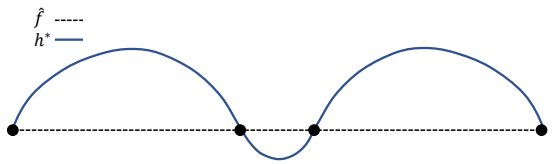


Figure 7. h^* would not fulfill D1 (Non-Negativity), if u was chosen as the identity and the architecture from Heiss et al. (2019) was used.

Solution However, if e.g., we choose $u : x \mapsto -e^{-c_{\text{exp}} x}$ with c_{exp} large enough, we highly penalize $h < \hat{f}$, and thus $\hat{f} \leq h^*$ D1 (Non-Negativity) is fulfilled. Since $h^* \in \mathcal{H}_{D^{\text{train}}}$, it follows that $h^*(x_i^{\text{train}}) - \hat{f}(x_i^{\text{train}}) = 0$ for all training points. This together with $\hat{f} \leq h^*$ implies that $\nabla \left(h^* - \hat{f}\right)(x_i^{\text{train}}) = 0$ at the training points (in the interior of X), which subsequently interrupts the downwards trend of $h^* - \hat{f}$ from one side of a point to the other side of a point as depicted in Figure 8.

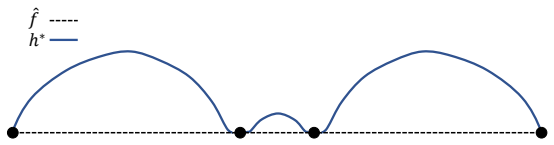


Figure 8. h^* fulfills D1–D3 if u prevents it from getting negative. See Appendix D for a detailed discussion how NOMU fulfills all five desiderata.

A.4. Proof of Theorem A.1

In the following, we prove Theorem A.4, an even more general version of Theorem A.1. The statement of Theorem A.1 follows from Theorem A.4 by setting

1. $X = \prod_{i=1}^d [a_i, b_i] \subset \mathbb{R}^d$ compact, $Y := \mathbb{R}$
2. $\mu = \lambda_d$, where λ_d is the Lebesgue measure on $\mathbb{R}^d$ (which is finite on every compact set and has full support).

Theorem A.4 *Let X be a nonempty topological space equipped with a finite measure μ with full support*[14]*, let Y be a normed and totally ordered space and let D^{train} denote a set of observations. Moreover, let $\mathcal{H}_{D^{\text{train}}} \subset (C(X, Y), \|\cdot\|_\infty)$ be compact, and $\hat{f} \in \mathcal{H}_{D^{\text{train}}}$. Assume further that $\mathcal{H}_{D^{\text{train}}}$ is upwards directed (see Assumption 1). Then, for every strictly-increasing and continuous $u : Y \to \mathbb{R}$ it holds that*

$$\overline{UB}_{\text{pw}} = \underset{h \in \mathcal{H}_{D^{\text{train}}}}{\arg\max} \int_X u(h(x) - \hat{f}(x)) \, d\mu(x). \tag{12}$$

Proof. First note that since μ is finite and $h, \hat{f}$ and $\overline{UB}_{\text{pw}}$ are bounded since $\mathcal{H}_{D^{\text{train}}}$ is assumed to be compact with respect to the $\|\cdot\|_\infty$-topology, it holds that the integral in (12) is finite.

Let $h^* \in \mathcal{H}_{D^{\text{train}}}$ denote an optimizer of (12), which exists since $\mathcal{H}_{D^{\text{train}}}$ is assumed to be compact and the operator $h \mapsto \int_X u(h(x) - \hat{f}(x)) \, d\mu(x)$ is continuous on the $\|\cdot\|_\infty$-topology.

We want to show that $h^*(x) = \overline{UB}_{\text{pw}}(x)$ for every $x \in X$.

Note that per definition for all $x \in X$ and $h \in \mathcal{H}_{D^{\text{train}}}$ it holds that

$$\overline{UB}_{\text{pw}}(x) = \sup_{f \in \mathcal{H}_{D^{\text{train}}}} f(x) \geq h(x). \tag{13}$$

Thus $\overline{UB}_{\text{pw}}(x) \geq h^*(x)$ for all $x \in X$.

For the reverse inequality assume on the contrary that there exists an $x' \in X$ such that

$$\overline{UB}_{\text{pw}}(x') > h^*(x'). \tag{14}$$

Then, we define $f_{x'} := \arg\max_{f \in \mathcal{H}_{D^{\text{train}}}} f(x')$, which exists because of compactness and continuity. Since $f_{x'}$ and h^* are both continuous and $f_{x'}(x') = \overline{UB}_{\text{pw}}(x') > h^*(x')$ there exists a neighbourhood $U_{x'}$ of x' such that

$$f_{x'}(x) > h^*(x) \text{ for all } x \in U_{x'}, \tag{15}$$

Using the *upwards directed* property with $f_1 := f_{x'}$ and $f_2 := h^*$, it follows that there exists a $\tilde{h} \in \mathcal{H}_{D^{\text{train}}}$ with

$$\tilde{h}(x) \geq \max(f_{x'}(x), h^*(x)) \text{ for all } x \in X. \tag{16}$$

[14] I.e. every nonempty open set has non-zero measure.

Using (15) together with (16) implies further that

$$\tilde{h}(x) \geq h^*(x) \text{ for all } x \in X \text{ and} \tag{17}$$

$$\tilde{h}(x) > h^*(x) \text{ for all } x \in U_{x'}. \tag{18}$$

However, since u is strictly increasing and $\mu(U_{x'}) > 0$ by the full support assumption, we get that

$$\int_X u(\tilde{h}(x) - \hat{f}(x))\, d\mu(x) > \int_X u(h^*(x) - \hat{f}(x))\, d\mu(x), \tag{19}$$

which is a contradiction to the assumption that h^* is the optimizer of (12). Therefore, it holds that $\overline{UB}_{\text{pw}}(x) \leq h^*(x)$ for all $x \in X$.

In total we get that $\overline{UB}_{\text{pw}}(x) = h^*(x)$ for all $x \in X$, which concludes the proof. □

Remark A.5 *For our algorithm we select* $\mathcal{H} := \left\{\mathcal{NN}_\theta^f : \|\theta\|_2 \leq \gamma\right\}$ *to be the class of regularized NNs with a continuous activation function. Thus, the assumption of* $\mathcal{H}_{D^{\text{train}}}$ *being compact and a subset of* $C(X, Y)$ *is fulfilled.*

B. Experiments

B.1. Benchmark Methods

In this section, we give a brief overview of each considered benchmark algorithm.

B.1.1. Gaussian Process (GP)

A GP defines a distribution over a set of functions $\{f : X \to Y\}$ where every finite collection of function evaluations follows a Gaussian distribution (see Williams & Rasmussen (2006) for a survey on GPs). A GP is completely specified by a *mean* $m : X \to Y$ and a *kernel* function $k_\pi : X \times X \to Y$, where π denotes a tuple of hyper-parameters. More formally, for any finite set of $k \in \mathbb{N}$ input points $\boldsymbol{x} := \{x_1, \ldots, x_k\}$, $x_i \in X$ and for $\boldsymbol{f} := (f_1, \ldots, f_k) \in Y^k$ with $f_i := f(x_i)$ it holds that

$$\boldsymbol{f} \sim \mathcal{N}_k\left(\boldsymbol{m}(\boldsymbol{x}), \boldsymbol{K}_\pi(\boldsymbol{x}, \boldsymbol{x})\right), \tag{20}$$

i.e., it follows a k-dimensional Gaussian distribution with covariance (or Gramian) matrix $[\boldsymbol{K}_\pi(\boldsymbol{x}, \boldsymbol{x})]_{i,j} := k_\pi(x_i, x_j)$, and mean vector $\boldsymbol{m}(\boldsymbol{x}) := (m(x_1), \ldots, m(x_k))$. Let

$$f \sim \mathcal{GP}(m(\cdot), k_\pi(\cdot, \cdot)), \tag{21}$$

denote a GP with mean function m and kernel k_π.

In the following, we summarize the main steps of GP regression for a 1D-setting and $m \equiv 0$.

1. *Define probabilistic model:*

$$y = f(x) + \varepsilon. \tag{22}$$

2. *Specify prior (kernel and data noise):*

$$f \sim \mathcal{GP}(0, k_\pi(\cdot, \cdot)), \tag{23}$$

$$\varepsilon | x \sim \mathcal{N}\left(0, \sigma_{\text{n}}^2(x)\right). \tag{24}$$

3. Calculate *likelihood* for training points $\boldsymbol{x}$ and $\boldsymbol{y} := \{y_1, \ldots, y_k\}$:

$$\boldsymbol{y} | \boldsymbol{x}, \pi \sim \mathcal{N}_k\left(\boldsymbol{0}, \boldsymbol{K}_\pi(\boldsymbol{x}, \boldsymbol{x}) + diag(\sigma_{\text{n}}^2(\boldsymbol{x}))\right), \tag{25}$$

with $\sigma_{\text{n}}^2(\boldsymbol{x}) := \left(\sigma_{\text{n}}^2(x_1), \ldots, \sigma_{\text{n}}^2(x_k)\right)$.

4. *Optimize kernel hyper-parameters (optional):*

$$\hat{\pi} \in \arg\max_\pi p(\boldsymbol{y} | \boldsymbol{x}, \pi). \tag{26}$$

5. Calculate *posterior predictive distribution* for new point (x^*):

$$f(x^*) | x^*, \boldsymbol{y}, \boldsymbol{x}, \hat{\pi} \sim \mathcal{N}\left(\hat{\mu}(x^*), \hat{\sigma}_f^2(x^*)\right), \tag{27}$$

where for $A := \left(\boldsymbol{K}_{\hat{\pi}}(\boldsymbol{x}, \boldsymbol{x}) + diag(\sigma_{\text{n}}^2(\boldsymbol{x}))\right)$ and $\boldsymbol{k}_{\hat{\pi}}(x^*, \boldsymbol{x}) := (k_{\hat{\pi}}(x^*, x_1), \ldots, k_{\hat{\pi}}(x^*, x_k))$ the parameters are given as

$$\hat{\mu}(x^*) := \boldsymbol{k}_{\hat{\pi}}(x^*, \boldsymbol{x}) A^{-1} f(\boldsymbol{x}) \tag{28}$$

$$\hat{\sigma}_f^2(x^*) := k_{\hat{\pi}}(x^*, x^*) - \boldsymbol{k}_{\hat{\pi}}(x^*, \boldsymbol{x}) A^{-1} \boldsymbol{k}_{\hat{\pi}}(x^*, \boldsymbol{x})^T. \tag{29}$$

Setting the data noise to zero $\sigma_{\text{n}} \equiv 0$ in (28) and (29) yields the mean prediction $\hat{f}$ and the model uncertainty prediction $\hat{\sigma}_f^2$ as

$$\hat{f}(x^*) = \boldsymbol{k}_{\hat{\pi}}(x^*, \boldsymbol{x}) \left(\boldsymbol{K}_{\hat{\pi}}(\boldsymbol{x}, \boldsymbol{x})\right)^{-1} f(\boldsymbol{x}), \tag{30}$$

$$\hat{\sigma}_f^2(x^*) = k_{\hat{\pi}}(x^*, x^*) - \boldsymbol{k}_{\hat{\pi}}(x^*, \boldsymbol{x}) \left(\boldsymbol{K}_{\hat{\pi}}(\boldsymbol{x}, \boldsymbol{x})\right)^{-1} \boldsymbol{k}_{\hat{\pi}}(x^*, \boldsymbol{x})^T, \tag{31}$$

which are then used to define the Gaussian process's UBs by $\left(\hat{f}(x) \mp c\, \hat{\sigma}_f(x)\right)$ with a calibration parameter $c \in \mathbb{R}_{\geq 0}$.

B.1.2. Monte Carlo Dropout (MCDO)

Let $\mathcal{NN}_\theta^f = W^K \circ \phi \circ W^{K-1} \circ \ldots \circ \phi \circ W^1(x)$ be an NN with $K - 1$ hidden layers, activation function ϕ, and fixed parameters $\theta = \{W^1, \ldots, W^K\}$, which have been trained with added dropout regularization. Furthermore, let $(p_1, \ldots, p_K)$ denote the dropout probability vector used when training $\mathcal{NN}_\theta^f$, i.e, p_i determines the probability for a single node in the ith hidden layer W^i to be dropped in each backpropagation step.[15]

To obtain model uncertainty one draws M different NNs according to the dropout probability vector and represents model uncertainty using sample estimates of the mean and

[15] One could also use different probabilities p_{ij} for each node within a hidden layer. The equations extend straightforwardly.

variance of the model predictions.[16] These predictions are frequently termed *stochastic forward passes*. More formally, given a dropout probability vector $(p_1, \dots, p_K)$, one draws M realisations $\{\theta^{(1)}, \dots, \theta^{(M)}\}$ of parameters θ, where $\theta^{(m)} := \{W^{1,(m)}, \dots, W^{K,(m)}\}$. $W^{k,(m)}$ is obtained from the original hidden layer W^k by dropping each column with probability p_i, i.e., for $W^k \in \mathbb{R}^{d_{row} \times d_{col}}$ set

$$W^{k,(m)} = W^k \left[z_1^{(m)}, \dots, z_{d_{col}}^{(m)}\right], \; z_j^{(m)} \in \mathbb{R}^{d_{row}} \tag{32}$$

$$\text{where } z_j^{(m)} := \begin{cases} \mathbf{0}, & \text{with probability } p_i \\ \mathbf{1}, & \text{with probability } 1 - p_i. \end{cases} \tag{33}$$

UBs that represent model uncertainty and *known* data noise σ_{n}^2 are then estimated for each $x \in X$ as

$$\hat{f}(x) := \frac{1}{M} \sum_{m=1}^{M} \mathcal{NN}^f_{\theta^{(m)}}(x), \tag{34}$$

$$\hat{\sigma}^2(x) := \underbrace{\frac{1}{M} \sum_{m=1}^{M} \left(\mathcal{NN}^f_{\theta^{(m)}}(x) - \hat{f}(x)\right)^2}_{\text{model uncertainty}} + \underbrace{\sigma_{\text{n}}^2(x)}_{\text{data noise}}. \tag{35}$$

The model uncertainty prediction $\hat{\sigma}_f^2$ is then given as

$$\hat{\sigma}_f^2(x) := \frac{1}{M} \sum_{m=1}^{M} \left(\mathcal{NN}^f_{\theta^{(m)}}(x) - \hat{f}(x)\right)^2 \tag{36}$$

which defines Mc dropout's UBs as $\left(\hat{f}(x) \mp c\, \hat{\sigma}_f(x)\right)$ with a calibration parameter $c \in \mathbb{R}_{\geq 0}$.

B.1.3. Deep Ensembles (DE)

Deep ensembles consists of the following two steps:[17]

1. Use a NN to define a predictive distribution $p_\theta(y|x)$, select a pointwise loss function (proper scoring rule) $\ell(p_\theta, (x, y))$, which measures the quality of the predictive distribution $p_\theta(y|x)$ for an observation (x, y) and define the empirical loss used for training as

$$L(\theta) := \sum_{(x,y) \in D^{\text{train}}} \ell(p_\theta, (x, y)). \tag{37}$$

2. Use an ensemble of NNs, each with different randomly initialized parameters to represent model uncertainty.

Concretely, for regression, Lakshminarayanan et al. (2017) use a NN $\mathcal{NN}^f_\theta$ with two outputs: $\hat{\mu}^\theta$ (mean prediction) and $\hat{\sigma}_{\text{n}}^\theta$ (data noise prediction) and train it using as pointwise loss function the Gaussian negative log-likelihood, i.e,

$$p_\theta(y|x) := \mathcal{N}\left(y; \hat{\mu}^\theta(x), \left(\hat{\sigma}_{\text{n}}^\theta(x)\right)^2\right), \tag{38}$$

$$\ell(p_\theta, (x, y)) := \frac{\log\left(\left(\hat{\sigma}_{\text{n}}^\theta(x)\right)^2\right)}{2} + \frac{(\hat{\mu}^\theta(x) - y)^2}{2\left(\hat{\sigma}_{\text{n}}^\theta(x)\right)^2}. \tag{39}$$

To add model uncertainty, Lakshminarayanan et al. (2017) use an ensemble of M NNs $\{\mathcal{NN}^f_{\theta^{(1)}}, \dots, \mathcal{NN}^f_{\theta^{(M)}}\}$, where each NN outputs a mean and data noise prediction, i.e, for $x \in X$ and $m \in \{1, \dots, M\}$

$$\mathcal{NN}^f_{\theta^{(m)}}(x) := \left(\hat{\mu}^{\theta^{(m)}}(x), \hat{\sigma}_{\text{n}}^{\theta^{(m)}}(x)\right). \tag{40}$$

This then defines the learned predictive distribution for each NN in regression as

$$p_{\theta^{(m)}}(y|x) = \mathcal{N}\left(y; \hat{\mu}^{\theta^{(m)}}(x), \left(\hat{\sigma}_{\text{n}}^{\theta^{(m)}}(x)\right)^2\right). \tag{41}$$

Finally, the ensemble is treated as uniformly-weighted Gaussian mixture model, i.e.,

$$\frac{1}{M} \sum_{m=1}^{M} p_{\theta^{(m)}}(y|x), \tag{42}$$

which is further approximated using a single Gaussian by matching the first and second moments. The deep ensemble predictions for the mean $\hat{f}$ and for the combined model uncertainty and data noise $\hat{\sigma}^2$ are given as

$$\hat{f}(x) := \frac{1}{M} \sum_{m=1}^{M} \hat{\mu}^{\theta^{(m)}}(x), \tag{43}$$

$$\hat{\sigma}^2(x) := \underbrace{\frac{1}{M} \sum_{m=1}^{M} \left(\hat{\sigma}_{\text{n}}^{\theta^{(m)}}(x)\right)^2}_{\text{data noise}} + \underbrace{\frac{1}{M} \sum_{m=1}^{M} \left[\hat{\mu}^{\theta^{(m)}}(x) - \hat{f}(x)\right]^2}_{\text{model uncertainty}}. \tag{44}$$

The model uncertainty prediction $\hat{\sigma}_f^2$ is then given as

$$\hat{\sigma}_f^2(x) := \frac{1}{M} \sum_{m=1}^{M} \left[\hat{\mu}^{\theta^{(m)}}(x) - \hat{f}(x)\right]^2. \tag{45}$$

which defines deep ensembles' UBs as $\left(\hat{f}(x) \mp c\, \hat{\sigma}_f(x)\right)$ with a calibration parameter $c \in \mathbb{R}_{\geq 0}$.

Remark B.1 *For known data noise σ_n, no estimation is required and one can use an NN $\mathcal{NN}^f_\theta$ with only one output $\hat{\mu}^\theta$. If additionally the data noise σ_n is assumed to be homoskedastic, one can train $\mathcal{NN}^f_\theta$ using the mean squared error (MSE) with suitably chosen L2-regularization parameter λ. To obtain predictive bounds instead of credible bounds, one can add σ_n^2 to (45) at the end.*

[16] Alternatively, one could also determine the UBs using empirical upper and lower quantiles of the different model predictions.

[17] Lakshminarayanan et al. (2017) also considered in their paper a third step: *adversarial training*. However, as the authors point out, the effectiveness of adversarial training drops quickly as the number of networks in the ensemble increases. Therefore, we do not consider adversarial training in this paper.

B.1.4. Hyper Deep Ensembles (HDE)

Hyper deep ensembles (HDE) (Wenzel et al., 2020b) is a simple extension of DE. In HDE, the ensemble is designed by varying not only weight initializations, but also hyperparameters. HDE involves (i) a random search over different hyperparameters and (ii) stratification across different random initializations. HDE inherits all the components (e.g., the architecture, or the loss function) from DE, which are presented in detail in Appendix B.1.3. Specifically, formulas (43) and (44) for the mean prediction $\hat{f}$ and the model uncertainty prediction $\hat{\sigma}^2_f(x)$ are the same for HDE.

Let $\mathcal{NN}^f_{\theta^{(m)},\pi^{(m)}}$ denote a NN for model predictions with weights $\theta^{(m)}$ and hyperparameters $\pi^{(m)}$ (e.g., the dropout rate). Furthermore, let $\texttt{rand\_search}(\kappa)$ denote the random search algorithm from (Bergstra & Bengio, 2012), where κ is the desired number of different NNs with randomly sampled hyperparameters.

The only difference of HDE compared to DE is the procedure how the ensemble is built, which we reprint in Algorithm 2. Algorithm 2 uses as subprocedure Algorithm 1 from (Caruana et al., 2004), which we present first.

Algorithm 1 greedily grows an ensemble among a given pre-defined set of models $\mathcal{M}$, until some target size M is met, by selecting *with-replacement* the NN leading to the best improvement of a certain score $\mathcal{S}$ on a validation set.[18]

Algorithm 1 `hyper_ens` (Caruana et al., 2004)

input $\mathcal{M}$, M
  Ensemble $\mathcal{E} := \{\}$; Score $\mathcal{S}(\cdot)$; $s_{best} = +\infty$
  while $|\mathcal{E}.\text{unique}()| \leq M$ **do**
    $\mathcal{NN}^f_{\theta^*} \in \arg\min_{\mathcal{NN}^f_\theta \in \mathcal{M}} \mathcal{S}\left(\mathcal{E} \cup \{\mathcal{NN}^f_\theta\}\right)$
    if $\mathcal{S}\left(\mathcal{E} \cup \{\mathcal{NN}^f_\theta\}\right) < s_{best}$ **then**
      $\mathcal{E} = \mathcal{E} \cup \{\mathcal{NN}^f_\theta\}$
    else
output $\mathcal{E}$
    end if
  end while
output $\mathcal{E}$

Note that the Algorithm 1 does not require the NNs to have the same random weight initialization. However, Wenzel et al. (2020b) consider a fixed initialization for HDE to isolate the effect of just varying the hyperparameters.

Finally, Algorithm 2 builds an ensemble of at most M unique NNs which can exploit both sources of diversity: different *random initializations* and different choices of *hyperparameters*.

[18] Note that we use a slightly different notation than Wenzel et al. (2020a) here.

Algorithm 2 `hyper_deep_ens` (Wenzel et al., 2020b)

input M, κ
  $\mathcal{M}_0 = \{\mathcal{NN}^f_{\theta^{(j)},\lambda^{(j)}}\}_{j=1}^{\kappa} \leftarrow \texttt{rand\_search}(\kappa)$
  $\mathcal{E}_0 \leftarrow \texttt{hyper\_ens}(\mathcal{M}_0, M)$
  $\mathcal{E}_{strat} = \{\}$
  for all $\mathcal{NN}^f_{\theta,\lambda} \in \mathcal{E}_0.\text{unique}()$ **do**
    for all $m \in \{1, \ldots, M\}$ **do**
      $\theta' \leftarrow$ random initialization (seed number m)
      $\mathcal{NN}^f_{\theta^{(m)},\lambda} \leftarrow$ train $\mathcal{NN}^f_{\theta',\lambda}$
      $\mathcal{E}_{strat} = \mathcal{E}_{strat} \cup \{\mathcal{NN}^f_{\theta^{(m)},\lambda}\}$
    end for
  end for
output $\texttt{hyper\_ens}(\mathcal{E}_{strat}, M)$

B.2. Regression

B.2.1. Metrics

In this section, we provide details on the three metrics, which we use to assess the quality of UBs in the regression settings. In the following, let

$$\left(\underline{UB}_c(x), \overline{UB}_c(x)\right) := \left(\hat{f}(x) \mp c\,\hat{\sigma}_f(x)\right) \tag{46}$$

denote UBs obtained from any of the considered models via a model prediction $\hat{f}$, a model uncertainty prediction $\hat{\sigma}_f$, and a calibration parameter $c \in \mathbb{R}_{\geq 0}$. Furthermore, we use the following shorthand notation:

$$\mathcal{UB}_c(x) := \left(\underline{UB}_c(x), \overline{UB}_c(x)\right). \tag{47}$$

In the following, let $D := \{(x_i, y_i) \in X \times Y, i \in \{1, \ldots, n\}\}$, with $n \in \mathbb{N}$ denote a set of input-output points (depending on the specific purpose, D would typically refer to a validation or test set).

Mean Width vs. Coverage Probability We first formalize the concepts of mean width (MW) and coverage probability (CP). Then we define the metric AUC.

Definition B.2 (Coverage Probability) *Let D denote a set of input-output points, and let $\mathcal{UB}_c$ denote UBs for a calibration parameter c. Then the coverage probability is defined as*

$$CP\left(D \,|\, \mathcal{UB}_c\right) := \frac{1}{|D|} \sum_{(x,y)\in D} \mathbb{1}_{\{\underline{UB}_c(x) \leq y \leq \overline{UB}_c(x)\}}. \tag{48}$$

Definition B.3 (Mean Width) *Let D denote a set of input-output points, and let $\mathcal{UB}_c$ denote UBs for a calibration parameter c. Then the mean width is defined as*

$$MW\left(D \,|\, \mathcal{UB}_c\right) := \frac{1}{|D|} \sum_{(x,y)\in D} |\overline{UB}(x) - \underline{UB}(x)|. \tag{49}$$

Remark B.4 *Note that in Definition B.3, uncovered points at which UBs (for some fixed calibration parameter c) are narrow have a positive effect on overall mean width. In order not to reward such overconfident mispredictions, a possible remedy is to consider in (49) only the subset $D^{capt} \subset D$ of input-output points captured by the UBs, i.e.,*

$$D^{capt} := \left\{(x, y) \in D : \underline{UB}(x) \leq y \leq \overline{UB}(x)\right\}.$$

However, focusing on captured data points only punishes UBs that capture some points with large widths and almost cover others with small widths unnecessarily harshly compared to UBs for which the reverse is true. In other words, a slight change in calibration parameter c can lead to very diverse evaluations of UBs that have been assessed almost equal under the original c. Since ultimately we are interested in comparing UBs based on a range of calibration parameters (see Measure 1) we decided to include all points in the calculation of MW in our experiments.

Ideally, MW should be as small as possible, while CP should be close to its maximal value 1. Clearly, CP is counteracting MW. This naturally motivates considering ROC-like curves, plotting MW against CP for a range of calibration parameters c, and comparing different UBs based on their *area under the curves* (AUC).

Metric 1 (AUC) *Let D denote a set of input-output points. Define further c^* as the minimal calibration parameter achieving full coverage of D for given UBs $\mathcal{UB}_c$, i.e.,*

$$c^* := \arg\min_{c \geq 0} \{CP\left(D \,|\, \mathcal{UB}_c\right) = 1\}.$$

AUC is then defined as the integral of the following curve

$$\{\left(CP\left(D \,|\, \mathcal{UB}_c\right), MW\left(D \,|\, \mathcal{UB}_c\right)\right) \,:\, c \in [0, c^*]\}.^{19}$$

Negative Log-likelihood Next, we first define the second metric: *average*[20] *negative log (Gaussian) likelihood (NLL)* and then present our third metric NLL_{min}.

Metric 2 (NLL) *Let D denote a set of input-output points, and let $\mathcal{UB}_c$ denote UBs for a calibration parameter c with corresponding model prediction $\hat{f} : X \to \mathbb{R}$ and model uncertainty prediction $\hat{\sigma}_f : X \to \mathbb{R}_{\geq 0}$. Then NLL is defined as*

$$NLL(D|\mathcal{UB}_c) := \tag{50}$$
$$\frac{1}{|D|} \sum_{(x,y) \in D} \left[\frac{\left(y - \hat{f}(x)\right)^2}{2\left(c\hat{\sigma}_f(x)\right)^2} + \ln\left(c\hat{\sigma}_f(x)\right) \right] + \ln(2\pi)/2$$

The first term in NLL measures the error of the model prediction, where large errors can be attenuated by large uncertainties while the second term penalizes logarithmically larger uncertainties. Thus, NLL penalizes both over-confident ($c\hat{\sigma}_f(x) \approx 0$) wrong predictions as well as under-confident ($c\hat{\sigma}_f(x) \gg 0$) predictions.

We define the third metric as the minimal value of the NLL when varying the calibration parameter c.

Metric 3 (MINIMUM NLL) *Let D denote a set of input-output points, and let $\mathcal{UB}_c$ denote UBs for a calibration parameter c. Then NLL_{min} is defined as*

$$NLL_{min} := \min_{c \in \mathbb{R}_{\geq 0}} NLL(D|\mathcal{UB}_c). \tag{51}$$

Remark B.5 (CALIBRATION) *Different approaches of obtaining UBs require rather different scaling to reach similar CP (as well as MW). For instance, we found that the calibration parameter c required to achieve a certain value of CP typically is larger for DE than those of the remaining methods by a factor of approximately 10. Therefore, in practice it is important to find a well-calibrated parameter c. Standard calibration techniques can be applied to NOMU (e.g., methods based on isotonic regression (Kuleshov et al., 2018), or when assuming Gaussian marginals of the posterior one can select c as the $\arg\min_{c \in \mathbb{R}_{\geq 0}} NLL(D|\mathcal{UB}_c)$ on a validation set D).*

B.2.2. TOY REGRESSION: CONFIGURATION DETAILS

All NN-based methods are fully-connected feed-forward NNs with ReLU activation functions, implemented in TENSORFLOW.KERAS and trained for 2^{10} epochs of TENSORFLOW.KERAS' Adam stochastic gradient descent with standard learning rate 0.001 and full batch size of all training points. Moreover, weights and biases are initialized uniformly in the interval $[-0.05, 0.05]$.

NOMU Setup For the MC-integration of the integral in the NOMU loss (4), i.e., term (c), we use $l = 128$ and $l = 256$ artificial input points (we draw new artificial input points in every gradient descent step) in 1D and 2D, respectively.[21] For numerical stability, we use the parameters θ

[19] In our experiments, we approximated this integral via the trapezoidal rule.

[20] We remark that NLL can be interpreted as average marginal predictive density over the set D, assuming these marginals are Gaussian with mean $\hat{f}$ and variance $c\hat{\sigma}_f$. In particular, we refrain from posing assumptions of posterior independence, that we believe are highly flawed, i.e., NLL should not be interpreted as joint negative log predictive density.

[21] In 1D, we tried MC-integration based on uniform samples and MC-integration based on a deterministic grid, where both approaches led to qualitatively similar results and the latter is presented in this paper.

Table 5. Detailed results of the AUC metric for NOMU, GP, MCDO, DE, HDE and HDE* (our HDE see Appendix B.2.2) for all ten 1D synthetic functions. Shown are the medians and a 95% bootstrap confidence interval over 500 runs. Winners are marked in grey.

	NOMU		GP		MCDO		DE		HDE		HDE*	
FUNCTION	AUC↓	95%-CI	AUC↓	95%-CI	AUC	95%-CI	AUC↓	95%-CI	AUC↓	95%-CI	AUC↓	95%-CI
ABS	0.07	[0.07, 0.08]	0.14	[0.13, 0.16]	0.16	[0.15, 0.17]	0.08	[0.07, 0.09]	1.02	[0.82, 1.20]	0.72	[0.65, 0.81]
STEP	0.29	[0.26, 0.31]	0.93	[0.85, 1.02]	0.25	[0.24, 0.27]	0.14	[0.13, 0.15]	1.90	[1.61, 2.67]	0.82	[0.66, 0.99]
KINK	0.08	[0.08, 0.09]	0.17	[0.15, 0.19]	0.22	[0.21, 0.23]	0.09	[0.07, 0.10]	1.81	[1.40, 2.31]	1.01	[0.9 , 1.24]
SQUARE	0.12	[0.11, 0.13]	0.16	[0.13, 0.19]	0.38	[0.36, 0.41]	0.14	[0.12, 0.15]	1.72	[1.50, 2.23]	1.86	[1.54, 2.53]
CUBIC	0.07	[0.07, 0.08]	0.00	[0.00, 0.00]	0.10	[0.10, 0.11]	0.10	[0.09, 0.11]	0.63	[0.54, 0.79]	0.52	[0.45, 0.66]
SINE 1	1.10	[1.03, 1.16]	1.14	[1.09, 1.2]	0.90	[0.87, 0.93]	1.21	[1.16, 1.26]	9.38	[7.42, 11.4]	9.75	[7.64, 12.1]
SINE 2	0.38	[0.37, 0.40]	0.42	[0.41, 0.44]	0.36	[0.35, 0.36]	0.50	[0.47, 0.53]	3.18	[2.82, 4.08]	2.43	[2.12, 3.38]
SINE 3	0.20	[0.19, 0.21]	0.31	[0.29, 0.34]	0.28	[0.26, 0.29]	0.20	[0.19, 0.21]	1.43	[1.19, 1.73]	1.33	[1.10, 1.57]
FORRESTER	0.19	[0.18, 0.20]	0.18	[0.17, 0.19]	0.26	[0.24, 0.28]	0.25	[0.24, 0.27]	1.28	[1.11, 1.51]	1.58	[1.35, 1.93]
LEVY	0.40	[0.39, 0.42]	0.53	[0.51, 0.56]	0.38	[0.36, 0.39]	0.45	[0.43, 0.48]	3.43	[2.67, 4.42]	3.58	[2.83, 4.18]

Table 6. Detailed results of the NLL_{min} metric (without constant $\ln(2\pi)/2$) for NOMU, GP, MCDO, DE, HDE and HDE* (our HDE see Appendix B.2.2) for all ten 1D synthetic functions. Shown are the medians and a 95% bootstrap confidence interval over 500 runs. Winners are marked in grey.

	NOMU		GP		MCDO		DE		HDE		HDE*	
FUNCTION	NLL_{MIN}↓	95%-CI	NLL_{MIN}↓	95%-CI	NLL_{MIN}↓	95%-CI	NLL_{MIN}↓	95%-CI	NLL_{MIN}↓	95%-CI	NLL_{MIN}↓	95%-CI
ABS	−2.85	[−2.95, −2.80]	−2.82	[−2.90, −2.71]	−1.76	[−1.80, −1.69]	−2.91	[−3.03, −2.76]	−0.19	[−0.36, 0.00]	−0.72	[−0.82, −0.56]
STEP	−1.04	[−1.12, −0.97]	−0.61	[−0.81, −0.49]	−0.78	[−0.85, −0.69]	−2.49	[−2.59, −2.36]	0.61	[0.38, 0.83]	−1.18	[−1.42, −0.84]
KINK	−2.74	[−2.80, −2.69]	−2.64	[−2.75, −2.54]	−1.28	[−1.32, −1.22]	−2.81	[−2.93, −2.68]	0.56	[0.26, 0.83]	−0.35	[−0.46, −0.17]
SQUARE	−2.45	[−2.51, −2.41]	−3.80	[−4.21, −3.38]	−0.56	[−0.65, −0.52]	−2.27	[−2.37, −2.16]	0.53	[0.31, 0.73]	0.07	[−0.17, 0.22]
CUBIC	−2.98	[−3.04, −2.94]	−6.03	[−6.15, −5.92]	−2.41	[−2.47, −2.33]	−2.65	[−2.70, −2.59]	−0.77	[−0.93, −0.64]	−1.42	[−1.63, −1.15]
SINE 1	0.10	[0.04, 0.18]	−0.06	[−0.13, 0.01]	0.09	[0.07, 0.13]	0.09	[0.04, 0.16]	1.97	[1.71, 2.28]	1.65	[1.45, 1.84]
SINE 2	−1.35	[−1.38, −1.32]	−1.48	[−1.52, −1.42]	−1.03	[−1.06, −1.00]	−0.92	[−0.96, −0.88]	1.21	[0.95, 1.65]	0.45	[0.26, 0.73]
SINE 3	−1.55	[−1.61, −1.47]	−1.69	[−1.85, −1.55]	−1.07	[−1.12, −1.01]	−1.66	[−1.75, −1.58]	0.26	[0.10, 0.40]	−0.02	[−0.21, 0.21]
FORRESTER	−1.98	[−2.03, −1.91]	−2.68	[−2.86, −2.49]	−1.10	[−1.19, −1.03]	−1.75	[−1.79, −1.69]	−0.11	[−0.23, 0.17]	−0.50	[−0.65, −0.28]
LEVY	−1.12	[−1.14, −1.07]	−0.93	[−0.98, −0.90]	−0.76	[−0.79, −0.73]	−1.04	[−1.08, −0.97]	1.35	[0.99, 1.53]	0.74	[0.53, 0.96]

that gave the best training loss during the training which are not necessarily the ones after the last gradient descent step.

Deep Ensembles Setup We consider the proposed number of five ensembles (Lakshminarayanan et al., 2017) with a single output $\hat{\mu}^\theta$ only (accounting for zero data noise), each with three hidden layers consisting of $2^8, 2^{10}$ and 2^9 nodes (resulting in ≈ 4 million parameters). We train them on standard regularized mean squared error (MSE) with regularization parameter $\lambda = 10^{-8}/n^{\text{train}}$ chosen to represent the same data noise assumptions as NOMU.

MC Dropout Setup The MC dropout network is set up with three hidden layers as well, with $2^{10}, 2^{11}$ and 2^{10} nodes (resulting in ≈ 4 million parameters). Both training and prediction of this model is performed with constant dropout probability $p := p_i = 0.2$, proposed in (Gal & Ghahramani, 2016). We perform 100 stochastic forward passes. To represent the same data noise assumptions as NOMU and deep ensembles we set the regularization parameter to $\lambda = (1-p) \cdot 10^{-8}/n^{\text{train}} = (1-0.2) \cdot 10^{-8}/n^{\text{train}}$ (based on equation (7) from (Gal & Ghahramani, 2016)).

Gaussian Process Setup Finally, we compare to a Gaussian process[22] with RBF-kernel,

$$k_\pi(x, x') := \kappa \cdot e^{-\left(\frac{\|x-x'\|_2^2}{h^2}\right)}, \tag{52}$$

with hyperparameters $\pi := (\kappa, h)$, where both the prior variance parameter κ and the length scale parameter h (initialized as $\kappa = h = 1$) are optimized in the default range $[10^{-5}, 10^5]$ using 10 restarts and the data noise level[23] is set to 10^{-7}.

Hyper Deep Ensembles Setup We consider the same network architectures as for DE with a single output $\hat{\mu}^\theta$ only (accounting for zero data noise), each with three hidden layers consisting of $2^8, 2^{10}$ and 2^9 nodes (resulting in ≈ 4 million parameters). We train them on standard regularized mean squared error (MSE).

Furthermore, we use the following proposed parameter values from Wenzel et al. (2020b) to build the ensembles: we consider $M := 5$ networks (termed K in (Wenzel et al., 2020b)) in the final ensemble and search over $\kappa = 50$ networks resulting from `random_search`. Moreover, as random hyperparameters we use the pro-

[22] We use *GaussianProcessRegressor* from SCIKIT-LEARN.

[23] This is mainly used for numerical stability.

Table 7. Aggregate results for NOMU, GP, MCDO, DE, HDE and HDE* (our HDE see Appendix B.2.2) for the set of all ten 1D synthetic functions. Shown are the medians and a 95% bootstrap CI of AUC and NLL_{min} (without constant $\ln(2\pi)/2$) across all runs. Winners are marked in grey.

METHOD	AUC↓	95%-CI	NLL_{MIN}↓	95%-CI
NOMU	0.21	[0.21, 0.22]	-1.76	[-1.81, -1.72]
GP	0.33	[0.32, 0.35]	-1.74	[-1.81, -1.69]
MCDO	0.30	[0.29, 0.30]	-1.02	[-1.05, -0.99]
DE	0.22	[0.21, 0.23]	-1.75	[-1.79, -1.70]
HDE	2.04	[1.90, 2.20]	0.59	[0.51, 0.66]
HDE*	1.73	[1.63, 1.84]	0.04	[-0.03, 0.10]

Table 8. Aggregate results for NOMU, GP, MCDO, DE, HDE and HDE* (our HDE see Appendix B.2.2) for the set of all eleven 2D synthetic functions. Shown are the medians and a 95% bootstrap CI of of AUC and NLL_{min} (without constant $\ln(2\pi)/2$) across all runs. Winners are marked in grey.

METHOD	AUC↓	95%-CI	NLL_{MIN}↓	95%-CI
NOMU	0.42	[0.41, 0.42]	-0.99	[-1.01, -0.96]
GP	0.36	[0.36, 0.37]	-1.07	[-1.09, -1.05]
MCDO	0.45	[0.44, 0.45]	-0.61	[-0.62, -0.60]
DE	0.42	[0.41, 0.43]	-0.95	[-0.97, -0.93]
HDE	5.06	[4.75, 5.49]	1.74	[1.66, 1.84]
HDE*	6.33	[5.87, 6.75]	1.58	[1.51, 1.67]

posed L2-regularization and the `dropout_rate`, which we draw as in (Wenzel et al., 2020b) log-uniform from $(0.001, 0.9)$ and $(10^{-8} \cdot 10^{-3}, 10^{-8} \cdot 10^{3})$, respectively. Furthermore, we use the proposed train/validation split of 80%/20% and as score $\mathcal{S}$ the NLL. Finally, as for MCDO and DE we scale the realized L2-regularization by $(1 - \texttt{dropout\_rate})/\texttt{floor}(0.8 \cdot n^{\text{train}})$ chosen to represent the same data noise assumptions as NOMU.

For our HDE version termed **HDE***, we use a train/validation split of 70%/30%, draw the dropout probability from $(0.001, 0.5)$ and continue training the final ensemble again on *all* n^{train} training points (rescaling the realized L2-regularization again by $\texttt{floor}(0.8 \cdot n^{\text{train}})/n^{\text{train}}$).

B.2.3. Toy Regression: Detailed Results

Results 1D In Table 5 (AUC) and Table 6 (NLL_{min}), we present detailed results, which correspond to the presented ranks from Table 1 in the main paper. In Table 7, we present aggregate results, i.e., median values (incl. a 95% bootstrap confidence interval (CI)) of AUC and NLL_{min} across *all* 5000 runs (500 runs for 10 test functions) for each algorithm. We see that NOMU performs well on both metrics and yields numbers similar to those of DE. The GP is only competitive on NLL_{min}. MCDO performs in both metrics worse than NOMU and DE. Not surprisingly, HDE, shown to be the state-of-the-art ensemble algorithm (Wenzel et al., 2020b), fails to produce reliable model uncertainty estimates in such a scarce and noiseless regression setting (see Appendix B.2.4 for a discussion).

Results 2D Test functions for 2D input are taken from the same library as in the 1D setting.[24] Specifically, we select the following 11 different test functions: *Sphere*, *G-Function*, *Goldstein-Price*, *Levy*, *Bukin N.6*, *Rosenbrock*, *Beale*, *Camel*, *Perm*, *Branin*, *Styblisnki-Tang*.

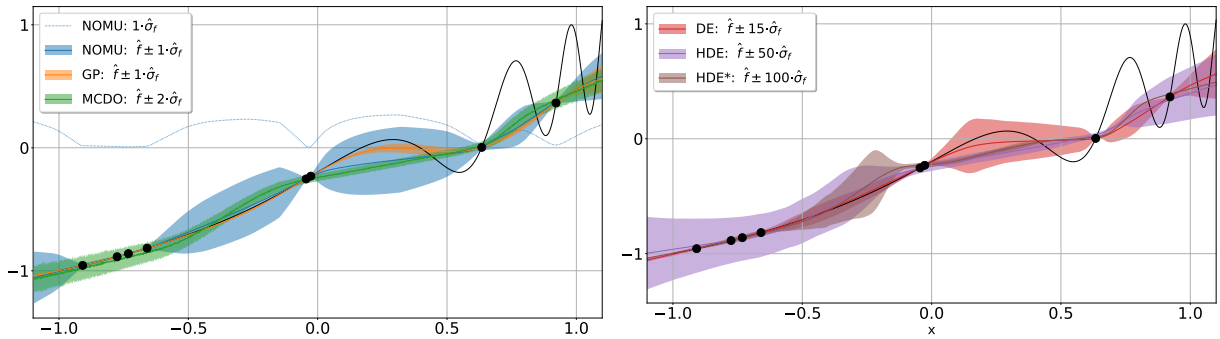
In Table 8, we present median values of AUC and NLL_{min} across *all* 5500 runs (500 runs for 11 test functions) for each algorithm. We also provide a 95% bootstrap confidence interval (CI) to assess statistical significance. We observe a similar ranking of the different algorithms as in 1D, however, now GP is ranked first in AUC, followed by NOMU and DE; and the GP is also ranked first in NLL_{min} followed by NOMU and DE, who share the second rank.

[24] See sfu.ca/ ssurjano/optimization.html. All test functions are scaled to $X = [-1, 1]^2$ and $f(X) = [-1, 1]$.

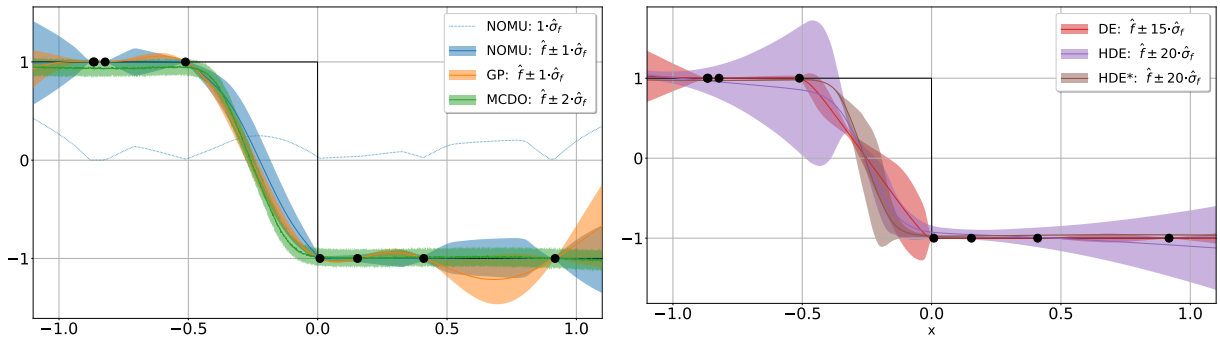
B.2.4. Toy Regression: Detailed Characteristics of Uncertainty Bounds

UB Characteristics in 1D Within the one-dimensional setting, characteristics of NOMU UBs can be nicely compared to those of the benchmark approaches. Figure 9 exemplarily shows typical outcomes of the UBs on a number of selected test functions as obtained in one run.

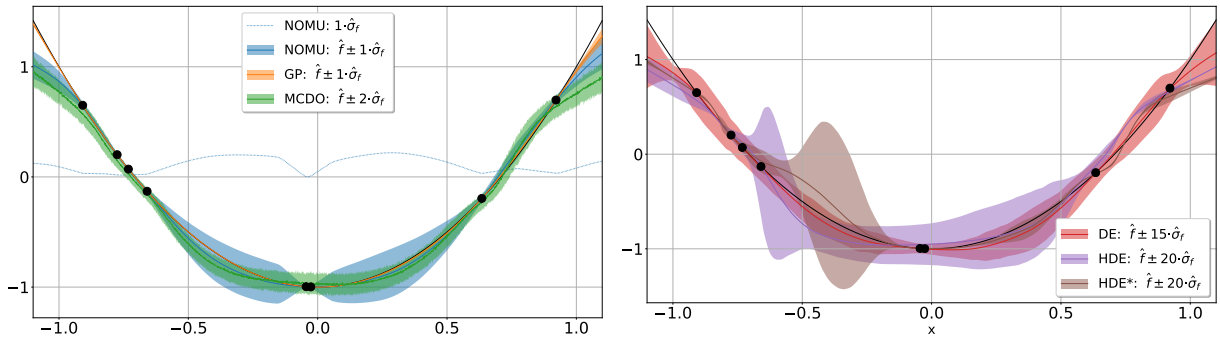
- **Deep Ensembles UBs:** Throughout the experiment, we observe that deep ensembles, while sometimes nicely capturing increased uncertainty in regions of high second derivative (e.g., around the kink in Figure 9e; pursuant in some form desiderata D4), still at times leads to UBs of somewhat arbitrary shapes. This can be seen most prominently in Figure 9b around $x \approx -0.25$, in Figure 9f around $x \approx -0.8$ or in Figure 9d around $x \approx 0.2$. Moreover, deep ensembles' UBs are very different in width, with no clear justification. This can be seen in Figure 9b when comparing the UBs for $x \geq 0$ against the UBs for $x \leq 0$ and at the edges of the input range in Figures 9a and 9f. In addition, we frequently observe that deep ensembles UBs do not get narrow at training points, as for instance depicted in Figure 9c for $x < -0.5$, in Figure 9f for $x > 0.2$, or in Figure 9d for $x \in [-1, -0.7]$ and thus are not able to handle well small or zero data noise.
- **Hyper Deep Ensembles UBs:** Throughout our experiments, HDE produces less accurate UBs, which vary more from one run to another, than DE (recall that here we consider the setting of noiseless scarce regression). Possible reasons for this are:
  1. The scarcity of training/validation data. HDE trains



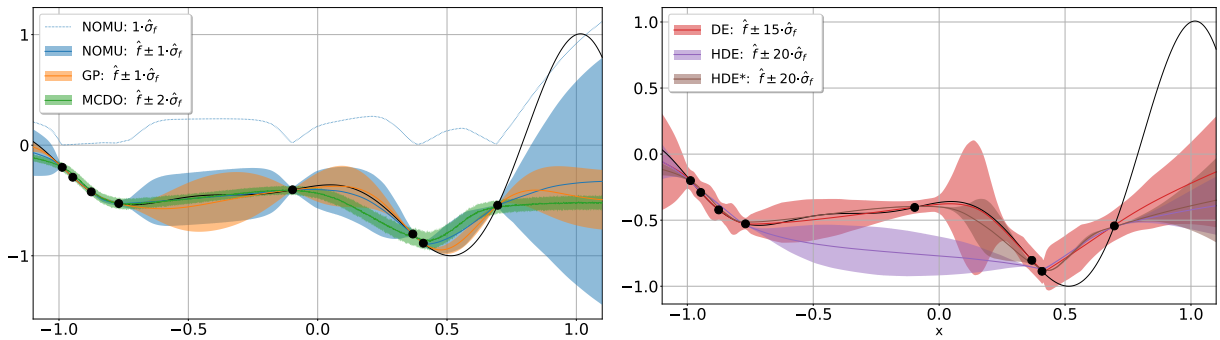
(a) Sine 3



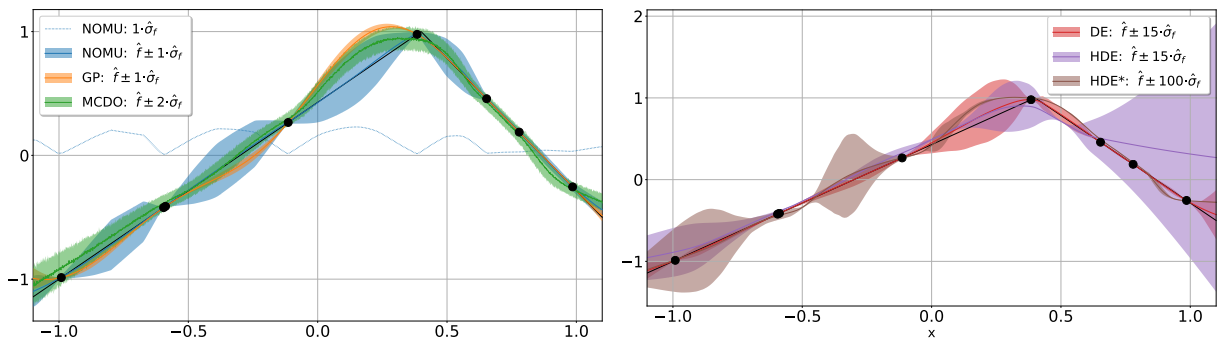
(b) Step



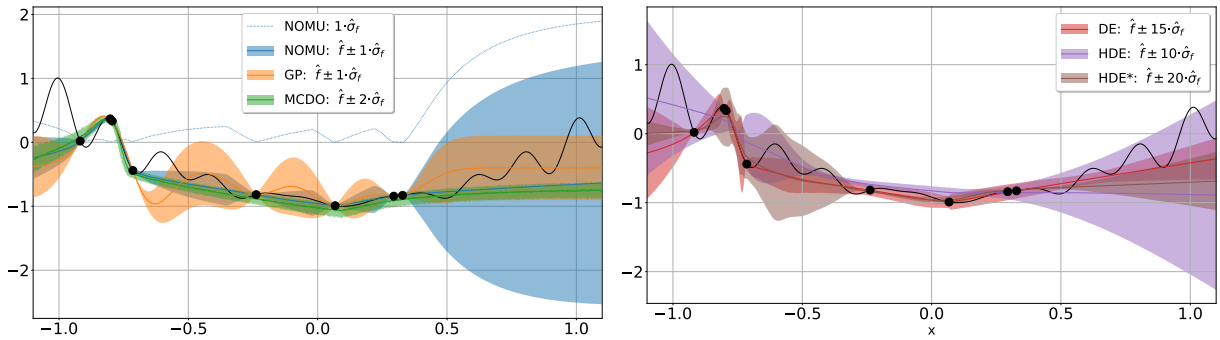
(c) Squared (GP would not behave reasonable for standard hyper-parameters in this instance, so we changed the range for the prior variance parameter κ to be $[10^{-5}, 4]$.)



(d) Forrester



(e) Kink



(f) Levy

Figure 9. Visualisations of resulting UBs for NOMU, GP, MCDO, DE, HDE and HDE* for a selection of test functions.

its NNs based on 80% of the training points and uses the remaining 20% to build an ensemble based on a score, whilst the other methods can use 100% of the training points for training. In a scarce data setting this implies that first, the mean prediction of HDE does not fit through all the training points, and second, the scoring rule is less reliable.

2. In a noiseless setting one already knows that the L2-regularization should be small, and thus optimizing this parameter is less useful here.

Therefore, we believe HDE is less well suited in a noiseless scarce regression setting to reliably capture model uncertainty. This manifests in Figure 9. HDE's uncertainty bounds often do not narrow at training points (Figures 9a–9c and 9e), while they suggest unreasonably over-confident mean predictions in some regions (Figures 9b and 9d). While our HDE* (see Appendix B.2.2 Hyper Deep Ensembles) manages to better narrow at training data, it tends to more frequently result in unnecessarily narrow bounds (Figures 9a, 9c and 9d). Overall, both HDE and HDE* vary a lot from run to run and thus tend to yield bounds of seemingly arbitrary shapes, where in each run the desiderata are captured in different regions.

- **MC Dropout UBs:** MCDO consistently yields tube-like UBs that do not narrow at training points. Interestingly, we remark that throughout the experiment MCDO samples frequently exhibit stepfunction-like shapes (e.g., see Figure 9d at $x \approx 0.5$ or Figure 9b for $x \in [-0.5, 0]$). This effect intensifies with increasing training time.
- **NOMU UBs:** In contrast, NOMU displays the behaviour it is designed to show. Its UBs nicely tighten at training points and expand in between and thus NOMU fulfills D1–D3 across all considered synthetic test functions.
- **Gaussian Process UBs:** The quality of the RBF-Gaussian process' UBs (as judged in this simulated setting) naturally varies with the true underlying function. While the UBs nicely capture parts of the true function with low widths in Figures 9c and 9d they have a hard time accurately enclosing true functions that are not as conformant with the prior belief induced by the choice of the RBF kernel (e.g., Figures 9b and 9e). Nonetheless, we also observe instances in which the training points are misleading to the GP's mean predictions even when considering ground truth functions for which this choice of kernel is very suitable. This manifests in over-confident mean predictions far away from the data generating true function (Figure 9a) or over-swinging behavior of the fitted mean (Figure 9f). It is true that one could find better

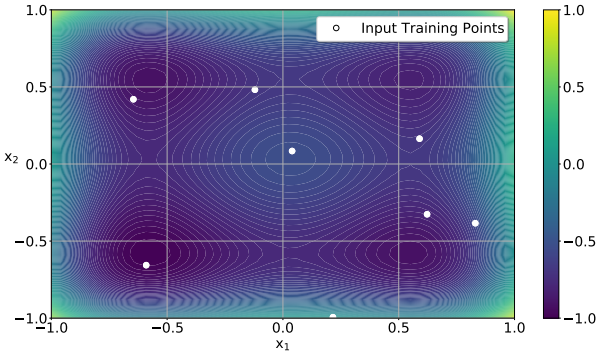


Figure 10. Contour plot of the 2D Styblinski test function.

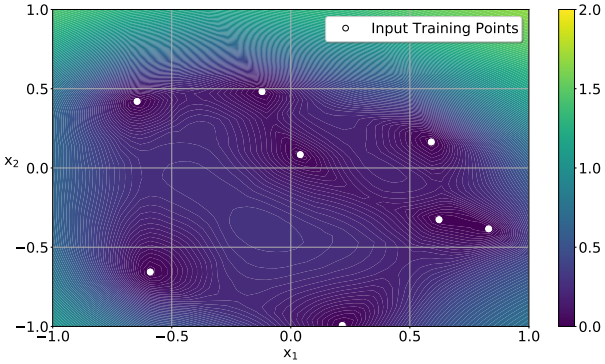
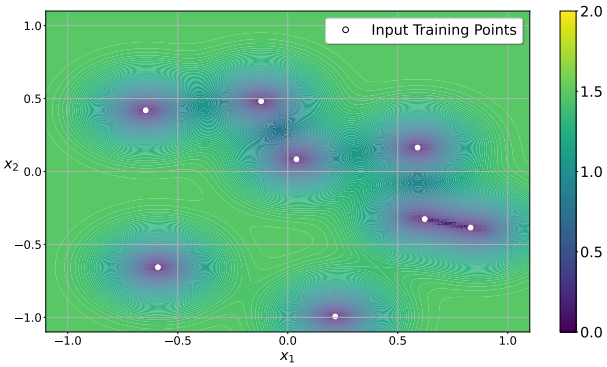


Figure 11. Estimated model uncertainty $\hat{\sigma}_f$ of NOMU for a single run of the Styblinski test function.

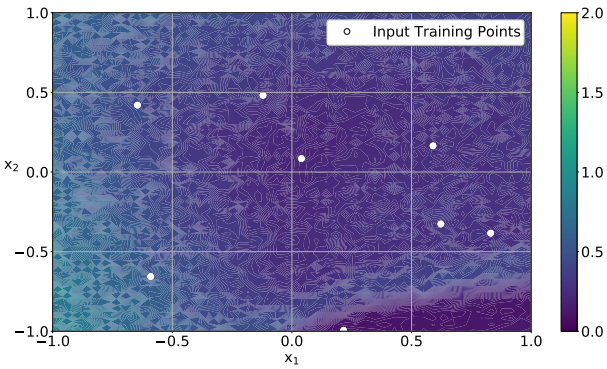
function-specific kernels. However, the RBF kernel is a good generic choice when evaluated across many different test functions without any specific prior information, which is why we choose this kernel for our comparison.

UB Characteristics in 2D To visualize the UBs in 2D, we select as the ground truth the *Styblinski* test function depicted in Figure 10. In Figure 11 (NOMU) and Figure 12 (benchmarks), we visualize the estimated model uncertainty as obtained in one run for this *Styblinski* test function.

- **NOMU (Figure 11):** As in 1D, we observe that NOMU's UBs nicely tighten at input training points and expand in-between, with overall steady shapes. Specifically, NOMU's UBs are larger for extrapolation, e.g., $[0, 1] \times [0.5, 1]$, compared to regions which are *surrounded* by input training data points, e.g., $[-0.25, 0.25] \times [-0.25, -0.75]$, even though the distance to the closest input training point is approximately the same. Thus, NOMU's UBs do not only depend on the distance to the closest training point but rather on the arrangement of all surrounding training points.
- **Benchmarks (Figure 12):** As in 1D, we see that the RBF-based GP's UBs do have the expected smooth and isotropic shape with zero model uncertainty at input training points. Moreover, as in 1D, MCDO's UBs exhibit a tubular shape of equal size ($\approx$ 0–0.25) across the whole input space. Whilst DE nicely captures zero model uncertainty at input training points, it again exhibits the somehow arbitrary behaviour in areas with few input training points. Both HDE and HDE* yield model uncertainty estimates that are small at input training points, except for HDE for one input training point $(x_1, x_2) \approx (0.6, 0.2)$, where the estimated model uncertainty is only small when continuing training on all training points (see HDE*). However, these estimates drastically fail to capture increased uncertainty in out-of-sample regions. For both algorithms, the model uncertainty estimate is unreasonably low in most regions of the input space and inexplicably high in a tiny fraction of the input space.

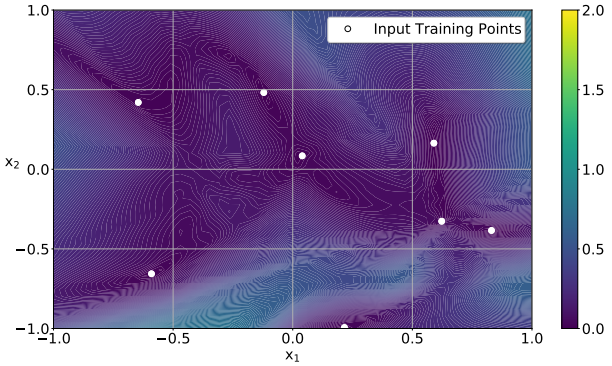


(a) GP with calibration constant $c = 1$.

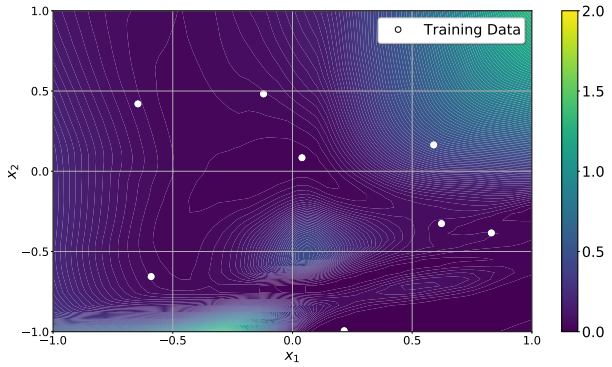


(b) MCDO with calibration constant $c = 10$.

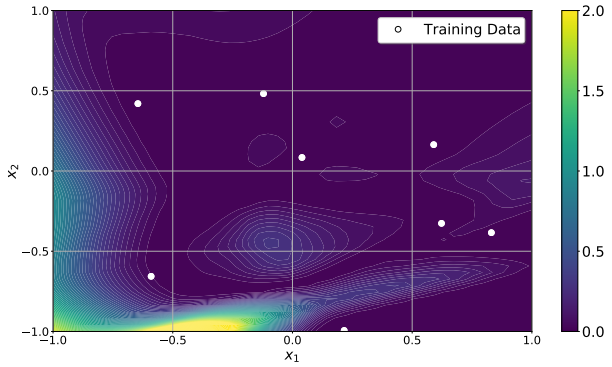
Figure 12. Estimated model uncertainty of Gaussian process (GP) and MC dropout (MCDO) for a single run of the Styblinski test function.



(c) DE with calibration constant $c = 15$.



(d) HDE with calibration constant $c = 30$.



(e) HDE* with calibration constant $c = 30$.

Figure 12. (cont.) Estimated model uncertainty of deep ensembles (DE), hyper deep ensembles (HDE) and HDE* for a single run of the Styblinski test function.

B.2.5. Generative Test-Bed: Details

In this section, we give a detailed description of the generative test-bed setting and provide further results for up to 20 dimensional input.

Detailed Setting

1. We sample a test-function (i.e., the ground truth function) $f : [-1, 1]^d \to \mathbb{R}$ from a BNN with i.i.d Gaussian weights. This means we sample i.i.d random parameters $\theta_i \sim \mathcal{N}(0, \sigma_{\text{prior}})$ and set $f = \mathcal{NN}_\theta$. The BNN is set up with three hidden layers with $2^{10}, 2^{11}$ and 2^{10} nodes, respectively. We choose σ_{prior} such that $\mathbb{E}_\theta \left[\sqrt{\mathbb{V}_x[\mathcal{NN}_\theta(x) \mid \theta]} \right] \approx 1$.[25] Resulting values for σ_{prior} are shown in Table 9.

Table 9. σ_{prior} depending on the input dimension d.

d	σ_{PRIOR}
1	0.114
2	0.102
5	0.092
10	0.084
20	0.070

2. We sample $n^{\text{train}} = 8 \cdot d$ input training points uniformly at random from the input space $[-1, 1]^d$ for Tables 2 and 10. In the case of Table 11, we sample $n^{\text{train}} = 8 \cdot d$ input training points from d-dimensional centered normal distribution with a covariance with $\min(5, d)$ eigenvalues of value 0.15 and the remaining eigenvalues only have the value 0.001. In both cases we get for input training point x_i^{train} the corresponding noiseless output-training point $y_i^{\text{train}} = f(x_i^{\text{train}}) = \mathcal{NN}_\theta(x_i^{\text{train}})$. And in both cases we create $n^{\text{test}} = 100 \cdot d$ test-data points from the same distribution as the training data points.[26]
3. We calculate UBs of all considered algorithms including NOMU. For this, we use for all algorithms the same configuration as in the 1D and 2D toy regression settings (see Section 4.1.1 and Appendix B.2.2) except for NOMU, where we set $\ell_{\text{min}} = 0.1$, $\ell_{\text{max}} = 1$ and use $l = 100 \cdot d$ artificial data points, where d denotes the respective dimension.
4. We measure the quality of the UBs via NLL for a fine grid of different values for the calibration parameter c.

We repeat these four steps $m_{\text{seeds}} = 200$-times. Then we choose for each method the value c where the average NLL is minimal. Importantly, the chosen calibration constant c *only* depends on the input-dimension d and on the algorithm but not on the randomly chosen test-function f. In Table 10 and Table 11, we report the mean and a 95% CI over these 200 runs for the chosen calibration constant c.

[25] $\mathbb{E}_\theta \left[\sqrt{\mathbb{V}_x[\mathcal{NN}_\theta(x) \mid \theta]} \right] \approx 1$ holds true for $x \sim \text{Unif}\left([-1, 1]^d\right)$. This might deviate (slightly) for Gaussian x.

[26] Note that we sample the artificial data points for NOMU always uniformly from $[-1, 1]^d$, since we assume that the low-dimensional manifold is often unknown in practice. NOMU could probably be improved if one tries to learn the distribution of the input data or if one already had some prior information about the distribution of the input data points. It is good to see how robustly NOMU can handle a different distribution of the input data points without the need to adapt the distribution of the artificial data points.

Table 10. Uniform data generation. Shown are average NLL (without constant $\ln(2\pi)/2$) and a 95% CI over 200 BNN samples.

FUNCTION	NOMU	GP	MCDO	DE	HDE	TUBE
BNN1D	-1.65±0.10	-1.08±0.22	-0.34±0.23	-0.38±0.36	8.47±1.00	-0.86±0.19
BNN2D	-1.16±0.05	-0.52±0.11	-0.33±0.13	-0.77±0.07	9.10±0.39	-0.81±0.06
BNN5D	-0.37±0.02	-0.33±0.02	-0.05±0.04	-0.13±0.03	8.41±1.00	-0.33±0.02
BNN10D	-0.09±0.01	-0.11±0.01	0.06±0.02	0.10±0.01	6.44±1.36	-0.12±0.01
BNN20D	0.15±0.01	0.06±0.01	0.18±0.01	0.30±0.01	3.58±0.81	0.07±0.01

Table 11. 5D-Gaussian data generation. Shown are average NLL (without constant $\ln(2\pi)/2$) and a 95% CI over 200 BNN samples.

FUNCTION	NOMU	GP	MCDO	DE	HDE	TUBE
BNN1D	-1.91±0.13	-1.13±0.27	-0.49±0.16	-0.27±0.54	8.95±1.18	-0.85±0.13
BNN2D	-1.55±0.05	-0.85±0.12	-0.62±0.13	-1.15±0.08	7.92±0.42	-0.95±0.11
BNN5D	-0.77±0.02	-0.72±0.02	-0.47±0.03	-0.55±0.03	9.68±1.08	-0.68±0.02
BNN10D	-1.36±0.01	-1.31±0.01	-1.08±0.03	-1.14±0.02	5.16±1.55	-1.25±0.01
BNN20D	-1.70±0.01	-1.72±0.01	-1.52±0.02	-1.53±0.01	0.99±0.39	-1.68±0.01

Discussion of the Results in Table 10 We see that for $d \leq 5$ NOMU significantly outperforms all other considered benchmarks. For dimensions $d \geq 10$ it gets harder to find a good metric for measuring the quality of the uncertainty bounds: For high dimensions almost no test data point sampled uniformly from $[-1, 1]^d$ is close to the training data points, so they all have quite high uncertainty. We further verified this for dimensions $d = 10$ and $d = 20$ by introducing another algorithm we call TUBE. TUBE uses NOMU's mean prediction and assigns the same (calibrated) constant uncertainty c to all test data points. As we can see in Table 10, TUBE is ranked first (en par with GP) in $d = 10$ and $d = 20$, highlighting that the metric NLL is flawed in this setting for dimensions $d \geq 10$. However, for dimensions $d \leq 5$ the trivial TUBE algorithm is significantly outperformed by other methods. Thus, the NLL-metric for dimensions $d \geq 10$ and uniformly distributed data on $[-1, 1]^d$ mainly measures if D3 (Out-of-Sample) is reliably fulfilled. D3 (Out-of-Sample) is particularly reliably fulfilled for TUBE and for GP. For BO however, especially D2 (In-Sample) is important to not get stuck in local optima, because in BO the test data-points are not uniformly distributed, and the predicted uncertainty close to the training points is particularly important. Therefore, in this setting, we only include results for input dimensions 1-5D (see Table 2) in the main part of the paper.

Discussion of the Results in Table 11 Another approach to circumvent the problem that the NLL metric loses its relevance for scenarios where the input test data points are all far away from the input training data points is to use a different distribution for X. The well known manifold-hypothesis (Cayton, 2005) states that in typical real world data-sets relevant input data points lie close to a low-dimensional manifold. Therefore, we run a further experiment where we sample from our training and test data from a Gaussian distribution that concentrates mainly close to a $\min(d, 5)$-dimensional flat manifold as described in Item 2 and report the results in Table 11. For this experiment, we see that dimension $d \geq 10$ are interesting too, since TUBE is significantly beaten by NOMU and GP. In Table 11, one can see that NOMU is among the best methods for all considered dimensions.

Our Generative Test-Bed vs. Osband et al. (2021) Osband et al. (2021) measures the Kullback–Leibler divergence between the posterior of any method to the posterior of shallow GP with a fixed (deep) neural tangent kernel (NTK). Thus, they evaluate methods by their similarity to a NTK-GP posterior. Because of the fixed kernel the posterior of the NTK-GP does not fulfill D4 (Metric Learning) at all. This implies that their evaluation metric does not reward D4 (Metric Learning) at all. However, we think that D4 (Metric Learning) is very important for real-world deep learning applications (see Footnote 4 and Appendix D.4). The posterior of a finite-width BNN fulfills D4 (Metric Learning). Therefore, approximating such a posterior is more desirable. However, in contrast to the NTK-GP posterior in (Osband et al., 2021), there is no closed formula for a finite-width BNN posterior. Thus, at first sight one cannot straightforwardly evaluate methods based on the Kullback–Leibler divergence between their posterior and this intractable BNN posterior as in Osband et al. (2021).

Nonetheless, in the following, we prove in Theorem B.7 that the metric we use in Tables 2, 10 and 11 indeed converges (up to a constant)[27] to the average Kullback–Leibler divergence $\bar{d}_{\text{KL}}$ to the posterior of a finite width BNN as m_{seeds} tends to infinity. First, we define the average Kullback–Leibler divergence $\bar{d}_{\text{KL}}$ and introduce some notation.

Definition B.6 (AVG KULLBACK–LEIBLER DIVERGENCE) *Let D^{train} be a finite set of training points and consider a prior distribution*[28] $\mathbb{P}[f \in \cdot]$ *on the function space $\{f : X \to Y\}$ and the corresponding posterior $\mathbb{P}[f \in \cdot \mid D^{\text{train}}]$ on the function space. Then the marginal of the posterior $\mathbb{P}[f(x) \in \cdot \mid D^{\text{train}}, x]$ is a measure on $\mathbb{R}$ for every given input data point $x \in X$ and every given training data set D^{train}. Let $\mathbb{Q}[\cdot \mid D^{\text{train}}, x]$ also be a measure on $\mathbb{R}$ for every given input data point $x \in X$ and every given training data set D^{train}.*[29] *The average Kullback–Leibler divergence is then defined as*

$$\bar{d}_{KL} = \mathbb{E}_{D^{\text{train}},x}\left[d_{KL}\left(\mathbb{P}[f(x) \in \cdot \mid D^{\text{train}}, x] \;||\; \mathbb{Q}[\cdot \mid D^{\text{train}}, x]\right)\right],$$

[27] If we are just interested in the relative performance of different methods compared to each other, the constant does not matter.

[28] In our generative test-bed the prior distribution is a BNN prior with i.i.d centered Gaussian weights.

[29] In our context $\mathbb{Q}[\cdot \mid D^{\text{train}}, x]$ can be the approximation of the marginal posterior at $x \in X$ given training data D^{train} obtained from any method such as NOMU, GP, MCDO, DE, HDE, etc.

where the expectation is taken over x and D^{train} according to $\mathbb{P}$, and d_{KL} is the classical Kullback–Leibler divergence between two probability measures on $\mathbb{R}$. This is equivalent to Equation (1) from (Osband et al., 2021).

Theorem B.7 *Using the notation from Definition B.6, let $q(\cdot \mid D^{train}, x)$ be the density of $\mathbb{Q}[\cdot \mid D^{train}, x]$ on $\mathbb{R}$. Let $(f_j(x_j), D_j^{train}, x_j)_{j\in\{1,\dots,m_{seeds}\}}$ be i.i.d samples of $\mathbb{P}[(f(x), D^{train}, x) \in \cdot]$.[30] Then, the average NLL*

$$\lim_{m_{seeds}\to\infty} \frac{1}{m_{seeds}} \sum_{j=1}^{m_{seeds}} -\log\left(q(f_j(x_j) \mid D_j^{train}, x_j)\right) = \bar{d}_{KL} + C_{\mathbb{P}}$$

converges ($\mathbb{P}$-a.s.) to $\bar{d}_{KL}$ up to an additive constant $C_{\mathbb{P}}$, where $C_{\mathbb{P}}$ only[31] depends on $\mathbb{P}$ and not on $\mathbb{Q}$.

Proof. Let p be the density of $\mathbb{P}[(f(x), D^{\text{train}}, x) \in \cdot]$ and $p(\cdot \mid D^{\text{train}}, x)$ the density on $\mathbb{R}$ of the marginal posterior $\mathbb{P}[f(x) \in \cdot \mid D^{\text{train}}, x]$. Further we write $p(D^{\text{train}}, x)$ for the density of of the marginal $\mathbb{P}[(D^{\text{train}}, x) \in \cdot]$ evaluated at (D^{train}, x). Since

$$\lim_{m_{\text{seeds}}\to\infty} \frac{1}{m_{\text{seeds}}} \sum_{j=1}^{m_{\text{seeds}}} -\log\left(q(f_j(x_j) \mid D_j^{\text{train}}, x_j)\right)$$

can be seen as a Monte-Carlo approximation, it converges ($\mathbb{P}$-a.s.) to

$$\begin{aligned}
&\mathbb{E}_{f(x),D^{\text{train}},x}[-\log(q(f(x) \mid D^{\text{train}}, x))] = \\
&= \int -\log(q(f(x) \mid D^{\text{train}}, x))\, p(f(x), D^{\text{train}}, x) d(f(x), D^{\text{train}}, x) = \\
&= \int -\log(q(f(x) \mid D^{\text{train}}, x))\, p(f(x) \mid D^{\text{train}}, x) p(D^{\text{train}}, x) d(f(x), D^{\text{train}}, x).
\end{aligned}$$

By Fubini this is equal to

$$\begin{aligned}
&\int\int -\log(q(f(x) \mid D^{\text{train}}, x))\, p(f(x) \mid D^{\text{train}}, x) d(f(x)) p(D^{\text{train}}, x) d(D^{\text{train}}, x) = \\
&= \mathbb{E}_{D^{\text{train}},x}\Bigg[\underbrace{\int -\log(q(f(x) \mid D^{\text{train}}, x))\, p(f(x) \mid D^{\text{train}}, x) d(f(x))}_{H(\mathbb{P}[f(x)\in\cdot \mid D^{\text{train}},x], \mathbb{Q}[\cdot \mid D^{\text{train}},x])}\Bigg],
\end{aligned}$$

where H is the cross-entropy. So this is further equal to

$$\begin{aligned}
&\mathbb{E}_{D^{\text{train}},x}\left[d_{\text{KL}}\left(\mathbb{P}[f(x) \in \cdot \mid D^{\text{train}}, x] \;||\; \mathbb{Q}[\cdot \mid D^{\text{train}}, x]\right)\right] + \\
&\mathbb{E}_{D^{\text{train}},x}\left[H\left(\mathbb{P}[f(x) \in \cdot \mid D^{\text{train}}, x]\right)\right] = \\
&= \bar{d}_{\text{KL}} + C_{\mathbb{P}},
\end{aligned}$$

where $C_{\mathbb{P}} = \mathbb{E}_{D^{\text{train}},x}\left[H\left(\mathbb{P}[f(x) \in \cdot \mid D^{\text{train}}, x]\right)\right]$ only depends on $\mathbb{P}$ and not on $\mathbb{Q}$ or q. □

To apply Theorem B.7 to the metrics reported in Tables 2, 10 and 11 one has to apply Footnote 30 and has to set $q(\cdot \mid D^{\text{train}}, x)$ to be the density corresponding to the already calibrated uncertainty at x obtained from any method trained on D^{train}, i.e., for example let c be a fitted calibration parameter, and $\hat{f}$ and $\hat{\sigma}$ be the fitted model and model uncertainty prediction of NOMU then $q(\cdot \mid D^{\text{train}}, x) := \mathcal{N}(\cdot; \hat{f}(x), c \cdot \hat{\sigma}(x))$ for $x \in X$.[32] In our setting, we made sure that the correct calibration constant c is chosen in the limit $m_{\text{seeds}} \to \infty$, since we chose one fixed value for c per dimension and method and do not over-fit on specific seeds.

B.2.6. Solar Irradiance Time Series: Details

Configuration Details We use for all algorithms the same configuration as in the 1D and 2D toy regression setting (see Section 4.1.1 and Appendix B.2.2) except that we now train all algorithms for 2^{14} epochs and set as L2-regularization $\lambda = 10^{-19}$ for NOMU, $\lambda = 10^{-19}/n^{\text{train}}$ for DE, and $\lambda = (1-0.2)\cdot 10^{-19}/n^{\text{train}}$ for MCDO. For HDE, we accordingly draw the L2-regularization parameter log-uniform from $(10^{-19}\cdot 10^{-3}, 10^{-19}\cdot 10^{+3})$ (which affects the L2-regularization parameter of HDE* in the same way as in Appendix B.2.2).

Results Figure 13 visualizes the UBs from NOMU and the benchmark algorithms. As becomes evident, NOMU manages to best fit the training data[33], while keeping model uncertainty in-between training points. The corresponding metrics for this specific run are given in Table 12.

B.2.7. UCI Data Sets: Details

To get a feeling for how well NOMU (without data noise extension or hyperparameter tuning) performs in noisy regression with high-dimensional input, we test our algorithm on the popular UCI benchmark (Hernández-Lobato & Adams,

[30] We formulate the theorem for the hardest case of $n^{\text{test}} = 1$. The convergence of

$$\lim_{m_{\text{seeds}}\to\infty} \frac{1}{m_{\text{seeds}}} \sum_{j=1}^{m_{\text{seeds}}} \frac{1}{n^{\text{test}}} \sum_{i=1}^{n^{\text{test}}} -\log\left(q(f_j(x_{j,i}) \mid D_j^{\text{train}}, x_{j,i})\right) = \bar{d}_{\text{KL}} + C_{\mathbb{P}}$$

is obviously even faster for $n^{\text{test}} > 1$. For the proof it is only important that $m_{\text{seeds}} \to \infty$.

[31] $\mathbb{P}$ does not depend on the chosen method. Only $\mathbb{Q}$ (and accordingly q) differs among the methods. Thus, $C_{\mathbb{P}}$ does not change the ranking amongst different methods.

[32] In theory, more general posteriors than Gaussians could be used too, but within this paper we always assumed Gaussian marginals of the posterior as all the considered benchmarks also output Gaussian distributed approximations of the marginal posteriors.

[33] Gal & Ghahramani (2016) consider the same data set to compare UBs of MC dropout and GPs. In this work however, NNs are trained for 10^6 epochs possibly explaining why MC dropout more nicely fits the training data in their case.

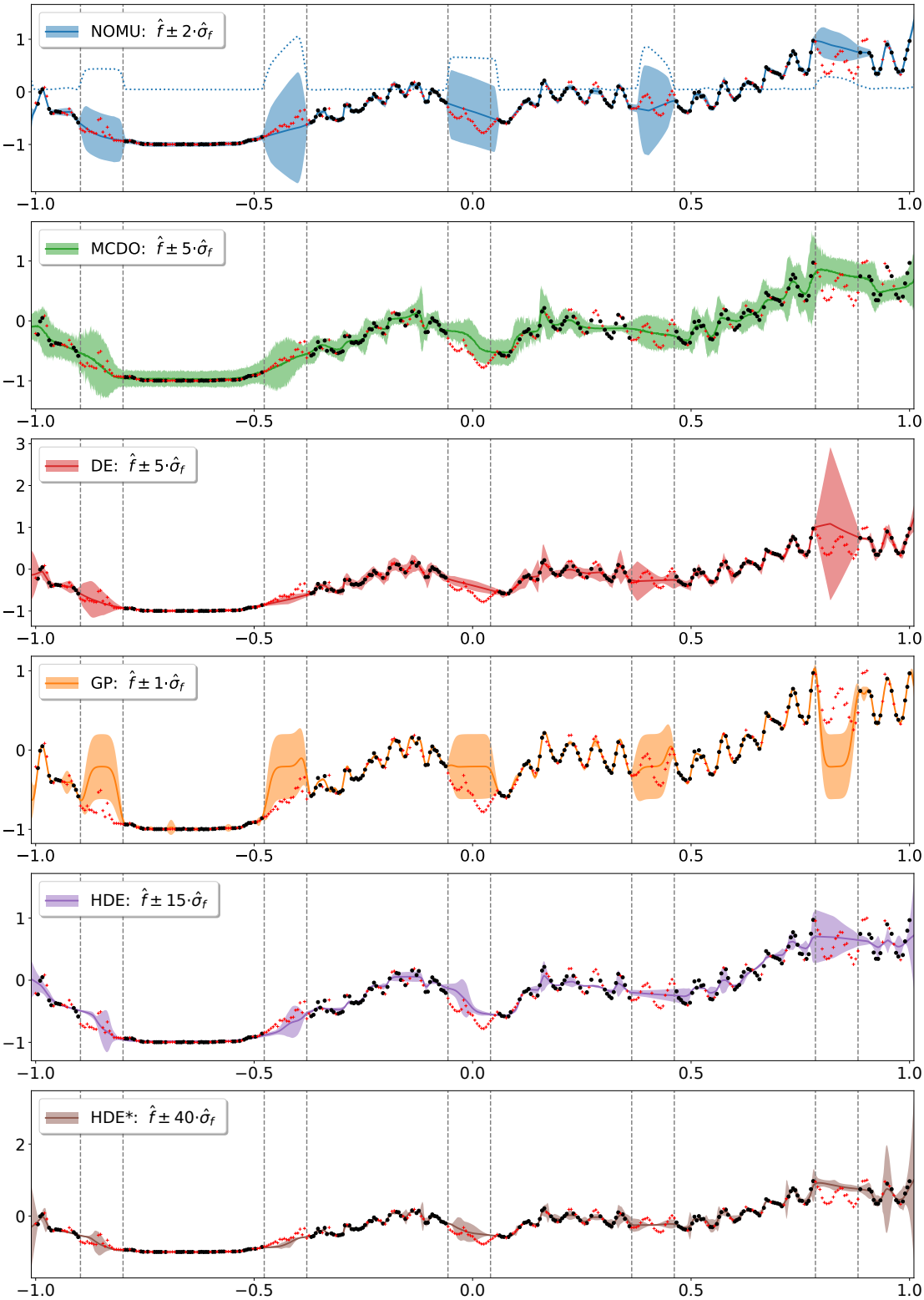


Figure 13. Visualization of each algorithm's model prediction (solid lines) and UBs (shaded areas) on the solar irradiance data set. Training and test points are shown as black dots and red crosses, respectively. We present UBs obtained by NOMU (c=2) compared to the benchmarks MC dropout (MCDO) (c=5), deep ensembles (DE) (c=5), Gaussian process (GP) (c=1) and hyper deep ensembles (HDE) (c=15) and (HDE*) (c=40). Additionally, NOMU's estimated and scaled model uncertainty $2\hat{\sigma}_f$ is shown as a dotted line.

Table 12. Metrics for solar irradiance time series.

METHOD	AUC↓	NLL_{MIN}↓
NOMU	0.32	-1.44
GPR	0.46	-1.24
MCDO	0.30	-1.11
DE	0.35	-1.28
HDE	0.47	-0.98
HDE*	0.59	-1.08

2015) and the UCI gap data sets introduced in (Foong et al., 2019).

The UCI gap data sets were designed to capture whether an algorithm's uncertainty estimate increases in between separated clusters of observations where epistemic uncertainty should be higher compared to regions with many data points. This is also required by D3. However, in (Foong et al., 2019) only those data points that had been removed to create gaps in the training data were used as test data for calculating NLL. Thus, NLL on UCI gap data fails by construction to capture the uncertainty quality outside of the gap: First, D2 is not properly measured by this evaluation, since there are not many test-points in the test data-set that are close to training points. Second, also D3 is not properly measured, since the smaller gaps in between data points outside of the gap are not part of the test data-set. Note that D3 also concerns these kinds of gaps. To provide better evaluation of uncertainty, future work should focus on mixed test data-sets (moving some input points outside of the gap from the training set to the test set) similar to our train/test split from the experimental setting in Appendix B.2.6.

Nonetheless, we test NOMU on the UCI and UCI gap data sets using the same experiment setup as in the respective works.

NOMU Setup In line with the literature, NOMU's main and side architectures were chosen as a single-hidden-layer NN with 50 hidden nodes (except for the large protein data set, where the number of hidden nodes was 100). NOMU was trained for 400 (UCI) respectively 40 (UCI gap) epochs, with L2-regularization $1e^{-09}$ and $1e^{-04}$ on the main- and side architectures respectively, using a batch size of 100 in Adam stochastic gradient descent with learning rate 0.01. NOMU used $\ell = 100$ artificial data points randomly sampled on each batch and was refit on validation data after the constant c had been calibrated on these. For the standard UCI data sets, we used the same loss parameters as in the remaining regression experiments, namely $\pi_{\text{exp}} = 0.01$, $\pi_{\text{sqr}} = 0.1$, and $c_{\text{exp}} = 30$. For the UCI gap data set, where uncertainty is only evaluated in gap regions, we chose $\pi_{\text{exp}} = 0.1$ and $\pi_{\text{sqr}} = 0.01$, relaxing the requirement of small uncertainty at observed data points and strengthening the pull upward on our raw uncertainty output.

Table 13. UCI-GAP average NLL and a 95% normal-CI.

DATASET	NOMU	LL	NLM-HPO	NLM
BOSTON	2.80 ±0.11	2.79 ±0.09	2.81 ±0.15	3.60 ±0.21
CONCRETE	3.38 ±0.08	3.53 ±0.13	3.72 ±0.11	3.89 ±0.21
ENERGY	3.71 ±1.62	6.49 ±5.50	3.78 ±2.27	3.40 ±1.99
KIN8NM	-0.97 ±0.02	-1.14 ±0.03	-1.06 ±0.03	-1.04 ±0.03
NAVAL	31.81 ±28.0	15.66 ±8.34	1.17 ±1.56	1.48 ±1.64
CCPP	2.91 ±0.04	2.89 ±0.03	2.96 ±0.09	2.90 ±0.05
PROTEIN	3.21 ±0.10	3.09 ±0.05	3.22 ±0.09	3.35 ±0.11
WINE	0.99 ±0.02	0.96 ±0.01	0.98 ±0.01	1.75 ±0.07
YACHT	2.15 ±0.30	1.33 ±0.29	1.72 ±0.33	1.44 ±0.21

Moreover, for numerical stability we use the following slight adaptation of the NOMU loss from Definition 3.3:

$$L^{\pi}_{\text{stable}}(\mathcal{NN}_\theta) := \underbrace{\sum_{i=1}^{n^{\text{train}}} (\hat{f}(x_i^{\text{train}}) - y_i^{\text{train}})^2}_{(a)} + \pi_{\text{sqr}} \cdot \underbrace{\sum_{i=1}^{n^{\text{train}}} \rho_{\text{stable}}\left(\hat{r}_f(x_i^{\text{train}})\right)}_{\text{stable version of } (b)} + \pi_{\text{exp}} \cdot \underbrace{\frac{1}{\lambda_d(X)} \int_X u_{\text{stable}}\left(\hat{r}_f(x)\right) dx}_{\text{stable version of } (c)}, \tag{53}$$

where

$$\rho_{\text{stable}}(r) = \begin{cases} r^2 & , |r| \leq 1 \\ 2|r| - 1 & , |r| > 1 \end{cases} \tag{54}$$

is the Huber-loss and

$$u_{\text{stable}}(r) = \begin{cases} u(r) = e^{-c_{\text{exp}} \cdot r} & , r \geq 0 \\ -c_{\text{exp}} \cdot r & , r < 0. \end{cases} \tag{55}$$

This stable loss L^{π}_{stable} behaves exactly the same as the standard NOMU-loss L^{π} as long as the outputs of the NN stay in a reasonable range. Only if the gradient descent gets unstable, such that the NN outputs very extreme value, this stabilized loss assures a bounded gradient of the loss L^{π}_{stable} with respect to the outputs of the NN $\mathcal{NN}_\theta$.

Results UCI Table 3 in the main paper reports NLL on test data averaged over 20 runs as in (Hernández-Lobato & Adams, 2015) for the UCI data set. It includes the following benchmark models:

- NLM-HPO and NLM correspond to BN(BO)-1 NL in Section D.1 respectively Section D.2. of (Ober & Rasmussen, 2019),
- LL corresponds to LL 1HL TANH in (Foong et al., 2019),
- MCDO2 represents Dropout (Convergence) in (Mukhoti et al., 2018),
- MCDO corresponds to Dropout in (Gal & Ghahramani, 2016) and DE to Deep Ensembles in (Lakshminarayanan et al., 2017).

We used the standard deviations that were reported in these respective works to derive 95%-normal CIs.

Results UCI Gap Table 13 lists Gaussian 95%-confidence intervals for NLL on the UCI gap data sets for NOMU as well as the values reported for benchmarks NLM, NLM-HPO and LL (Linearized Laplace) as listed above. These values result from different runs where in each run the gap were introduced in a different input dimension (see Foong et al. (2019)). We see that NOMU performs comparably, and confirm the observation that LL tends to best capture uncertainty in the large gaps artificially introduced in the training data.

B.3. Bayesian Optimization

B.3.1. BO: Configuration Details

In the following, we describe the detailed hyperparameter setup of all algorithms.

NOMU Setup For NOMU, we set $\pi_{\text{sqr}} = 1$, $\ell_{\text{min}} = 1\text{e}{-6}$ and use $l = 500$ artificial input points for 5D, 10D and 20D. Otherwise we use the exact same hyperparameters as in 1D and 2D regression (see in Section 4.1 the paragraph **Algorithm setup**).

Deep Ensembles Setup For deep ensembles, we use the exact same hyperparameters as in 1D and 2D regression (see Appendix B.2.2).

MC Dropout Setup For MC dropout, due to computational constraints, we were only able to use 10 stochastic forward passes (instead of 100 in the regression setting) However, this still results in an increase in compute time by a factor 10 of every single acquisition function evaluation compared to NOMU. Otherwise, we use the exact same hyperparameters as in 1D and 2D regression (see Appendix B.2.2) .

Gaussian Process Setup For Gaussian processes (pGP and GP), we use the exact same setup as in 1D and 2D regression (see Appendix B.2.2).

Hyper Deep Ensembles Setup For hyper deep ensembles, due to computational constraints, we were only able to use $\kappa = 20$ (instead of $\kappa = 50$ in the regression setting). However, this still results in an increase of training time by a factor of 5 in each single BO step compared to NOMU. Otherwise we use the exact same hyperparameters as in the 1D and 2D regression setting (see Appendix B.2.2).

Acquisition Function Optimization For each algorithm and each BO step, we determine the next BO input point by maximizing the corresponding upper-bound acquisition function, i.e., $(\hat{f}(x) + c\,\hat{\sigma}_f(x))$. We maximize this upper-bound acquisition function using the established DIRECT (Dividing Rectangles) algorithm.

Gaussian BNN Test Functions Note that for the Gaussian BNN test functions in BO (see Table 4 and Table 14, respectively) we do not have access to the ground truth global maximum. Thus, to determine the final regret, we use the established DIRECT (Dividing Rectangles) algorithm. With DIRECT, we calculate a point x^{direct} s.t. $f(x^{\text{direct}}) \approx \max_{x \in X} f(x)$ and report $|f(x^{\text{direct}}) - \max_{i \in \{1,\dots,72\}} f(x_i)| / f(x^{\text{direct}})$ in Table 4 and Table 14, respectively.

B.3.2. BO: Calibration

MW Scaling In general, having a good understanding of the prior scale is very hard. Nevertheless, often it is easier to have at least some intuition in which order of magnitude the posterior UBs should be on average. When function values approximately range in $[-1, 1]$, it is reasonable to require that after observing 8 initial points the mean width (MW) of the UBs lies within the order of magnitudes 0.05 and 0.5. Hence our motivation for choosing the calibration parameter c accordingly. An advantage of such a calibration is that it can be applied to every method equally, whereas there is in general no clear notion of setting the prior scale of two different methods (e.g., MC dropout and deep ensembles) to the same value. Note, that we only use MW to calibrate c directly after mean and variance predictions were fit based on the 8 initial data points. So MW is not fixed when further data points are observed in subsequent steps of the BO.

Dynamic c The initial choice of c can still be corrected in each BO step. Certainly, in the noiseless case it does not make sense to pick an input point $x_{i'}$ that is identical to an already observed input point $x_i, i < i'$, where nothing new is to be learned. Therefore, we want our NN-agent to get "bored" if its acquisition function optimization would suggest to pick an input point $x_{i'} \approx x_i, i < i'$. The idea of Dynamic c is to encourage the agent, in case it was "bored", to become more "curious" to explore something new instead of picking a "boring" input point. This can be achieved by iteratively increasing c until the acquisition function optimization suggests an input point $x_{i'} \not\approx x_i, \forall i < i'$. We then only apply the expensive function evaluation for $x_{i'} \not\approx x_i, \forall i < i'$ that we obtained after increasing c enough to not "bore" the NN. However, towards the last BO-steps, if we already know approximately where a global optimum is and we only want to fix the last digits of the optimizer, we have to evaluate the function closer to already observed points. In contrast, a very "young" NN (i.e., a network that

Table 14. Results for 15 different BO tasks. Shown are the average final regrets and a 95% normal-CI per dimension and for each individual function over 100 (5D) and 50 (10D and 20D) runs for an initial mean width (MW) of 0.05 and 0.5, respectively. Winners are marked in grey.

Function	NOMU MW 0.05	NOMU MW 0.5	GP MW 0.05	GP MW 0.5	MCDO MW 0.05	MCDO MW 0.5	DE MW 0.05	DE MW 0.5	HDE MW 0.05	HDE MW 0.5	pGP MW	RAND MW
Levy5D	2.12e−3 ± 5.95e−4	1.52e−3 ± 4.34e−4	1.10e−3 ± 1.68e−4	8.84e−3 ± 1.15e−3	1.98e−2 ± 3.59e−3	1.27e−2 ± 2.57e−3	7.09e−3 ± 3.76e−3	9.09e−2 ± 9.16e−3	4.48e−3 ± 1.25e−3	3.76e−3 ± 1.22e−3	6.25e−3 ± 5.26e−4	5.44e−2 ± 4.22e−3
Rosenbrock5D	1.75e−4 ± 2.76e−5	3.57e−4 ± 6.39e−5	3.62e−4 ± 6.52e−5	8.44e−4 ± 1.62e−4	2.83e−4 ± 4.64e−5	7.35e−4 ± 1.08e−4	7.93e−4 ± 7.74e−4	5.96e−3 ± 1.78e−3	1.11e−3 ± 7.25e−4	5.02e−4 ± 1.04e−4	7.56e−4 ± 8.13e−5	4.73e−3 ± 7.04e−4
G-Function5D	1.12e−1 ± 1.99e−2	1.06e−1 ± 1.97e−2	1.68e−1 ± 1.62e−2	2.60e−1 ± 1.72e−2	2.66e−2 ± 8.57e−3	7.80e−2 ± 8.82e−3	1.88e−1 ± 2.08e−2	2.03e−1 ± 2.20e−2	1.35e−1 ± 2.46e−2	1.37e−1 ± 2.59e−2	1.78e−1 ± 1.22e−2	3.63e−1 ± 1.09e−2
Perm5D	4.06e−4 ± 1.70e−4	2.58e−4 ± 1.13e−4	7.76e−5 ± 3.19e−5	1.80e−3 ± 4.44e−4	7.79e−5 ± 4.19e−5	7.11e−5 ± 1.45e−5	6.96e−4 ± 2.20e−4	1.73e−3 ± 4.55e−4	2.74e−3 ± 5.20e−4	2.50e−3 ± 5.58e−4	2.06e−4 ± 7.03e−5	4.62e−4 ± 1.20e−4
BNN5D	6.60e−2 ± 4.03e−2	3.77e−2 ± 1.89e−2	1.16e−1 ± 8.63e−2	4.53e−2 ± 3.04e−2	2.37e−1 ± 8.04e−2	2.04e−1 ± 9.29e−2	6.32e−2 ± 5.28e−2	7.92e−2 ± 5.34e−2	2.70e−1 ± 8.78e−2	2.36e−1 ± 8.27e−2	2.23e−2 ± 6.42e−3	5.42e−1 ± 9.21e−2
Average Regret 5D	3.62e−2 ± 8.99e−3	2.91e−2 ± 5.46e−3	5.71e−2 ± 1.76e−2	6.33e−2 ± 6.99e−3	5.67e−2 ± 1.62e−2	5.92e−2 ± 1.87e−2	5.19e−2 ± 1.14e−2	7.62e−2 ± 1.17e−2	8.26e−2 ± 1.82e−2	7.59e−2 ± 1.73e−2	4.16e−2 ± 2.76e−3	1.93e−1 ± 1.86e−2
Levy10D	6.27e−3 ± 2.22e−3	6.27e−3 ± 2.22e−3	1.04e−2 ± 2.10e−3	1.04e−2 ± 2.10e−3	2.16e−2 ± 3.71e−3	2.17e−2 ± 5.30e−3	8.65e−2 ± 2.01e−2	1.46e−1 ± 1.41e−2	6.21e−3 ± 2.48e−3	5.81e−3 ± 1.99e−3	6.16e−3 ± 8.94e−4	1.06e−1 ± 7.64e−3
Rosenbrock10D	2.34e−3 ± 1.43e−3	2.01e−3 ± 6.91e−4	9.14e−4 ± 1.25e−4	5.67e−3 ± 1.60e−3	2.25e−3 ± 2.58e−4	4.96e−3 ± 4.40e−4	1.42e−2 ± 4.43e−3	7.65e−2 ± 1.03e−2	5.10e−3 ± 1.44e−3	5.00e−3 ± 1.70e−3	3.09e−3 ± 5.77e−4	2.82e−2 ± 3.92e−3
G-Function10D	2.92e−1 ± 3.85e−2	3.53e−1 ± 2.74e−2	3.79e−1 ± 2.24e−2	4.21e−1 ± 2.35e−2	1.79e−1 ± 2.54e−2	4.15e−1 ± 1.12e−2	3.24e−1 ± 2.28e−2	3.33e−1 ± 2.06e−2	2.53e−1 ± 3.76e−2	2.55e−1 ± 3.72e−2	3.80e−1 ± 1.34e−2	4.50e−1 ± 6.48e−3
Perm10D	3.74e−4 ± 1.26e−4	4.27e−4 ± 1.34e−4	3.59e−4 ± 2.13e−4	6.52e−4 ± 2.04e−4	3.37e−4 ± 8.49e−5	5.53e−4 ± 1.33e−4	1.01e−3 ± 3.10e−4	1.48e−3 ± 3.92e−4	4.07e−4 ± 1.21e−4	4.08e−4 ± 1.48e−4	5.57e−4 ± 1.43e−4	1.81e−4 ± 5.44e−5
BNN10D	1.23e−1 ± 3.76e−2	1.00e−1 ± 3.43e−2	1.96e−1 ± 5.16e−2	1.68e−1 ± 4.09e−2	1.45e−1 ± 3.60e−2	1.61e−1 ± 2.86e−2	1.51e−1 ± 4.27e−2	1.13e−1 ± 4.00e−2	2.58e−1 ± 4.96e−2	2.02e−1 ± 3.79e−2	8.65e−2 ± 2.29e−2	5.92e−1 ± 5.34e−2
Average Regret 10D	8.48e−2 ± 1.08e−2	9.24e−2 ± 8.79e−3	1.17e−1 ± 1.13e−2	1.21e−1 ± 9.45e−3	6.97e−2 ± 8.84e−3	1.21e−1 ± 6.24e−3	1.15e−1 ± 1.05e−2	1.34e−1 ± 9.65e−3	1.05e−1 ± 1.25e−2	9.36e−2 ± 1.06e−2	9.52e−2 ± 5.30e−3	2.35e−1 ± 1.09e−2
Levy20D	1.51e−2 ± 1.69e−3	1.40e−2 ± 1.63e−3	1.98e−2 ± 8.61e−3	2.64e−2 ± 1.12e−2	4.27e−2 ± 4.16e−3	6.91e−2 ± 9.00e−3	1.88e−1 ± 1.15e−2	2.01e−1 ± 1.07e−2	1.13e−2 ± 4.60e−3	1.61e−2 ± 5.47e−3	1.98e−2 ± 7.88e−3	1.48e−1 ± 6.58e−3
Rosenbrock20D	3.47e−2 ± 7.08e−3	6.03e−3 ± 9.93e−4	8.94e−3 ± 4.00e−3	1.91e−2 ± 4.04e−3	7.41e−3 ± 1.14e−3	7.61e−3 ± 2.15e−3	6.46e−2 ± 1.22e−2	1.41e−1 ± 1.29e−2	2.96e−3 ± 1.36e−4	4.27e−3 ± 1.33e−3	1.09e−2 ± 1.52e−3	7.80e−2 ± 8.38e−3
G-Function20D	4.16e−1 ± 8.99e−3	4.18e−1 ± 1.70e−2	4.59e−1 ± 6.77e−3	4.86e−1 ± 3.95e−3	4.70e−1 ± 4.43e−3	4.92e−1 ± 1.58e−3	4.18e−1 ± 1.33e−2	4.02e−1 ± 1.61e−2	3.79e−1 ± 2.97e−2	3.73e−1 ± 2.56e−2	4.47e−1 ± 9.37e−3	4.86e−1 ± 2.44e−3
Perm20D	7.45e−5 ± 2.09e−5	7.46e−5 ± 2.50e−5	2.20e−4 ± 8.98e−5	1.57e−4 ± 7.50e−5	9.98e−5 ± 3.19e−5	2.03e−4 ± 8.86e−5	3.54e−5 ± 6.52e−6	1.85e−4 ± 6.90e−5	7.98e−5 ± 2.19e−5	9.28e−5 ± 3.89e−5	9.44e−5 ± 3.25e−5	1.90e−5 ± 4.67e−6
BNN20D	1.58e−1 ± 4.12e−2	1.23e−1 ± 3.21e−2	1.84e−1 ± 3.46e−2	1.42e−1 ± 2.70e−2	1.77e−1 ± 3.46e−2	2.45e−1 ± 4.41e−2	1.89e−1 ± 3.98e−2	3.05e−1 ± 7.02e−2	2.93e−1 ± 5.72e−2	3.05e−1 ± 5.01e−2	1.11e−1 ± 2.06e−2	6.85e−1 ± 3.29e−2
Average Regret 20D	1.25e−1 ± 8.57e−3	1.12e−1 ± 7.27e−3	1.34e−1 ± 7.31e−3	1.35e−1 ± 5.95e−3	1.39e−1 ± 7.03e−3	1.63e−1 ± 9.02e−3	1.72e−1 ± 9.03e−3	2.10e−1 ± 1.48e−2	1.37e−1 ± 1.29e−2	1.40e−1 ± 1.13e−2	1.18e−1 ± 4.81e−3	2.80e−1 ± 6.94e−3

has been trained on few training points) should get "bored" much more easily, since it is not sensible to exploit a given local optimum up to the last digits, when there is plenty of time to reach other, possibly better local optima.

Thus, it makes sense to first explore on a large scale where the good local optima are approximately, then to find out which of them are the best ones and finally to exploit the best one in greater detail at the very end.

Therefore, we want the threshold δ_i, that determines if $x_{i'} \overset{\delta_{i'}}{\approx} x_i \iff \|x_{i'} - x_i\|_2 \leq \delta_{i'}$ to decrease in each BO step. In our experiment, we choose an exponential decay of

$$\delta_i = \delta_{n^{\text{start}}} \cdot \left(\frac{\delta_{n^{\text{end}}}}{\delta_{n^{\text{start}}}}\right)^{(i-n^{\text{start}})/(n^{\text{end}}-n^{\text{start}})}, \tag{56}$$

with $\delta_{n^{\text{start}}} = \frac{1}{16}$ and $\delta_{n^{\text{end}}} = 0.01$.

Concretely, we only evaluate f at $x_{i'}$ if $\|x_{i'} - x_i\|_2 > \delta_{i'} \forall i < i'$ is fulfilled. Otherwise we double c until it is fulfilled (With larger c more emphasis is put on exploration, so there is a tendency that $x_{i'}$ will be further away from the observed input points the larger we pick c, if D3 (Out-of-Sample) is fulfilled). After doubling c 15 times without success, we evaluate f at $x_{i'}$ no matter how close it is to the already observed input points (for methods that have severe troubles to fulfill D3 (Out-of-Sample), such as MCDO, even doubling c infinite times would not help if the maximal uncertainty is within an $\delta_{i'}$-ball around an already observed input point).

B.3.3. BO: Detailed Results

In Table 14, we present the mean final regrets, which correspond to the ranks shown in Table 4 in the main paper.

B.3.4. BO: Regret Plots

In Figures 14–16, we present the regret plots for each test function and both MW values.

C. Extensions

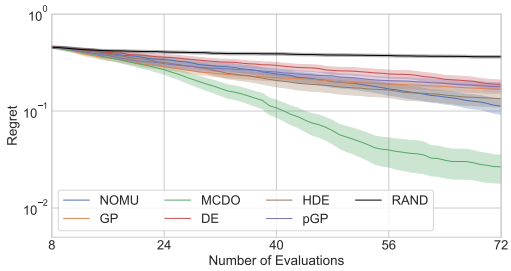
C.1. Incorporating Data Noise

We now discuss one way to incorporate data noise in NOMU. If σ_{n} is unknown, one option to learn it, is to add another output $\hat{\sigma}_{\text{n}}$ to its architecture and change the loss function to

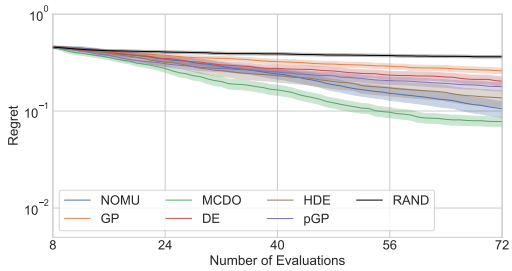
$$\begin{aligned} L^{\pi}(\mathcal{NN}_\theta) := &\sum_{i=1}^{n^{\text{train}}} \left(\frac{\left(\hat{f}(x_i^{\text{train}}) - y_i^{\text{train}}\right)^2}{2\left(\hat{\sigma}_{\text{n}}(x_i^{\text{train}})\right)^2} + \ln\left(\hat{\sigma}_{\text{n}}(x_i^{\text{train}})\right) \right) \\ &+ \pi_{\text{sqr}} \cdot \sum_{i=1}^{n^{\text{train}}} \frac{\left(\hat{r}_f(x_i^{\text{train}})\right)^2}{2\left(\hat{\sigma}_{\text{n}}(x_i^{\text{train}})\right)^2} \\ &+ \pi_{\text{exp}} \cdot \frac{1}{\lambda_d(X)} \int_X e^{-c_{\text{exp}} \cdot \hat{r}_f(x)}\, dx, \end{aligned} \tag{57}$$

in the case of Gaussian data noise uncertainty. In this case we recommend to first train the model prediction $\hat{f}$ and the data noise $\hat{\sigma}_{\text{n}}$ only and then freeze their parameters and train the $\hat{r}_f$-network for capturing model uncertainty $\hat{\sigma}_f$. Note that in the NOMU loss (4) we implicitly assumed a constant very small and negligible data noise σ_{n}^2, absorbed as a factor into the hyperparameter π_{exp} and the regularization factor λ. Thus, when using the loss (57) instead of the NOMU loss (4), π_{exp} and λ need to be chosen significantly larger.

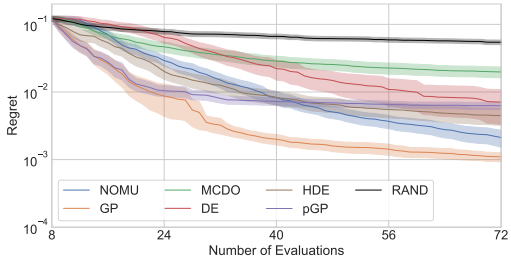
In the case of known (heteroscedastic) data noise $\sigma_{\text{n}}(x)$, (57) can be simplified, replacing $\hat{\sigma}_{\text{n}}$ by σ_{n} and dropping the term $\ln(\sigma_{\text{n}})$ (in this case, one can then also drop the $\hat{\sigma}_{\text{n}}$-output of the NOMU architecture).



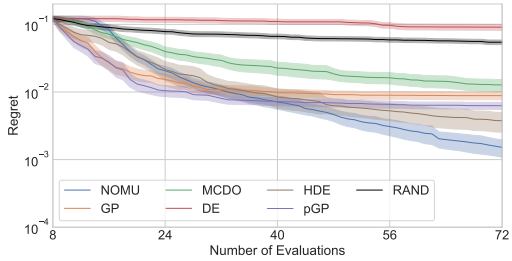
(a) Regret plot G-Function, 0.05 MW



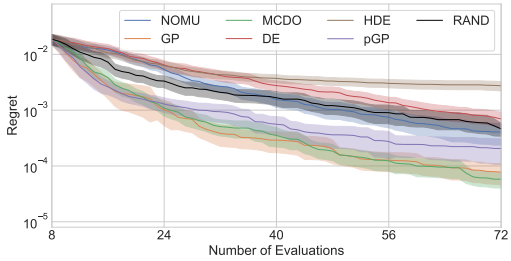
(b) Regret plot G-Function, 0.5 MW



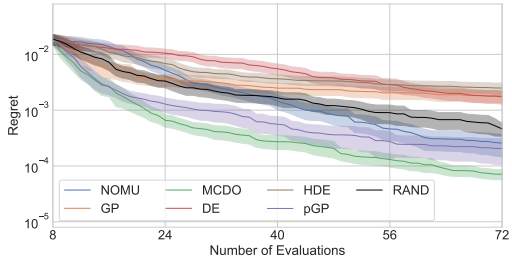
(c) Regret plot Levy, 0.05 MW



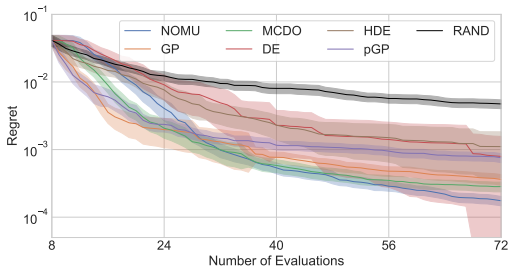
(d) Regret plot Levy, 0.5 MW



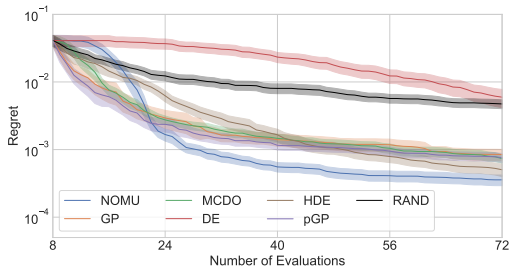
(e) Regret plot Perm, 0.05 MW



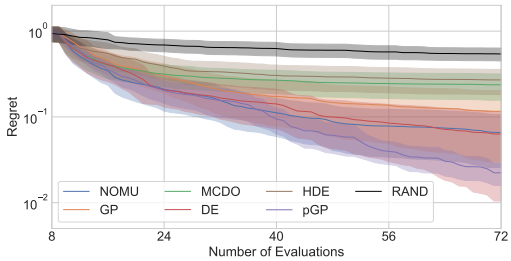
(f) Regret plot Perm, 0.5 MW



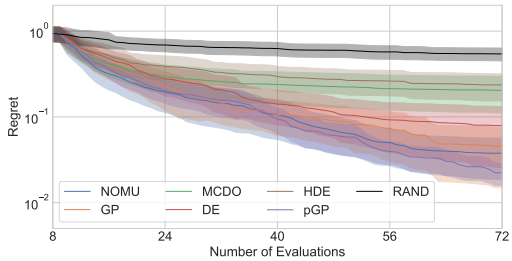
(g) Regret plot Rosenbrock, 0.05 MW



(h) Regret plot Rosenbrock, 0.5 MW

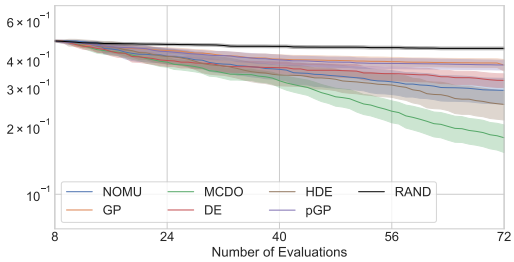


(i) Regret plot BNN, 0.05 MW

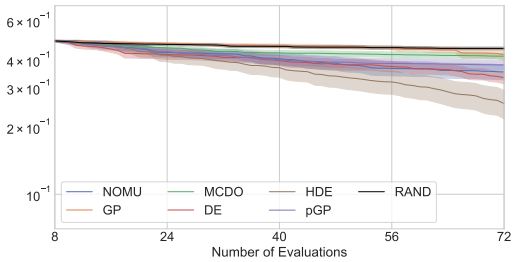


(j) Regret plot BNN, 0.5 MW

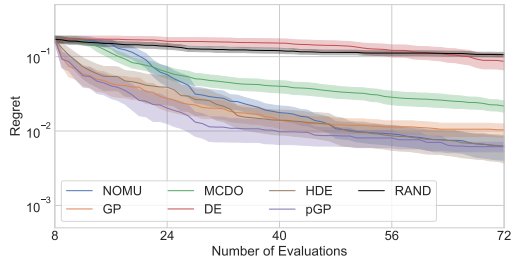
Figure 14. Regret plots for all 5D test functions and MWs of 0.05 and 0.5, respectively. We show regrets averaged over 100 runs (solid lines) with 95% CIs.



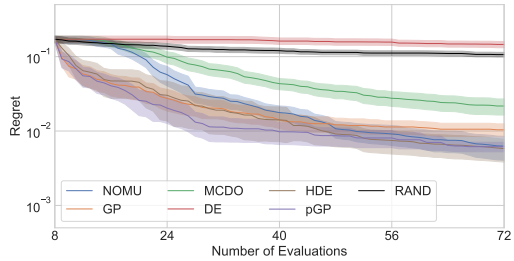
(a) Regret plot G-Function, 0.05 MW



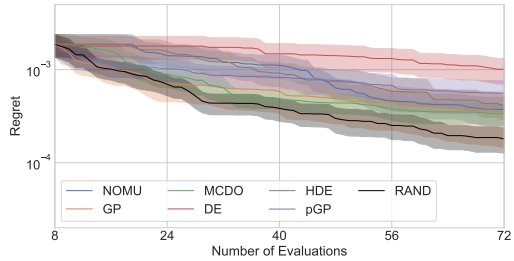
(b) Regret plot G-Function, 0.5 MW



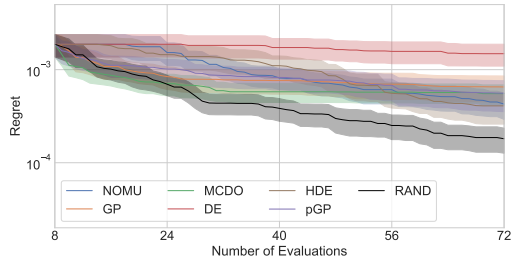
(c) Regret plot Levy, 0.05 MW



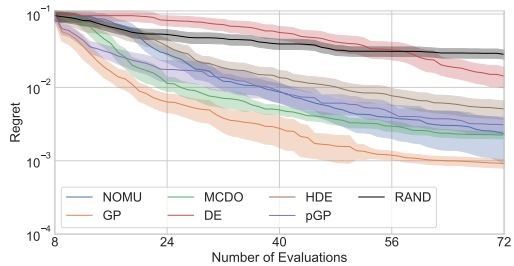
(d) Regret plot Levy, 0.5 MW



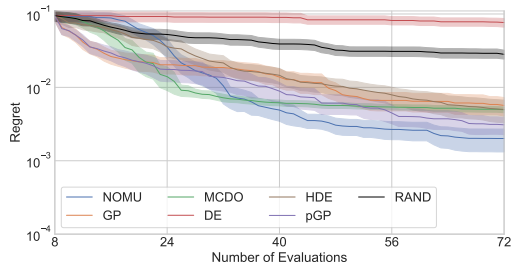
(e) Regret plot Perm, 0.05 MW



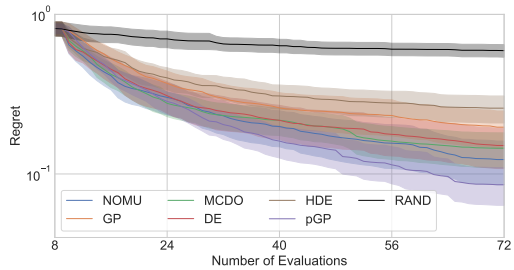
(f) Regret plot Perm, 0.5 MW



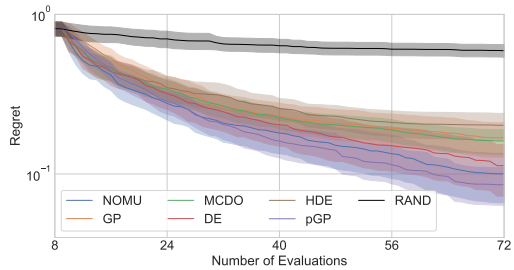
(g) Regret plot Rosenbrock, 0.05 MW



(h) Regret plot Rosenbrock, 0.5 MW

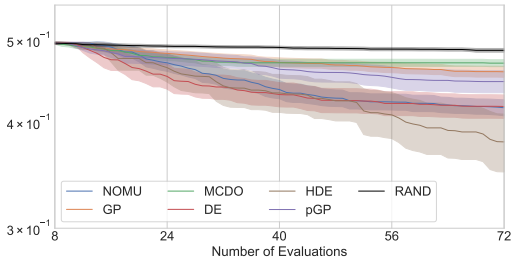


(i) Regret plot BNN, 0.05 MW

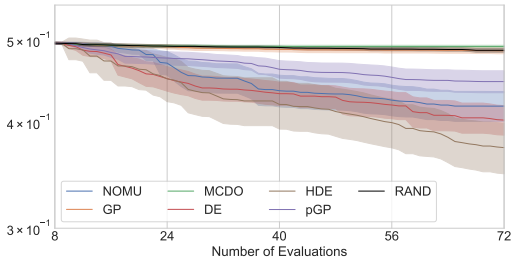


(j) Regret plot BNN, 0.5 MW

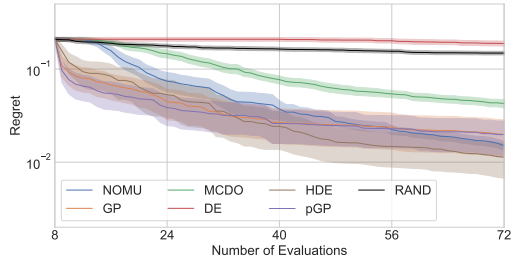
Figure 15. Regret plots for all 10D test functions and MWs of 0.05 and 0.5, respectively. We show regrets averaged over 100 runs (solid lines) with 95% CIs.



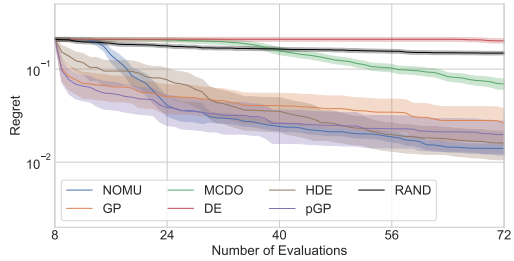
(a) Regret plot G-Function, 0.05 MW



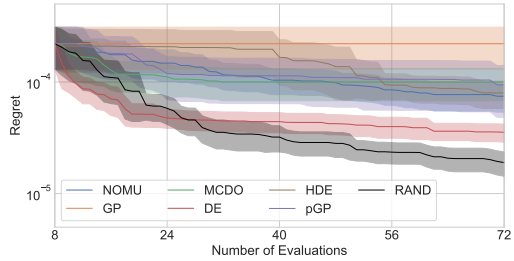
(b) Regret plot G-Function, 0.5 MW



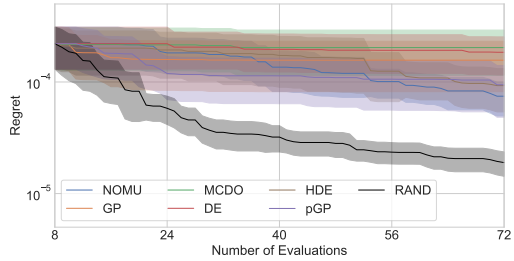
(c) Regret plot Levy, 0.05 MW



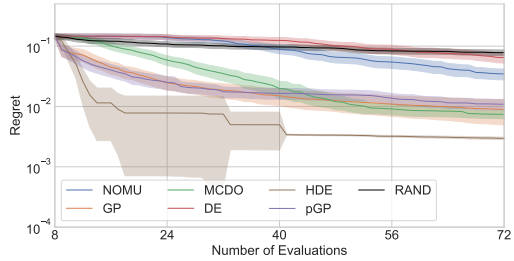
(d) Regret plot Levy, 0.5 MW



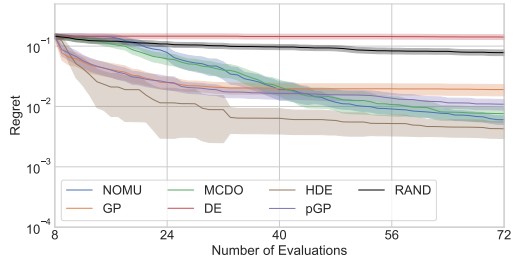
(e) Regret plot Perm, 0.05 MW



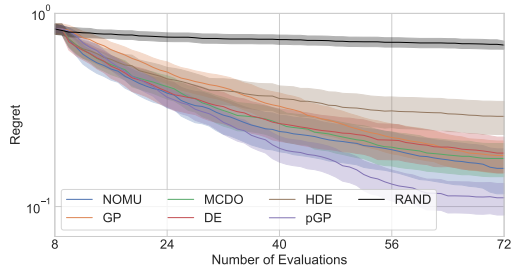
(f) Regret plot Perm, 0.5 MW



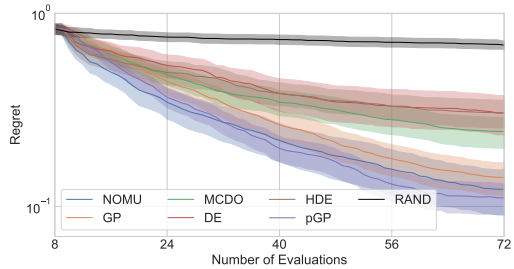
(g) Regret plot Rosenbrock, 0.05 MW



(h) Regret plot Rosenbrock, 0.5 MW



(i) Regret plot BNN, 0.05 MW



(j) Regret plot BNN, 0.5 MW

Figure 16. Regret plots for all 20D test functions and MWs of 0.05 and 0.5, respectively. We show regrets averaged over 100 runs (solid lines) with 95% CIs.

Predictive UBs are then obtained as

$$\left(\hat{f}(x) \mp \sqrt{c_1\, \hat{\sigma}_f^2(x) + c_2\, \hat{\sigma}_{\mathrm{n}}^2(x)}\right) \tag{58}$$

with suitable calibration parameters $c_1, c_2 \in \mathbb{R}_{\geq 0}$.

In the case of known normally distributed data noise (and under the assumption that the posterior of each $f(x)$ is Gaussian), it is sufficient to calibrate one calibration parameter $\tilde{c}$ to obtain approximate α predictive bounds

$$\left(\hat{f}(x) \mp \Phi^{-1}(1 - \frac{1-\alpha}{2})\sqrt{\tilde{c}\, \hat{\sigma}_f^2(x) + \sigma_{\mathrm{n}}^2(x)}\right), \tag{59}$$

where $\tilde{c}$ relates to the typically unknown prior scale.

C.2. NOMU for Upwards and Downwards Directed Hypothesis Classes

As mentioned in Appendix A.1 and discussed in more detail in Appendix A.3, often the set $\mathcal{H}_{D^{\text{train}}}$ is not upwards directed for typical NN-architectures and Equation (7) of Theorem A.1 is not fulfilled in general. Therefore, we carefully designed our NOMU algorithm to be able to cope with settings where the set $\mathcal{H}_{D^{\text{train}}}$ is not upwards and/or downwards directed. The downwards directed property is defined analogously as follows:

Assumption 2 (DOWNWARDS DIRECTED) *For every $f_1, f_2 \in \mathcal{H}_{D^{train}}$ there exists an $f \in \mathcal{H}_{D^{train}}$ such that $f(x) \leq \min(f_1(x), f_2(x))$ for all $x \in X$.*

However, in the following, we discuss a modification of NOMU, which is specifically designed for the case if $\mathcal{H}_{D^{\text{train}}}$ is indeed upwards and/or downwards directed. In this case, by Theorem A.1, we can directly solve

$$\underset{h \in \mathcal{H}_{D^{\text{train}}}}{\arg\max} \int_X u(h(x) - \hat{f}(x))\, dx \tag{60}$$

to obtain an upper UB and/or

$$\underset{h \in \mathcal{H}_{D^{\text{train}}}}{\arg\min} \int_X u(h(x) - \hat{f}(x))\, dx \tag{61}$$

to obtain a lower UB, without the need for any modifications as used in the proposed NOMU algorithm (we do not need the dashed connections in the architecture from Figure 2, we do not need a specific choice of u and we do not need to introduce $\hat{r}_f$ and the final activation function φ). The UBs obtained in this way *exactly* coincide then with the pointwise upper and lower UBs defined in (5), respectively.

Moreover in this case, $\hat{f}$ can be even removed from Equations (60) and (61) (as can be seen from the proof of Theorem A.4). Thus, in the following loss formulation, we will remove $\hat{f}$ in the respective term $(\tilde{c})$.

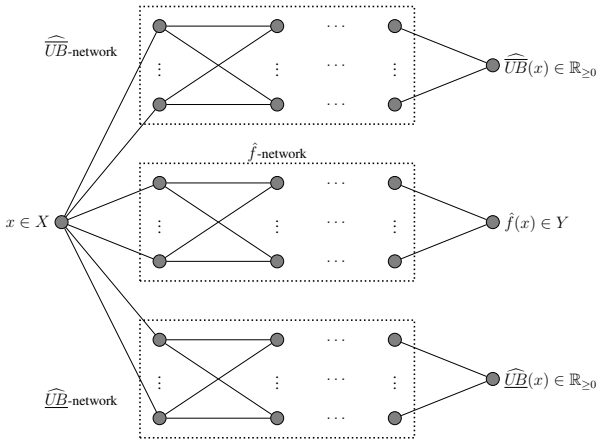


Figure 17. $\widetilde{\mathcal{NN}}_\theta$: a modification of NOMU's original network architecture for upwards and downwards directed hypothesis classes.

C.2.1. THE ARCHITECTURE

Under Assumption 1 (upwards directed) *and* Assumption 2 (downwards directed), we propose an architecture $\widetilde{\mathcal{NN}}_\theta$ consisting of three sub-networks with three outputs: the model prediction $\hat{f}$, the estimated lower UB $\widehat{\underline{UB}}$ and the estimated upper UB $\widehat{\overline{UB}}$. In Figure 17, we provide a schematic representation of $\widetilde{\mathcal{NN}}_\theta$.

C.2.2. THE LOSS FUNCTION

From Equations (60) and (61) we can then directly formulate the following modified NOMU loss function $\tilde{L}^\pi$.

Definition C.1 (NOMU LOSS UPWARDS AND DOWNWARDS DIRECTED) *Let $\pi := (\pi_{sqr}, \pi_{exp}, c_{exp}) \in \mathbb{R}^3_{\geq 0}$ denote a tuple of hyperparameters. Let λ_d denote the d-dimensional Lebesgue measure. Furthermore, let $u : Y \to \mathbb{R}$ be strictly-increasing and continuous. Given a training set D^{train}, the loss function $\tilde{L}^\pi$ is defined as*

$$\tilde{L}^\pi(\widetilde{\mathcal{NN}}_\theta) := \underbrace{\sum_{i=1}^{n^{train}} (\hat{f}(x_i^{train}) - y_i^{train})^2}_{(\tilde{a})} \tag{62}$$

$$+\ \pi_{sqr} \cdot \underbrace{\sum_{i=1}^{n^{train}} (\widehat{\overline{UB}}(x_i^{train}) - y_i^{train})^2 + (\widehat{\underline{UB}}(x_i^{train}) - y_i^{train})^2}_{(\tilde{b})} \tag{63}$$

$$-\ \pi_{exp} \cdot \underbrace{\frac{1}{\lambda_d(X)} \int_X u(-\widehat{\underline{UB}}(x)) + u(\widehat{\overline{UB}}(x))\, dx}_{(\tilde{c})}. \tag{64}$$

The interpretations of the three terms $(\tilde{a})$, $(\tilde{b})$ and $(\tilde{c})$ are

analogous to the ones in the original NOMU loss.

Note that, the three sub-networks: the $\widehat{\overline{UB}}$-network, the $\widehat{\underline{UB}}$-network and the $\hat{f}$-network can also be trained independently using the corresponding terms in the loss function. Moreover, if one is only interested in the upper (lower) UB or $\mathcal{H}_{D^{\text{train}}}$ is only upwards (downwards) directed, i.e., fulfills only Assumption 1 (Assumption 2), then one can remove the respective sub-network from the architecture as well as the corresponding terms in the loss function. Furthermore, note that, now the obtained UBs can be asymmetric too.[34]

D. Discussion of the Desiderata

In this section, we discuss in more detail the desiderata proposed in Section 3.1. Specifically, we discuss how NOMU fulfills them and thereby prove several propositions. First, we establish a relation of NOMU to the classical Bayesian approach.

The Bayesian point of view allows for mathematically rigorous estimation of uncertainty. However, in general a fully Bayesian approach for quantifying uncertainty is very challenging and involves to

i. formulate a realistic prior,
ii. use an algorithm to approximate the posterior (challenging to get a good approximation in feasible time for complex models such as NNs),
iii. use this approximation of the posterior to obtain UBs.

We follow a different approach by directly approximating iii. based on essential properties of the posterior, e.g., for zero data noise model uncertainty vanishes at, and becomes larger far away from training data. This can be a reasonable approach in applications since many Bayesian state-of-the-art methods even fail to fulfill these basic properties when implemented in practice (Malinin & Gales, 2018) (see Figure 1). Since especially for NNs ii. is very costly, we ask ourselves the question, whether in practice it is even true that one has more intuition about the important properties of the prior than about the important properties of the posterior? In other words, can i. and ii. be skipped by directly approximating iii.? In the case of mean predictions, many successful algorithms following this procedure already exist. These algorithms directly try to approximate the posterior mean by exploiting one's intuition how the posterior of a realistic prior should behave without the need of precisely specifying the prior, e.g.,:

1. *Spline regression*: In many applications it is very intuitive that a good approximation of the posterior mean should not have unnecessarily large second derivative, without explicitly stating the corresponding prior. Even though spline regression can be formulated as the posterior mean of a limit of priors (Wahba, 1978), for a practitioner it can be much more intuitive to decide whether spline regression fits to one's prior beliefs by looking at the smoothness properties of the posterior mean than looking at such complex priors.
2. *Convolutional Neural Networks (CNNs)*: In image recognition, it is very intuitive to see that a good approximation of the posterior mean should fulfill typical properties, e.g., two pictures that are just slightly shifted should be mapped to similar outputs. CNNs fulfill such typical properties to a large extent and thus have proven to be very successful in practice. Nevertheless, from a Bayesian point of view these properties rely on a yet unknown prior.

[34] After the publication of the present paper, Weissteiner et al. (2023) implemented such an upper UB for monotonically non-decreasing functions.

D.1. Desideratum D1 (Non-Negativity)

Desideratum D1 (Non-Negativity) is trivial, since $\sigma_f \geq 0$ per definition. Credible bounds $\underline{CB}$ and $\overline{CB}$ are lower and upper bounds of an interval $[\underline{CB}, \overline{CB}]$, therefore $\underline{CB} \leq \overline{CB}$ holds by definition as well. To the best of our knowledge, every established method to estimate model uncertainty yields bounds that fulfil D1 (Non-Negativity). Furthermore, note that D1 (Non-Negativity) also holds in the presence of data noise uncertainty.

D.1.1. How Does NOMU Fulfill D1 (Non-Negativity)?

By definition, NOMU exactly fulfills D1 (Non-Negativity).

Proposition D.1.a *For NOMU, $\hat{\sigma}_f \geq 0$ and thus $\underline{UB}_c = \hat{f} - c\sigma_f \leq \hat{f} \leq \hat{f} + c\sigma_f = \overline{UB}_c$ for all $c \geq 0$.*

Proof. This holds since $\ell_{\text{min}} \geq 0$ in the readout map φ (see Equation (2)). □

D.2. Desideratum D2 (In-Sample)

In the case of zero data noise, Desideratum D2 (In-Sample) holds true exactly.

Proposition D.2.a (Zero Model Uncertainty at Training Points) *Let $\sigma_n \equiv 0$. Furthermore, let D^{train} be a finite set of training points and consider a prior distribution $\mathbb{P}[f \in \cdot]$ on the function space $\{f : X \to Y\}$ such that there exists a function in the support of $\mathbb{P}[f \in \cdot]$ that exactly fits through the training data. Then for the posterior distribution it holds that for all $(x^{train}, y^{train}) \in D^{train}$ that*

$$\mathbb{P}(f(x^{train}) = y^{train} | D^{train}) = 1 \tag{65}$$

$$\mathbb{P}(f(x^{train}) \neq y^{train} | D^{train}) = 0. \tag{66}$$

In words, there is no model uncertainty at input training points, i.e., $\sigma_f(x^{train}) = 0$ for all $x^{train} \in \{x^{train}$:

$(x^{train}, y^{train}) \in D^{train}\}$.

Proof. Intuitively, if the noise is zero, the data generating process is $y^{\text{train}} = f(x^{\text{train}}) + 0$. Thus, if we observe $(x^{\text{train}}, y^{\text{train}}) \in D^{\text{train}}$, we know that $f(x^{\text{train}}) = y^{\text{train}}$ with zero uncertainty. More formally, let $(x^{\text{train}}, y^{\text{train}}) \in D^{\text{train}}$ and define for some $\epsilon > 0$

$$U_\epsilon(y^{\text{train}}) := (y^{\text{train}} - \epsilon, y^{\text{train}} + \epsilon)$$
$$U_\epsilon(D^{\text{train}}) := \bigcup_{(x,y)\in D^{\text{train}}} U_\epsilon(x, y),$$

where $U_\epsilon(x, y)$ denotes an ϵ-ball around $(x, y) \in X \times Y$. Furthermore, let D be a random variable describing the data generating process assuming zero noise. Then it holds that

$$\begin{aligned}&\mathbb{P}(f(x^{\text{train}}) \in U_\epsilon(y^{\text{train}})^c | D \in U_\epsilon(D^{\text{train}}))\\ =&\frac{\mathbb{P}(D \in U_\epsilon(D^{\text{train}}) \wedge f(x^{\text{train}}) \in U_\epsilon(y^{\text{train}})^c)}{\mathbb{P}(D \in U_\epsilon(D^{\text{train}}))}\\ =&\frac{0}{\mathbb{P}(D \in U_\epsilon(D^{\text{train}}))}.\end{aligned}$$

Note that $\mathbb{P}(D \in U_\epsilon(D^{\text{train}})) > 0$ for every $\epsilon > 0$, since by assumption there exists a function in the support of the prior that exactly fits through the training data.[35] Defining the posterior

$$\begin{aligned}&\mathbb{P}(f(x^{\text{train}}) \neq y^{\text{train}} | D^{\text{train}}) :=\\ &\lim_{\epsilon\to 0} \mathbb{P}(f(x^{\text{train}}) \in U_\epsilon(y^{\text{train}})^c | D \in U_\epsilon(D^{\text{train}})),\end{aligned}$$

in the canonical way concludes the proof. □

Even if theoretically, we know that $\sigma_f(x^{\text{train}}) = 0$ at all training points x^{train}, in practice $\hat{\sigma}_f(x^{\text{train}}) \approx 0$ can be acceptable (due to numerical reasons).

D.2.1. Non-Zero Data Noise

For non-zero data noise there is non-zero *data-noise induced model uncertainty* at input training points. However, also for non-zero but small data noise the model uncertainty at input training points should not be significantly larger than the data noise. In fact, for GPs one can rigorously show that $\sigma_f(x^{\text{train}}) \leq \sigma_{\text{n}}(x^{\text{train}})$ in the case of known σ_{n}.

Proposition D.2.b (GPs Model Uncertainty at Training Points) *Let D^{train} be a set of training points. For a prior $f \sim \mathcal{GP}(m(\cdot), k(\cdot,\cdot))$ and fixed σ_n it holds that*

$$\sigma_f(x^{train}) \leq \sigma_n(x^{train}), \tag{67}$$

for all input training points x^{train}.

[35] Formally, $\exists f^*$ that fits exactly through D^{train} with the property $\mathbb{P}[f \in U_\epsilon(f^*)] > 0$. Since, $0 < \mathbb{P}[f \in U_\epsilon(f^*)] < \mathbb{P}(D \in U_\epsilon(D^{\text{train}}))$ (for the canonical L_∞-topology) the claim follows. Given this one can also see that Proposition D.2.a still holds true with the even weaker assumption $\mathbb{P}(D \in U_\epsilon(D^{\text{train}})) > 0$.

Proof. We prove the proposition by induction over the number of training points n^{train}. For this let

$$A^{n^{\text{train}}} := \{x^{\text{train}} : (x^{\text{train}}, y^{\text{train}}) \in D^{\text{train}}\}.$$

- **Base case** $n^{\text{train}} = 1$: In this case $A^{n^{\text{train}}} = A^1 = \{x_1^{\text{train}}\}$. Let $k := k(x_1^{\text{train}}, x_1^{\text{train}})$. Since

$$\sigma_f^2(x_1^{\text{train}}) \overset{(29)}{=} k - k\frac{1}{k + \sigma_{\text{n}}^2(x_1^{\text{train}})} \cdot k \leq \sigma_{\text{n}}^2(x_1^{\text{train}}) \tag{68}$$
$$\iff k^2 \geq k^2 - \sigma_{\text{n}}^4(x_1^{\text{train}}), \tag{69}$$

  the claim follows.
- $n^{\text{train}} = m$: Let $K(A^m, A^m)$ be the Gram matrix and let

$$P := [K(A^m, A^m) + diag(\sigma_{\text{n}}(A^m))].$$

  We then assume for all $x^{\text{train}} \in A^m$ that

$$\sigma_f^2(x^{\text{train}}|A^m) \overset{(29)}{=} \tag{70}$$
$$k(x^{\text{train}}, x^{\text{train}}) - k(x^{\text{train}}, A^m)^T P^{-1} k(x^{\text{train}}, A^m) \leq \sigma_{\text{n}}^2(x^{\text{train}}) \tag{71}$$

- **Inductive step** $n^{\text{train}} = m + 1$: We now show that under the inductive assumption for any $x^{\text{train}} \in A^{m+1}$ we have

$$\begin{aligned}&\sigma_f^2(x^{\text{train}})\\ &\overset{(29)}{=} k(x^{\text{train}}, x^{\text{train}}) - \begin{pmatrix} k(x^{\text{train}}, A^m)\\ k(x^{\text{train}}, x_{m+1}^{\text{train}})\end{pmatrix}^T \begin{pmatrix} P & Q\\ R & S\end{pmatrix}^{-1} \begin{pmatrix} k(x^{\text{train}}, A^m)\\ k(x^{\text{train}}, x_{m+1}^{\text{train}})\end{pmatrix} \qquad (72)\\ &\leq \sigma_{\text{n}}^2(x^{\text{train}})\end{aligned}$$

  with

$$\begin{aligned}R &:= (k(x_1^{\text{train}}, x_{m+1}^{\text{train}}), \dots k(x_m^{\text{train}}, x_{m+1}^{\text{train}})) = k(x_{m+1}^{\text{train}}, A^m)^T,\\ Q &:= (k(x_1^{\text{train}}, x_{m+1}^{\text{train}}), \dots k(x_m^{\text{train}}, x_{m+1}^{\text{train}}))^T = R^T,\\ S &:= k(x_{m+1}^{\text{train}}, x_{m+1}^{\text{train}}) + \sigma_{\text{n}}^2(x_{m+1}^{\text{train}}).\end{aligned}$$

  Setting $k := k(x^{\text{train}}, x^{\text{train}}), v := k(x^{\text{train}}, A^m)$, and $w := k(x^{\text{train}}, x_{m+1}^{\text{train}})$ (72) can be rewritten as

$$k - \begin{pmatrix} v\\ w\end{pmatrix}^T \begin{pmatrix} \tilde{P} & \tilde{Q}\\ \tilde{R} & \tilde{S}\end{pmatrix} \begin{pmatrix} v\\ w\end{pmatrix} \leq \sigma_{\text{n}}^2(x^{\text{train}}) \tag{73}$$

  with submatrices $\tilde{P}, \tilde{Q}, \tilde{R}, \tilde{S}$ as in (Williams & Rasmussen, 2006, (A.12)). Furthermore, with $M := (S - RP^{-1}Q)^{-1}$ as in (Williams & Rasmussen, 2006, (A.12)), we get that (73) is equivalent to

$$\begin{aligned}&k - (v^T\tilde{P}v + v^T\tilde{Q}w + w\tilde{R}v + w\tilde{S}w) \leq \sigma_{\text{n}}^2(x^{\text{train}})\\ \iff &k - (v^T P^{-1} v + v^T P^{-1} QMRP^{-1} v - v^T P^{-1} QMw\\ &\quad - wMRP^{-1}v + wMw) \leq \sigma_{\text{n}}^2(x^{\text{train}})\\ \iff &k - v^T P^{-1} v - (v^T P^{-1} Q - w)M(RP^{-1}v - w) \leq \sigma_{\text{n}}^2(x^{\text{train}})\\ \iff &\underbrace{k - v^T P^{-1} v}_{=\sigma_f^2(x^{\text{train}}|A^m)} - (v^T P^{-1} Q - w)^2 M \leq \sigma_{\text{n}}^2(x^{\text{train}}) \qquad (74)\end{aligned}$$

where the last line follows since $R^T = Q$ and P symmetric. Note that

$$\begin{aligned} M &= \Big(k(x_{m+1}^{\text{train}}, x_{m+1}^{\text{train}}) + \sigma_{\text{n}}^2(x_{m+1}^{\text{train}}) \\ &\quad - k(x_{m+1}^{\text{train}}, A^m)^T P^{-1} k(x_{m+1}^{\text{train}}, A^m) \Big)^{-1} \\ &= \left(\sigma_{\text{n}}^2(x_{m+1}^{\text{train}}) + \sigma_f^2(x_{m+1}^{\text{train}}|A^m)\right)^{-1}. \end{aligned}$$

With this, (74) can be further reformulated as

$$\begin{aligned} &\sigma_f^2(x^{\text{train}}|A^m) - \underbrace{\frac{\left(k(x^{\text{train}}, A^m)^T P^{-1} k(x_{m+1}^{\text{train}}, A^m) - k(x^{\text{train}}, x_{m+1}^{\text{train}})\right)^2}{\sigma_{\text{n}}^2(x_{m+1}^{\text{train}}) + \sigma_f^2(x_{m+1}^{\text{train}}|A^m)}}_{\geq 0} \\ &\leq \sigma_{\text{n}}^2(x^{\text{train}}). \end{aligned} \tag{75}$$

First, for $x^{\text{train}} \in A^m$, (75) holds true by assumption. Second, for $x^{\text{train}} = x_{m+1}^{\text{train}}$ we obtain

$$\begin{aligned} &\sigma_f^2(x_{m+1}^{\text{train}}|A^m) - \frac{\left(\sigma_f^2(x_{m+1}^{\text{train}}|A^m)\right)^2}{\sigma_{\text{n}}^2(x_{m+1}^{\text{train}}) + \sigma_f^2(x_{m+1}^{\text{train}}|A^m)} \leq \sigma_{\text{n}}^2(x_{m+1}^{\text{train}}) \\ \iff &\sigma_f^4(x_{m+1}^{\text{train}}|A^m) - \sigma_{\text{n}}^4(x_{m+1}^{\text{train}}) \leq \sigma_f^4(x_{m+1}^{\text{train}}|A^m) \\ \iff &-\sigma_{\text{n}}^4(x_{m+1}^{\text{train}}) \leq 0. \end{aligned}$$

Thus, (73) holds true for any $x^{\text{train}} \in A^{m+1}$. □

In fact, Proposition D.2.b does not come as a surprise: Even if we only observe one training point $(x^{\text{train}}, y^{\text{train}})$ and ignore all our prior knowledge by using a flat "uninformative" improper prior $p(f(x^{\text{train}})) \propto 1$, this results in $\sigma_f(x^{\text{train}}) = \sigma_{\text{n}}(x^{\text{train}})$. Introducing additional information, e.g., observing more additional training points and introducing additional prior information (such as smoothness assumptions instead of a flat uninformative prior), typically reduces model uncertainty further. Thus, we believe that $\sigma_f(x^{\text{train}}) \leq \sigma_{\text{n}}(x^{\text{train}})$ holds for most reasonable priors.

Finally, note that Proposition D.2.a and Proposition D.2.b hold true for *every* prior respectively *every Gaussian* process prior as long as there exists an f in the support of this prior which explains the observed training points (even if this prior is strongly misspecified). For example this assumption is obviously fulfilled for the prior of Gaussian distributed weights of an overparameterized NN (BNN).

D.2.2. Why Does MC Dropout Strongly Violate D2 (In-Sample) ?

In Figure 1, MC dropout (MCDO) predicts for every input training point $\hat{\sigma}_f(x_i^{\text{train}}) > 100\sigma_{\text{n}}$. Thus, if $\hat{\sigma}_f(x_i^{\text{train}})$ was correctly calculated as posterior model uncertainty, this would be an practically unobservable event as long as f actually comes from this prior ($\mathbb{P}\left[|y_i^{\text{train}} - f(x_i^{\text{train}})| > 100\sigma_{\text{n}}\right] < 10^{-2173}$). Therefore, this is clear statistical evidence that MCDO severely fails to estimate posterior model uncertainty at training points. This can have one of the following three reasons:

1. MCDO severely fails in correctly approximating the posterior given its prior (i.i.d. Gaussian on weights).
2. MCDO's prior does not fit to the data generating process at all.
3. During our experiments we very often observed very extreme events that should only happen with probabilities smaller than 10^{-2000}.

We agree with prior work (Gal & Ghahramani, 2016; Blundell et al., 2015) that a Gaussian prior on the weights of a NN, i.e., the prior mentioned in Item 2, is a very reasonable assumption. Note that NOMU can also be seen as a heuristic to approximate the posterior model uncertainty given exactly the same prior (see Appendix A.2). Therefore, since Item 3 can be ruled out, we can conclude that MCDO's problem is Item 1.

MC Dropout's Failure in Approximating the Posterior Table 10 and Table 11 show that even though we generate the ground truth function from the *same* prior assumed by MCDO (and also assumed by most BNN algorithms), NOMU significantly outperforms MCDO. This empirically shows (with the help of Theorem B.7) that (i) NOMU is able to better approximate posterior BNN-credible bounds than MCDO in terms of average Kullback-Leibler divergence $\bar{d}_{\text{KL}}$ (including further popular variational BNN approximations from (Graves, 2011; Blundell et al., 2015; Hernández-Lobato & Adams, 2015), which themselves are outperformed by MCDO) and (ii) MCDO's variational approximation algorithm severely fails in approximating the targeted posterior.

D.2.3. Importance of D2 (In-Sample) in BO

Especially in Bayesian optimization (BO) it is particularly important to fulfill D2 (In-Sample) as much as possible, since D2 (In-Sample) helps a lot to prevent the BO-algorithm from getting stuck in local maxima. For NNs, we often observed that at the i'-th step, the mean prediction $\hat{f}$ is maximized/minimized either at the boundary or exactly at a training point with the largest/smallest function value observed so far $\max_{i\in\{1,\dots,i'\}} f(x_i)/\min_{i\in\{1,\dots,i'\}} f(x_i)$ (see Figure 9). In the latter case, without model uncertainty (or with almost constant model uncertainty as is the case in MC dropout), one would query all future function evaluations at exactly this point without learning anything new. E.g., consider the situation of Figure 9d when minimizing the *Forrester* function. Each new function evaluation of MC Dropout would be sampled at an already observed training point $x \approx 0.4$. This intuitively explains why estimating model uncertainty precisely at the training data points is

especially important in BO and why it can be very problematic in BO, if the model uncertainty does not decrease sufficiently at the training data points. To summarize, D2 (In-Sample) strongly influences the acquisition function in a direction that discourages the algorithm from choosing the same point again and D2 (In-Sample) together with D3 (Out-of-Sample) can prevent the BO-algorithm from getting stuck (see also Appendix B.3.2).

D.2.4. Dominating Data Noise

In the case of dominating data noise uncertainty $\sigma_{\text{n}} \gg 0$, the model uncertainty σ_f should not be small at input training points (only if one observes a very large amount of input training points very close to a input training point x^{train} the model uncertainty should become small.) However, in this paper, we do not focus on the case of large data noise uncertainty, but on the case of negligible or zero data noise. In particular, D2 (In-Sample) is only formulated for this case.

D.2.5. How Does NOMU Fulfill D2 (In-Sample)?

Recall, that we train NOMU by minimizing

$$L^{\pi}(\mathcal{NN}_{\theta}) + \lambda \|\theta\|_2^2 \,, \tag{76}$$

where the NOMU loss $L^{\pi}(\mathcal{NN}_{\theta})$ is defined as:

$$L^{\pi}(\mathcal{NN}_{\theta}) := \underbrace{\sum_{i=1}^{n^{\text{train}}} (\hat{f}(x_i^{\text{train}}) - y_i^{\text{train}})^2}_{(a)} + \pi_{\text{sqr}} \cdot \underbrace{\sum_{i=1}^{n^{\text{train}}} (\hat{r}_f(x_i^{\text{train}}))^2}_{(b)} \tag{77}$$

$$+ \pi_{\text{exp}} \cdot \underbrace{\frac{1}{\lambda_d(X)} \int_X e^{-c_{\text{exp}} \cdot \hat{r}_f(x)} \, dx}_{(c)} \,. \tag{78}$$

Then, the following proposition holds:

Proposition D.2.c *Let $\lambda, \pi_{exp}, c_{exp} \in \mathbb{R}_{\geq 0}$ be fixed and let $\hat{\sigma}_f$ be NOMU's model uncertainty prediction. Then, it holds that $\hat{\sigma}_f(x_i^{train})$ converges to ℓ_{min} for $\pi_{sqr} \to \infty$ for all input training points x_i^{train}, where $\ell_{min} \geq 0$ is an arbitrarily small constant modelling a minimal model uncertainty used for numerical reasons.*

Proof. By the definition of $L^{\pi}(\mathcal{NN}_{\theta})$, i.e., since (b) dominates the loss function if $\pi_{\text{sqr}} \to \infty$, it follows that $\hat{r}_f(x_i^{\text{train}}) = 0$. More precisely, for the NN $\mathcal{NN}_{\theta^*} = (\hat{f}^*, \hat{r}_f^*)$ with parameters θ^* that minimize (76) it holds that

$$\begin{aligned}
& L^{\pi}(\mathcal{NN}_{\theta^*}) + \lambda \|\theta^*\|_2^2 \leq L^{\pi}(0) + \lambda \|0\|_2^2 \\
& \quad = \sum_{i=1}^{n^{\text{train}}} (y_i^{\text{train}})^2 + \pi_{\text{exp}} \cdot 1 \\
\iff & \sum_{i=1}^{n^{\text{train}}} (\hat{f}^*(x_i^{\text{train}}) - y_i^{\text{train}})^2 + \pi_{\text{sqr}} \cdot \sum_{i=1}^{n^{\text{train}}} \left(\hat{r}_f^*(x_i^{\text{train}})\right)^2 + \\
& \pi_{\text{exp}} \cdot \frac{1}{\lambda_d(X)} \int_X e^{-c_{\text{exp}} \cdot \hat{r}_f^*(x)} \, dx + \lambda \|\theta^*\|_2^2 \leq \sum_{i=1}^{n^{\text{train}}} (y_i^{\text{train}})^2 + \pi_{\text{exp}} \\
\iff & \pi_{\text{sqr}} \cdot \sum_{i=1}^{n^{\text{train}}} \left(\hat{r}_f^*(x_i^{\text{train}})\right)^2 \leq \sum_{i=1}^{n^{\text{train}}} (y_i^{\text{train}})^2 + \pi_{\text{exp}} =: C
\end{aligned}$$

where for fixed parameters $\lambda, \pi_{\text{exp}}, c_{\text{exp}} \in \mathbb{R}_{\geq 0}$, $C > 0$ is a constant. Assume now that for $\hat{r}_f^*$ does not vanish at all training data points for π_{sqr} to infinity, i.e., that there exists an $\epsilon > 0$ such that for every π_{sqr} large enough $\sum_{i=1}^{n^{\text{train}}} \left(\hat{r}_f^*(x_i^{\text{train}})\right)^2 > \epsilon$. This however implies

$$\pi_{\text{sqr}} \cdot \epsilon < C \iff \pi_{\text{sqr}} < \frac{C}{\epsilon} \quad \forall \pi_{\text{sqr}} \text{ large enough,}$$

which yields a contradiction. Thus, $\lim_{\pi_{\text{sqr}} \to \infty} \hat{r}_f^*(x^{\text{train}}) = 0$ for all training input points x^{train}. Finally, by Equation (2) it follows that $\hat{\sigma}_f(x_i^{\text{train}}) = \ell_{\text{min}}$. □

Note that even for a finite (sufficiently large) π_{sqr}, the raw model uncertainty $\hat{r}_f$ converges to zero as λ goes to zero ($\frac{\pi_{\text{sqr}}}{\lambda} \to \infty$), if the model is sufficiently over-parameterized. Empirically one can see in Figures 1, 4, 5, 9, 11, 13, 18 and 19 that NOMU fulfills D2 (In-Sample) with a high precision for our choice of hyper-parameters.

D.3. Desideratum D3 (Out-of-Sample)

We first consider the case of zero (or negligible) data noise $\sigma_{\text{n}} \approx 0$ and then discuss possible extensions to settings with non-zero data noise.

D.3.1. Zero Data Noise

The notion of distance used in D3 (Out-of-Sample) heavily depends on the specific application (i.e., on the prior used in this application). More concretely, there are the following two "hyperparameters".

1. First, the metric[36] $d : X \times X \to \mathbb{R}_{\geq 0}$ on X we use to measure distances can heavily depend on the prior for the specific application. For example, in the case of image recognition, two pictures that are only slightly shifted can be seen as very close to each other even if the Euclidean distance of their pixel-values is quite high.[37] If one uses a CNN-architecture in NOMU this prior belief on d is

[36] We use the term "metric" to describe a general pseudometric.

[37] For example, if one sees a 1920×1080-pixel image, which is

approximately captured. The successful generalization properties of many different network architectures can be explained precisely by their use of application-dependent *non-Euclidean* metrics (Bronstein et al., 2017). (Additionally, instead of fixing d apriori, further aspects of the metric d can be learned from the training data as we discuss in detail in Appendix D.4.)

2. Second, even if we can precisely write down a metric $d : X \times X \to \mathbb{R}_{\geq 0}$, a priori it is not clear how to define the distance $\tilde{d} : X \times 2^X \to \mathbb{R}_{\geq 0}$ between a point x and the input training points from $D_X^{\text{train}} := \{x^{\text{train}} : (x^{\text{train}}, y^{\text{train}}) \in D^{\text{train}}\}$. Both common definitions $\tilde{d}(x, D_X^{\text{train}}) := \inf_{z \in D_X^{\text{train}}} d(x, z)$ and $\tilde{d}(x, D_X^{\text{train}}) := \inf_{z \in \text{Conv}(D_X^{\text{train}})} d(x, z)$, where $\text{Conv}(\cdot)$ denotes the convex hull, are inappropriate choices for $\tilde{d}$.[38] In Section 3.1, we consider a point x closer to the input training points if it is "surrounded" by input training points in all directions, as opposed to a point x which only has close input training points in some directions and there is a large range of directions without any close input training points. This implies that, for example,

   (i) very close to noiseless input training points that are surrounded by many other noiseless input training points there is very little model uncertainty.

   (ii) for extrapolation one typically has more uncertainty than for interpolation.

   (iii) far away from the convex hull of the training points model uncertainty is very high.

   (iv) also within the convex hull of the training data, model uncertainty is high within big gaps in-between training points.

   Figure 11 shows how well NOMU fulfills these properties of $\tilde{d}$ similarly to a GP (see Figure 12a).

D.3.2. Non-Zero Homoscedastic Data Noise

If there is homoscedastic non-zero data noise $\sigma_n(x) \equiv \sigma_n > 0$, it is important that the "distance" $\tilde{d}$ of x to the input training points is not minimal if it exactly equals one of the input training points. Instead, one should use a notion of distance $\tilde{d}$ that can even get smaller if there are multiple input training points at x or very close to x.

perfectly recognizable as a cat, every 10-pixel shift of this picture is also recognizable as a cat with almost no uncertainty (even though this cannot be proven mathematically). Thus, it is very desirable to predict very small model uncertainty $\hat{\sigma}_f(x)$ for every image $x \in X$ which is only shifted by less than 10 pixel from at least one noiselessly labeled training image x^{train}.

[38] E.g., for GPs, none of these two classical notions of distance between a point and a set is entirely applicable (see Equation (29)). An appropriate choice of $\tilde{d}$ should be a compromise between these two notions.

D.3.3. Non-Zero Heteroscedastic Data Noise

One can also extend D3 (Out-of-Sample) to heteroscedastic settings. In that case, the used notion of "distance" $\tilde{d}$ of x to the input training points needs to be weighted by the precision of the input training points, i.e., if x is close to multiple input training points with low data noise $\sigma_n(\cdot)$ you consider x "closer" to the input training points than if x is close to multiple input training points with high data noise.

D.3.4. Example for $\tilde{d}$ Based on GPs

In this section, we give the concrete example of Gaussian process regression (GPR) from Appendix B.1.1 in which $\tilde{d}$ from D3 (Out-of-Sample) can be written down explicitly in closed form.

For any arbitrary metric d on X, consider for instance the kernel $k(x_i, x_j) = e^{-d(x_i, x_j)^2}$. Then, $\tilde{d}(x, D_X^{\text{train}}) = \hat{\sigma}_f(x | D_X^{\text{train}})$, with posterior model uncertainty $\hat{\sigma}_f$ from Equation (29). While this is one of the simplest possible ways to define $\tilde{d}$, alternatively one could also define it differently if it shares similar qualitative properties.[39]

Why do we still consider it interesting to formulate D3 (Out-of-Sample) *vaguely*, given that there is already such a precise formula as is the case for GPs?

1. The GP's formula only holds true for the specific prior of a GP. We however, want to formulate desiderata that capture the most essential properties of credible bounds that almost all reasonable priors have in common.

2. We want to provide some easy to understand intuition for D3 (Out-of-Sample): It might be challenging to see directly from the GP's formula (29) how the posterior model uncertainty qualitatively behaves as visualized in Figure 12a.

To summarize, both the exact notion of distances $d, \tilde{d}$ *and* the exact rate of how model uncertainty increases with increasing distance to the input training points depend on one's prior belief. However, Section 3.1 gives a qualitative description of properties that most reasonable (generic) priors have in common (see Items (i)–(iv)).

D.3.5. How Does NOMU Fulfill D3 (Out-of-Sample)?

Recall, that we train NOMU by minimizing

$$L^{\pi}(\mathcal{NN}_{\theta}) + \lambda \|\theta\|_2^2 , \tag{79}$$

[39] For instance, $\tilde{d}$ could also be defined based on a kernel of the form $k(x_i, x_j) = g(d(x_i, x_j))$ with a monotonically decreasing function g, e.g., a Matérn-typed kernel.

where the NOMU loss $L^{\pi}(\mathcal{NN}_{\theta})$ is defined as:

$$L^{\pi}(\mathcal{NN}_{\theta}) := \underbrace{\sum_{i=1}^{n^{\text{train}}} (\hat{f}(x_i^{\text{train}}) - y_i^{\text{train}})^2}_{(a)} + \pi_{\text{sqr}} \cdot \underbrace{\sum_{i=1}^{n^{\text{train}}} (\hat{r}_f(x_i^{\text{train}}))^2}_{(b)} \tag{80}$$

$$+ \pi_{\text{exp}} \cdot \underbrace{\frac{1}{\lambda_d(X)} \int_X e^{-c_{\text{exp}} \cdot \hat{r}_f(x)} \, dx}_{(c)} . \tag{81}$$

The interplay of (b), (c), and regularization promotes D3 (Out-of-Sample) (note that the behaviour of $\hat{r}_f$ directly translates to the behaviour of σ_f): Term (c) pushes $\hat{r}_f$ towards infinity across the whole input space X. However, due to the counteracting force of (b) as well as regularization, $\hat{r}_f$ increases continuously as you move away from the training data – see for example Figure 9 and Figure 11 (or any other plot showing NOMU, i.e., Figures 1, 4, 5, 13, 18 and 19). In Figure 11, one can see how NOMU fulfills the properties (i)–(iv) of $\tilde{d} : X \times 2^X \to \mathbb{R}_{\geq 0}$ mentioned in Appendix D.3.1. In Figures 18 and 19, one can observe how NOMU behaves when a non-stationary metric $d_{\text{Figure 18}}(x, x') \neq |x - x'|$ respectively non-stationary non-isotropic metric $d_{\text{Figure 19}}(x, x') \neq \|x - x'\|_2$ is used (because $d_{\text{Figure 18}}$ and $d_{\text{Figure 19}}$ were learned from the data as desired by D4 (Metric Learning) in these examples).

The hyperparameters π_{exp} and c_{exp} control the size and shape of the UBs. Concretely, the larger π_{exp}, the wider the UBs; the larger c_{exp}, the narrower the UBs at points x with large $\hat{\sigma}_f(x)$ and the wider the UBs at points x with small $\hat{\sigma}_f(x)$.

Finally, we give some intuition that if CNNs are used for the two sub-networks in NOMU's architecture, D3 (Out-of-Sample) will be fulfilled with respect to an almost shift-invariant metric d: In the noiseless setting, we can choose π_{sqr} large enough such that D2 (In-Sample) is fulfilled, so that we have $\hat{\sigma}_f(x^{\text{train}}) \approx 0$ at any training input point x^{train}. Regularized CNNs have the property that if you slightly shift the input the output barely changes. So if x can be obtained from x^{train} by slightly shifting it, the CNN-output $\hat{\sigma}_f(x) \approx \hat{\sigma}_f(x^{\text{train}}) \approx 0$ also does not move too far away from zero. Only if you move further away with respect to the almost shift-invariant metric d, the CNN-output $\hat{\sigma}_f$ is able to move further away from zero. The same principle can also be used for other geometric NNs (e.g., graph neural networks (GNNs)) which correspond to different (non-Euclidean) metrics Bronstein et al. (2017).

D.4. Desideratum D4 (Metric Learning)

A priori, it is often not obvious which metric d to choose in D3 to measure distances. In many applications, it is therefore best to *learn* this metric from the training data (as explained in Footnote 4 on Page 4).

In the following section, we present visualizations of D4 (Metric Learning) for all benchmark algorithms in easy to understand, low dimensional settings.

D.4.1. Visualization of D4 (Metric Learning)

1D In order to visualize D4 and show how for NOMU the mean prediction impacts it's model uncertainty prediction we conduct the following experiment. We sample 16 *equidistant* noiseless training points of a trend-adjusted version of *Sine 3*. We then train NOMU (hyperparameters are as in Appendix B.2.2 with $\pi_{\text{sqr}} = 0.5$, $\ell_{\text{min}} = 10^{-4}$, regularization parameter 10^{-4} on the $\hat{r}_f$-network, and number of training epochs 2^{12}) and compute the corresponding UBs. Figure 18 shows that NOMU UBs are wider (cp. the dotted blue line) in those areas of the input space where small changes of x lead to large variation in the target ($\approx x \geq 0$) compared to areas without large variation in the target ($\approx x \leq 0$). This effect is present even though the input training points are sampled from an *equidistant* grid, and thus isolates the effect of D4.

2D Analogously, we visualize D4 for two-dimensional input by training NOMU on 16 training points sampled on an *equidistant* 4×4-grid and evaluated at the two-dimensional extension of the *Step* function, i.e.,

$$f = \mathbb{R}^2 \to \mathbb{R} : (x_1, x_2) \mapsto \begin{cases} -1 & \text{if } x_1 < 0 \\ 1 & \text{if } x_1 \geq 0. \end{cases} \tag{82}$$

Here, D4 can be interpreted as follows: imagine we do not have any prior knowledge of whether x_1 or x_2 is more important for predicting the unknown function f. However, when NOMU observes the 16 training points it should be able to learn that x_1 is more important for the model prediction than x_2, *and* that the function is more regular/predictable far away from $\{x_1 \approx 0\}$. D4 requires in this example that feature x_1 should have a higher impact than feature x_2 also

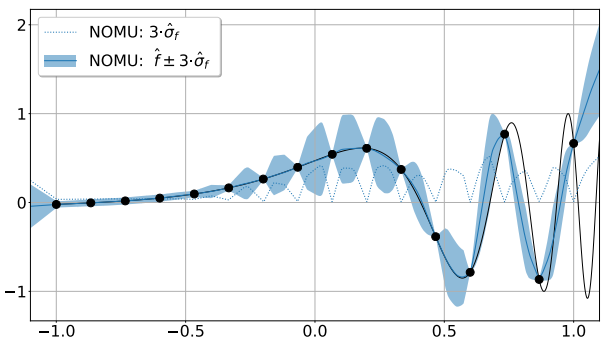


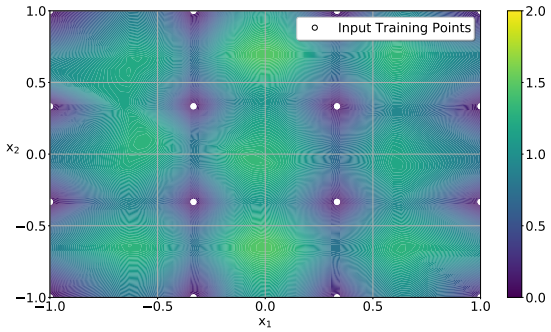
Figure 18. Visualisation of D4 (Metric Learning).

on the model uncertainty prediction. If a model for UBs did not incorporate D4, we would expect the uncertainty in this example to fulfill $\hat{\sigma}_f((x_1, x_2)) = \hat{\sigma}_f((x_2, x_1))$ because of the equidistant grid of the training points (this is indeed the case for GPs, see Figure 20a).

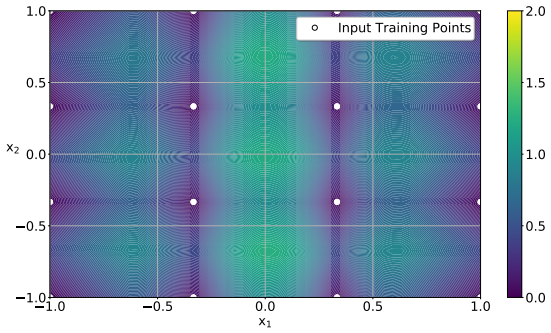
For NOMU however, we have very good control on how strongly we enforce D4, e.g., we can strengthen D4 by increasing the L2-regularization of the hidden layers in the $\hat{r}_f$-network and/or decreasing the size of the $\hat{r}_f$-network.

NOMU: Visualization of D4 in 2D In Figure 19, we present the estimated model uncertainty $\hat{\sigma}_f$ obtained for *different* hyperparameters of the $\hat{r}_f$-network with *fixed* $\hat{f}$-architecture among all four subplots. Thus, Figure 19 shows how D4 realizes in different magnitudes. In Figure 19a, we use the same hyperparameters for the $\hat{r}_f$-network as for the $\hat{f}$-network. In 19b, we only increase the L2-regularization of the $\hat{r}_f$-network. In Figure 19c, we only decrease the size of $\hat{r}_f$-networks. In Figure 19d, we combine both, i.e., we increase the L2-regularization of the $\hat{r}_f$-network *and* decrease the size of the $\hat{r}_f$-network. While D4 is barely visible in Figure 19a, it is clearly visible in Figures 19b–19d. In Figures 19b–19d, we observe that the estimated model uncertainty $\hat{\sigma}_f$ grows faster in horizontal directions (corresponding to changes in x_1) than in vertical directions. In Figures 19b–19d, we further observe that the estimated model uncertainty $\hat{\sigma}_f$ is larger around $\{x_1 \approx 0\}$ than far away from this region. The magnitude of both these effects increases from Figure 19b to Figure 19d. Both of these effects can also be observed for MC dropout (MCDO) and deep ensembles (DE) (see Figure 20b and Figure 20c).

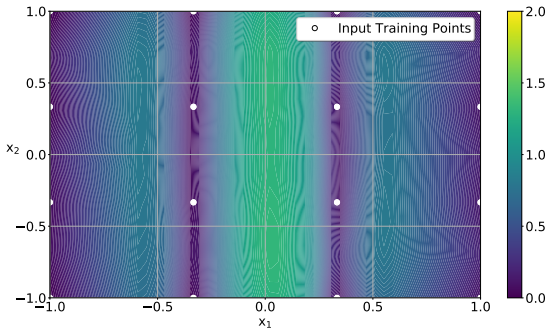
Benchmarks: Visualization of D4 in 2D In Figure 20, we present uncertainty plots of all benchmark methods. We can see that deep ensembles (DE) gives high preference to capturing D4, even though its estimated model uncertainty still is subject to some randomness with non-uniform patterns for $x_1 \in [-0.25, 0.25]$ (Figure 20c). Moreover, MC dropout (MCDO) also captures higher model uncertainty for $x_1 \in [-0.25, 0.25]$ as desired by D4, but it does not fulfill D2 (Figure 20b). The Gaussian process (GP) with RBF kernel does not account for D4 (Figure 20a), which directly follows from the definition. Similarly to deep ensembles (DE), hyper deep ensembles (HDE) and HDE* strongly capture D4 but show even more random behaviour. This randomness is visible most prominently along $x_1 = 0$ where one should observe large model uncertainty, whereas their estimated model uncertainty is surprisingly close to 0.



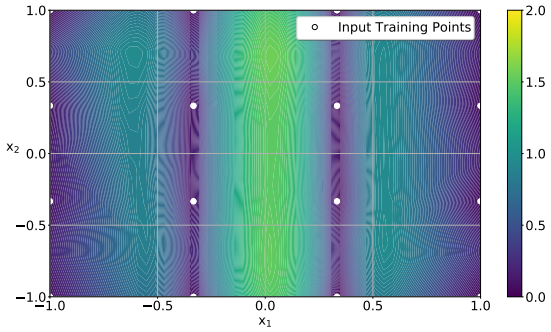
(a) Same L2-regularization on the $\hat{r}_f$-network and $\hat{f}$-network ($\lambda = 10^{-8}$).



(b) Larger L2-regularization on the $\hat{r}_f$-network ($\lambda = 10^{-4}$) than on the $\hat{f}$-network ($\lambda = 10^{-8}$).

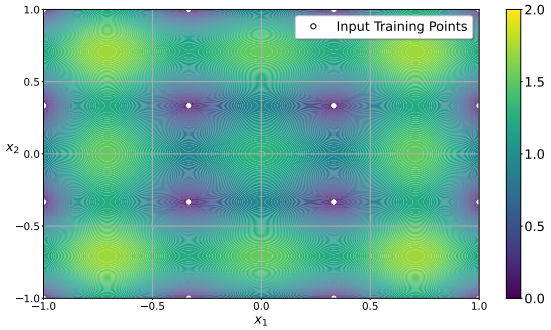


(c) Shallow $\hat{r}_f$-network consisting of 4 hidden nodes.

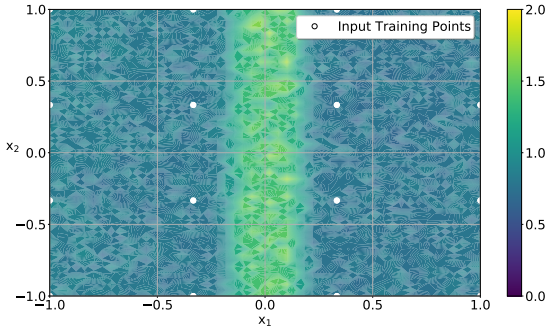


(d) Shallow $\hat{r}_f$-network consisting of 4 hidden nodes *and* larger regularization of $\lambda = 10^{-4}$ on $\hat{r}_f$-network.

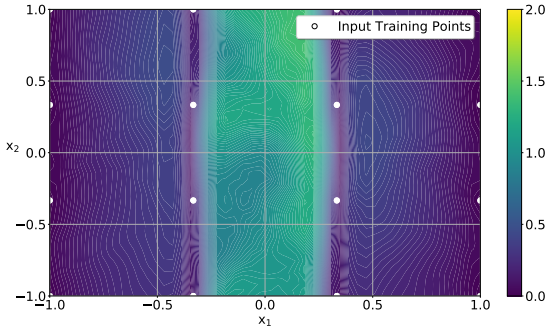
Figure 19. Estimated model uncertainty $\hat{\sigma}_f$ of NOMU: visualizing D4 in 2D.



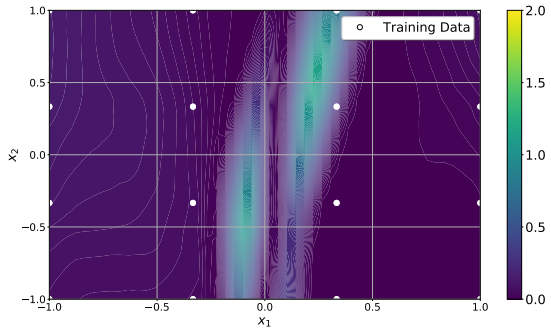
(a) GP (c=15)



(b) MCDO (c=20)

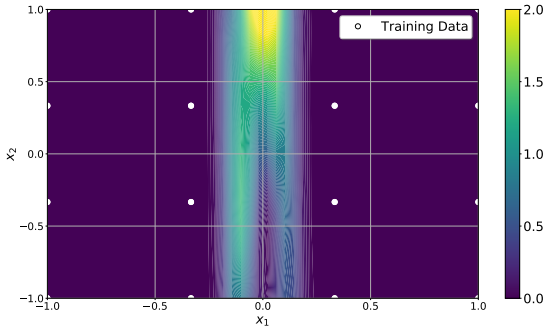


(c) DE (c=30)



(d) HDE (c=30)

Figure 20. Estimated model uncertainty of Gaussian process (GP), MC dropout (MCDO), deep ensembles (DE), and hyper deep ensembles (HDE)



(e) HDE* (c=30)

Figure 20. (cont.) Estimated model uncertainty of HDE*.

D.4.2. How Does NOMU Fulfill D4 (Metric Learning)?

Recall NOMU's architecture depicted in Figure 2 in the main paper.

The $\hat{r}_f$-network learns the raw model uncertainty and is connected with the $\hat{f}$-network through the last hidden layer (dashed lines in Figure 2). This connection enables $\hat{r}_f$ to re-use features that are important for the model prediction $\hat{f}$. More theory on how features are reused in L2-regularized deep neural networks can be found in (Heiss et al., 2022). This behaviour directly translates to $\hat{\sigma}_f$ (see Equation (2)), implementing Desideratum D4 (Metric Learning).

In Figures 18 and 19, one can observe that NOMU fulfills D4 (Metric Learning). Moreover, Figure 19 shows how to control the strength of D4 (Metric Learning) by varying the L2-regularization and the number of neurons of the $\hat{r}_f$-architecture.

D.5. Desideratum D5 (Vanishing)

First, note that D3 (Out-of-Sample) already suggest D5 (Vanishing), since for $n^{\text{train}} \to \infty$ every point x in the support of the input data generating distribution is infinitely close to infinitely many other i.i.d input training points x^{train} and thus infinitely close to the input training set.

To the best of our knowledge, D5 (Vanishing) is fulfilled by most reasonable algorithms that estimate model uncertainty. Specifically, NOMU, GPs, DE and HDE capture D5 (Vanishing). This can be nicely observed in Figure 13, where all these algorithms result in zero model uncertainty in areas with many input training points (up to numerical precision).

MC Dropout can be seen as a variational algorithm for approximating BNNs. While in theory, BNNs should also fulfill D5 (Vanishing), MC Dropout often struggles to do so, as can be observed in Figure 13 (see also D.2.2 for a discussion why MC Dropout struggles to approximate BNNs).

Finally, D5 (Vanishing) is widely accepted (Kendall & Gal, 2017; Malinin & Gales, 2018) and most of the time loosely stated along the lines of "While data noise uncertainty (aleatoric uncertainty) is irreducible, model uncertainty (epistemic uncertainty) vanishes with an increasing number of training observations." In other words, the width of credible bounds converges to zero whilst the width of predictive bounds does not converge to zero in the presence of data noise.

However, whilst such statements are qualitatively true, formally, D5 (Vanishing) only holds *in the limit* of the number of i.i.d training points n^{train} to infinity and for $x \in X$ that are *in the support* of the input data generating distribution. Furthermore, note that D3 (Out-of-Sample) also holds in the presence of data noise uncertainty.

Moreover, note that while D1–D4 are statements on *relative* model uncertainty, i.e., statements that are independent of the calibration parameter $c \geq 0$ (see A.2.1), D5 (Vanishing) is a statement about *absolute* model uncertainty. Thus, D5 (Vanishing) only holds for a fixed calibration parameter $c \geq 0$ (if c increases sufficiently fast with increasing n^{train}, model uncertainty does not vanish).

D.5.1. Why does D5 (Vanishing) only hold in the limit?

1. Even for fixed $c \geq 0$, in the case of large *unknown* data noise uncertainty that is simultaneously learned by the algorithm, adding another input training point x close to existing input training points x^{train} whose corresponding target y is very far away from y^{train} could lead to an increase in model uncertainty, since this new training point (x, y) would increase the predicted data noise uncertainty and thus increase the data noise induced model uncertainty.

2. Even if there is no data noise uncertainty $\sigma_{\text{n}} \equiv 0$ and $c \geq 0$ is fixed, adding another input training point can increase the model uncertainty, when D4 (Metric Learning) is fulfilled. To see this, consider the following scenario: an already observed set of training points suggest that f is very flat/simple/predictable (e.g., linear) in a certain region. However, adding a new training point (x, y) shows that f is much more irregular in this region than expected. Then, the learned metric can drastically change resulting in increased model uncertainty in this region (outside an ϵ-ball around the new input training point x).

D.5.2. How Does NOMU Fulfill D5 (Vanishing)?

Recall, that we train NOMU by minimizing

$$L^{\pi}(\mathcal{NN}_{\theta}) + \lambda \|\theta\|_2^2 \,, \tag{83}$$

where the NOMU loss $L^{\pi}(\mathcal{NN}_{\theta})$ is defined as:

$$L^{\pi}(\mathcal{NN}_{\theta}) := \underbrace{\sum_{i=1}^{n^{\text{train}}} (\hat{f}(x_i^{\text{train}}) - y_i^{\text{train}})^2}_{(a)} + \pi_{\text{sqr}} \cdot \underbrace{\sum_{i=1}^{n^{\text{train}}} (\hat{r}_f(x_i^{\text{train}}))^2}_{(b)} \tag{84}$$

$$+ \pi_{\text{exp}} \cdot \underbrace{\frac{1}{\lambda_d(X)} \int_X e^{-c_{\text{exp}} \cdot \hat{r}_f(x)} \, dx}_{(c)} \,. \tag{85}$$

Then, the following proposition holds:

Proposition D.5.a *Let $\lambda, \pi_{exp}, c_{exp}, \pi_{sqr} \in \mathbb{R}_{>0}$ be fixed and let the activation-functions of $\mathcal{NN}_{\theta}$ be Lipschitz-continuous and let $\hat{\sigma}_f(x)$ be NOMU's model uncertainty prediction. Then, it holds that $\hat{\sigma}_f(x)$ converges in probability to ℓ_{min} for $n^{train} \to \infty$ (with $x_i^{train} \overset{i.i.d}{\sim} \mathbb{P}_X$) for all input points x in the support of the input data generating distribution $\mathbb{P}_X$, i.e., more formally $\forall x \in supp\,(\mathbb{P}_X), \forall \epsilon > 0, \forall \delta \geq 0 : \exists n^0 : \forall n \geq n^0$:*

$$\mathbb{P}\left[|\hat{\sigma}_f(x) - \ell_{min}| \geq \epsilon\right] < \delta, \tag{86}$$

where $\ell_{min} \geq 0$ is an arbitrarily small constant modelling a minimal model uncertainty used for numerical reasons.

Proof. Let $x \in \text{supp}\,(\mathbb{P}_X)$, $\delta \geq 0$ and $\epsilon > 0$.

First, let $L^{\pi}(0) := c < \infty$, i.e., the value of the loss function when inserting the constant zero function. Then for the optimal solution $\mathcal{NN}_{\theta^*}$ of (83) it immediately follows that

$$\|\theta^*\|_2^2 \leq \frac{c}{\lambda}. \tag{87}$$

Using the fact that all activation-functions ϕ of $\mathcal{NN}_{\theta}^*$ are Lipschitz-continuous together with (87), one can show that there exists a constant $L := L(c, \lambda, \phi, \text{architecture})$ such that the raw model uncertainty prediction $\hat{r}_f^{\theta^*}$ is Lipschitz-continuous with constant L.

Next, let $U := U_{\frac{\epsilon}{4L}}(x)$ be an open ball with radius $\frac{\epsilon}{4L}$ around x. Given that the diameter of U is equal to $\frac{\epsilon}{2L}$ and the fact that $\hat{r}_f^{\theta^*}$ is Lipschitz-continuous with constant L, it follows that

$$\max_{z \in U} \hat{r}_f^{\theta^*}(z) - \min_{z \in U} \hat{r}_f^{\theta^*}(z) < \frac{\epsilon}{2}. \tag{88}$$

Given U let $p := \mathbb{P}[x \in U]$. Since $x \in \text{supp}\,(\mathbb{P}_X)$, per definition it holds that $p > 0$.

In the following let $D_{n,x}^{\text{train}} \sim (\mathbb{P}_X)^n$ denote the random variable representing a set of n input training points. Now, let n^0 be sufficiently large such that

$$\mathbb{P}\left[|D_{n^0,x}^{\text{train}} \cap U| > 4 \cdot \frac{c}{\epsilon^2}\right] > 1 - \delta. \tag{89}$$

Note that one can explicitly calculate the value of n^0, since $|D^{\text{train}}_{n^0,x} \cap U| \in \mathbb{N}_0$ is binomial distributed with $p > 0$ and $n^0 \in \mathbb{N}$.

Finally, we show that $\hat{r}^{\theta^*}_f(x) < \epsilon$ by contradiction. For this, assume on the contrary that

$$\hat{r}^{\theta^*}_f(x) \geq \epsilon. \tag{90}$$

Using (88) it follows that for all $z \in D^{\text{train}}_{n^0,x} \cap U$ it holds that $\hat{r}^{\theta^*}_f(z) > \frac{\epsilon}{2}$ with probability larger than $1-\delta$. This together with the fact that each summand in the term (b) in the NOMU loss function L^π is non-negative implies that

$$(b) \geq \sum_{x^{\text{train}} \in D^{\text{train}}_{n^0,x} \cap U} (\hat{r}_f(x^{\text{train}}))^2 > \left(\frac{\epsilon}{2}\right)^2 \cdot 4 \cdot \frac{c}{\epsilon^2} = c. \tag{91}$$

Putting everything together and using the fact that each term in the NOMU loss is non-negative implies that

$$L^\pi(\mathcal{NN}_{\theta^*}) + \lambda \|\theta^*\|_2^2 \geq (b) \overset{(91)}{>} c = L^\pi(0) + \lambda \|0\|_2^2,$$

which is a contradiction for $\mathcal{NN}_{\theta^*}$ being optimal in (83).

Therefore, we can conclude that $\hat{r}^{\theta^*}_f(x) < \epsilon$ with probability larger than $1-\delta$. By definition of $\hat{\sigma}_f$ this implies that $|\hat{\sigma}_f - \ell_{\text{min}}| < \epsilon$ with probability larger than $1-\delta$, which concludes the proof. □

Note that empirically one can see in Figure 13 (in the areas with many input training points) how well NOMU fulfills D5 (Vanishing) in real-world settings. Furthermore, one can see that the statement only holds true and is only desirable for x in the support of the input data generating distribution $\mathbb{P}_X$ (not in the gaps).

E. NOMU vs. Prior Networks

In this section, we highlight several differences of NOMU compared to *prior regression networks* that were recently introduced in a working paper by Malinin et al. (2020a).

While the high level idea of introducing a separate loss term for the in-sample-distribution and *out-of-distribution (OOD)* distribution is related to NOMU, there are several important differences, which we discuss next:

1. Malinin et al. (2020a)'s approach focuses on estimating both model and data noise uncertainty. Thus, to properly compare it to NOMU, we consider their approach for known and negligible data noise uncertainty, e.g., for $\sigma_{\text{n}} = 10^{-10}$ we need to set in their paper $(L, \nu) = (I \cdot l^{-1}, \frac{1}{\sigma_{\text{n}}} l)$ with $l \to \infty$, such that the their corresponding model uncertainty prediction is given by $(\kappa(x)\Lambda(x))^{-1} \overset{l\to\infty}{=} \frac{\sigma_{\text{n}}}{\kappa(x)} \cdot I$. In the following, we will consider for simplicity a one-dimensional output, i.e., $\hat{\sigma}_f = \frac{\sigma_{\text{n}}}{\kappa(x)}$.
2. They explicitly define a prior "target" distribution independent of $x \in X$, which is parametrized by κ_0 (model uncertainty) and m_0 (mean prediction) for OOD input points. Specifically, for OOD input points their mean prediction $\hat{f}$ is pushed towards m_0. In many applications the success of classical mean predictions of deep NNs is evident. In none of these applications there was a term that pushed the mean prediction to a fixed predefined prior mean. Therefore, for NOMU we keep the mean prediction untouched.
3. Instead of our loss, their loss (derived from a reverse KL-divergence) is of the form:

$$\underbrace{\sum_{i=1}^{n^{\text{train}}} \frac{\left(\hat{f}(x_i^{\text{train}}) - y_i^{\text{train}}\right)^2}{2\left(\sigma_{\text{n}}\right)^2} + \sum_{i=1}^{n^{\text{train}}} \frac{1}{\kappa(x)}}_{\text{in-sample}} + \tag{92}$$

$$\underbrace{\int_X \frac{\kappa_0\left(\hat{f}(x) - m_0\right)^2}{2\left(\sigma_{\text{n}}\right)^2} + \frac{\kappa_0}{\kappa(x)} - \log\frac{\kappa_0}{\kappa(x)} - 1 \, d\mu_{\text{OOD}}(x)}_{\text{out-of-distribution}},$$

   where m_0, κ_0 are the prior parameters for the mean and model uncertainty prediction and μ_{OOD} is an OOD measure. Specifically, they enforce zero model uncertainty at input training points via the linear term $\frac{1}{\kappa(x)}$, while we use a quadratic term. Moreover, they only use an OOD term and no *out-of-sample (OOS)* term (see below Item 4).
4. In their loss formulation in (92), they only use an *out-of-distribution* (OOD) term, while we use an *out-of-sample (OOS)* term. By OOD they refer to input training points *only* far away from the training data, e.g., in (Malinin et al., 2020a, Section 3) μ_{OOD} only has support far away from the convex hull of the input training points. Thus, they do not enforce model uncertainty in gaps between input training points. In contrast, by *out-of-sample (OOS)* we refer to a distribution with no mass on the input training points, i.e., we sample new input points that are not equal to the input training points but come from the same range (we recommend to use a distribution that is similar to the data generating process). Therefore, our loss explicitly also enforces model uncertainty in gaps between input training points.
5. They use a different architecture and train only *one* NN. This implies that their mean prediction $m(x)$ can be influenced in unwanted ways by the model uncertainty prediction $(\kappa(x)\Lambda(x))^{-1}$.
6. Their theoretical motivation substantially differs from ours: They only partially specify a prior belief by defining *marginal* distributions for $f(x)$ for each input point

$x \in X$, without specifying a joint prior distribution for f. However, given only marginals no joint distribution, which is the crucial aspect when defining a prior in regression, can be derived without further information (E.g., consider Gaussian processes (GPs); here all one-dimensional marginal distributions are simply given by $\mathcal{N}((m(x)), k(x,x))$. However, the crucial part is how to define $k(x, x^{'})$ specifying the relation of x and $x^{'}$. Only defining the marginals does not suffice to fully define GPs, leaving this most crucial part undefined).
However, for NOMU, we give in Appendix A a theoretical connection to BNNs, with Gaussian prior on the weights. This induces a prior on the function space, i.e., a distribution over f rather than separate *marginal* distributions over $f(x)$ for each $x \in X$.

7. Parametrizing the model precision instead of the model uncertainty can have negative effects due to (implicit) regularization of NNs in the case of negligible or zero data noise. To get uncertainties in gaps between the input training points (small $\kappa(x)$) while having almost zero uncertainty at these input training points ($\kappa(x) \to \infty$), would imply very high regularization costs for the function $\kappa(x)$ and thus is very hard to learn for a NN. For NOMU, we therefore parameterize directly the model uncertainty (which is always finite) instead of the model precision (that should be infinite at noiseless training data points).

8. Our experimental results suggest that NOMU clearly outperforms DE in BO, whilst DE outperforms *prior regression networks* in their set of experiments.

F. NOMU vs. Neural Processes

In this section, we discuss the differences between neural processes (NPs) introduced by Garnelo et al. (2018a;b) and NOMU. Specifically, we explain in the following why NPs and NOMU are solving very different problems in different settings.

For training NPs, one has to observe data from 1000s of realizations of f_k, sampled i.i.d. from the prior distribution (for each f_k one observes x_i and $f_k(x_i)$ to train the NP). This is often mentioned in Garnelo et al. (2018a;b), e.g., Garnelo et al. (2018a, Section 4.1): *"We generate [...] datasets that consist of functions generated from a GP [...]* ***At every training step we sample a curve from the GP*** *[...]."*. For NOMU we consider the **very different** task of estimating $p(y^{\text{test}}|x^{\text{test}}; D^{\text{train}})$ based on a **single dataset**, i.e, generated from a single realization $f = f_1$.

For example in the case of the Boston housing data set from Section 4.1.4, there is only one function $f = f_1$ involved that maps a (multidimensional) input data point x corresponding to a house in Boston to its price $f(x)$. For this data set, NPs would not be well suited, since it only contains data $(x_i, y_i) = (x_i, f(x_i) + \varepsilon_i)$ coming from this specific function f. One does not have access to data corresponding to another function f_2 that had been sampled from the same prior distribution.

The same is true for the other data sets we consider in this paper (e.g., for the solar irradiance data set we only use the data visible in Figure 5 and NOMU does not have access to any data coming from other time series to make its predictions). Thus, NPs cannot be applied to the tasks considered in this paper.

Summary. NOMU, GP, MCDO, DE and HDE are designed to be trained on data coming from *one* unknown function f without having access to data from other functions $f_2, f_3, \dots$. In contrast, NPs are designed to be trained on *multiple* data sets generated from *multiple* functions $f_1, f_2, f_3, \dots$.

G. Aleatoric Neural Networks: Aleatoric vs. Epistemic Uncertainty

In this section, we discuss the classical approach of a NN with two outputs, one output for a model prediction and another for aleatoric uncertainty, which is trained using the (scaled) Gaussian negative log-likelihood as introduced by Nix & Weigend (1994). We will use the terms aleatoric uncertainty and data noise as well as model uncertainty and epistemic uncertainty interchangeably.

In what follows, we call this method *aleatoric neural network (ANN)*. Within this section, we show that such an ANN does not explicitly estimate model uncertainty (in contrast to all other benchmark methods discussed in this paper), i.e., when using the aleatoric uncertainty output $\hat{\sigma}_{\text{n}}$ naively as $\hat{\sigma}_f$, the so obtained $\hat{\sigma}_f := \hat{\sigma}_{\text{n}}$ does not represent model uncertainty (epistemic uncertainty).[40] First, we give the definition of an ANN.

Definition G.1 (ANN) *An ANN is a fully-connected feed-forward NN $\mathcal{NN}^{\text{ANN}} : \mathbb{R}^d \to \mathbb{R} \times \mathbb{R}_+$ with two outputs: (i) the model prediction $\hat{f} \in \mathbb{R}$ and (ii) a model uncertainty prediction $\hat{\sigma}_n \in \mathbb{R}_+$ that is trained for a given set of training points D^{train} using the following loss function:*

$$L^{\text{ANN}}(\mathcal{NN}^{\text{ANN}}) := \frac{1}{|D^{\text{train}}|} \sum_{(x,y) \in D^{\text{train}}} \left[\frac{\left(y - \hat{f}(x)\right)^2}{2\left(\hat{\sigma}_n(x)\right)^2} + \ln\left(\hat{\sigma}_n(x)\right) \right] \tag{93}$$

The idea of ANN is to estimate the noise scale $\sigma_{\text{n}}(x) = \sqrt{\mathbb{V}[\varepsilon]} = \sqrt{\mathbb{V}[y|x, f(x)]}$. Figure 21 shows that, as we

[40] The reminder of this section only targets readers who do not directly see that substituting $\hat{\sigma}_f$ by $\hat{\sigma}_{\text{n}}$ is an extremely bad idea.

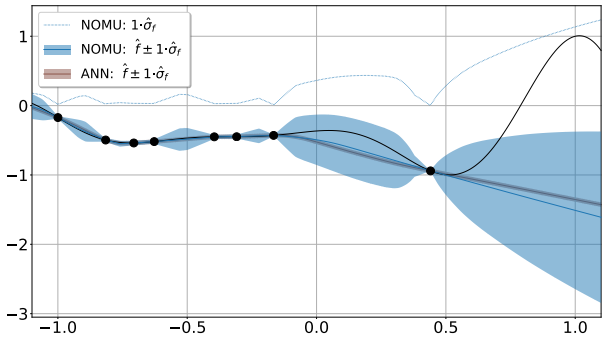


Figure 21. Comparison of UBs resulting from ANN ($c = 1$) (where $\hat{\sigma}_\text{n}$ is used as a substitute for $\hat{\sigma}_f$) vs. NOMU for the Forrester function (solid black line). For NOMU, we also show $\hat{\sigma}_f$ as a dotted blue line. Training points are shown as black dots.

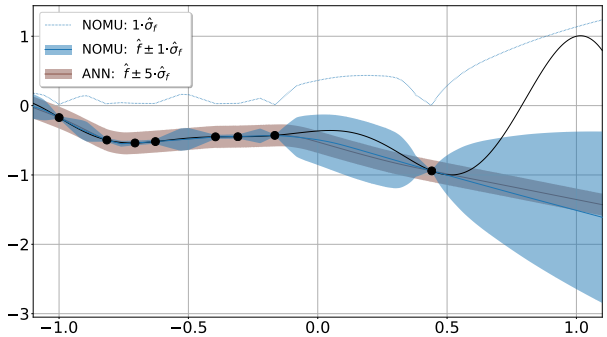


Figure 22. Comparison of UBs resulting from ANN ($c = 5$) (where $\hat{\sigma}_\text{n}$ is used as a substitute for $\hat{\sigma}_f$) vs. NOMU for the Forrester function (solid black line). For NOMU, we also show $\hat{\sigma}_f$ as a dotted blue line. Training points are shown as black dots.

would expect, the trained ANN has learned the true $\sigma_\text{n} \equiv 0 \approx \hat{\sigma}_\text{n}$ quite precisely. However, it as also becomes evident that the ANN does not learn any form of model uncertainty. Very far away from all observed training data points, the ANN does not express any uncertainty about its prediction (in Figure 22, to the right of $x = 0.5$, the predictions are very far away from the truth, but $\hat{\sigma}_\text{n}$ does not capture this uncertainty). Therefore, the ANN's aleatoric uncertainty output $\hat{\sigma}_\text{n}$ does not fulfill desideratum D3 (Out-of-Sample). The problem of misusing $\hat{\sigma}_\text{n}$ as substitute for σ_f is not that $\hat{\sigma}_\text{n}$ is too small (as we study relative uncertainty in this paper (see Appendix A.2.1), one can always scale the uncertainty by a factor c). However, Figure 22 shows that also the scaled uncertainty completely fails to capture the desiderata, i.e., the aleatoric uncertainty output $\hat{\sigma}_\text{n}$ is almost constant for all input points x. Thus, UBs of an ANN do not fulfill D2 (In-Sample) and D3 (Out-of-Sample). In Figure 22, we can see that $5\hat{\sigma}_\text{n}$ is way too underconfident at input training points and at the same time way too overconfident far away from the observed input training points. This would result in very bad NLL scores on a test set. (Moreover in noisy settings, i.e., $\sigma_\text{n} \neq 0$, the aleatoric uncertainty output $\hat{\sigma}_\text{n}$ of an ANN does not fulfill D5 (Vanishing) either: $\hat{\sigma}_\text{n}$ should converge to σ_n while $\hat{\sigma}_f$ should converge to zero as the amount of training data increases.)

Furthermore, especially in Bayesian optimization (BO) $c\hat{\sigma}_\text{n}$ would be a terribly bad substitute for σ_f: Maximizing the upper UB acquisition function $\hat{f} + c\hat{\sigma}_\text{n}$, would be almost equivalent to maximizing $\hat{f}$ since $c\hat{\sigma}_\text{n}$ is almost constant because of the lack of D2 (In-Sample) and D3 (Out-of-Sample). If one wants to maximize the function in Figure 22 on $[-1, 1]$, the next BO-step would propose to query an input training point at the left boundary $x = -1$ (even for large c). However, one does not learn anything new from evaluating at $x = -1$, because this input training point has already been evaluated in a previous BO-step. All subsequent BO steps would propose to query the same point $x = -1$ without exploring any other region of the input space and one would never find the true maximum at $x \approx 1$. In contrast, a reasonable model for estimating σ_f (such as NOMU), would directly (after scaling up c dynamically by a factor 2 as described in Appendix B.3.2) choose a point in the unexplored right region $x \approx 1$, because the left side $x \approx -1$ is already well explored.

Overall, aleatoric uncertainty σ_n and epistemic uncertainty σ_f are two very different objects. Thus, an estimator $\hat{\sigma}_\text{n}$ designed to estimate σ_n is usually a bad estimator for σ_f.

H. Hyperparameter Sensitivity Analysis

In this section, we provide a sensitivity analysis with respect to NOMU's loss hyperparameters, i.e., π_sqr, π_exp, c_exp, and D^art. First, we present a visual qualitative analysis in 1D showing how each of these hyperparameters affects the shape of NOMU's UBs (Appendix H.1). Second, we also present an extensive quantitative sensitivity analysis in the generative test-bed setting from Section 4.1.2, where in addition to the loss hyperparameters we also include the hyperparameters of the readout map ℓ_min, and ℓ_max in our analysis (Appendix H.2).

H.1. Qualitative Sensitivity Analysis

In this section, we consider the setting of Section 4.1.1, and visualize the effect of increasing or decreasing each of NOMU's loss hyperparameters π_sqr, π_exp, c_exp, and D^art on the example of the 1D Levy function. For reference, Figure 23 shows NOMU's UBs (with scaling factor $c = 2$) for the default loss hyperparameters $\pi_\text{sqr} = 0.1$, $\pi_\text{exp} = 0.01$, $c_\text{exp} = 30$, and $D^\text{art} = 128$ that are used in Section 4.1.1 in the main paper.

For each of D^art and c_exp, we fit two additional NOMU mod-

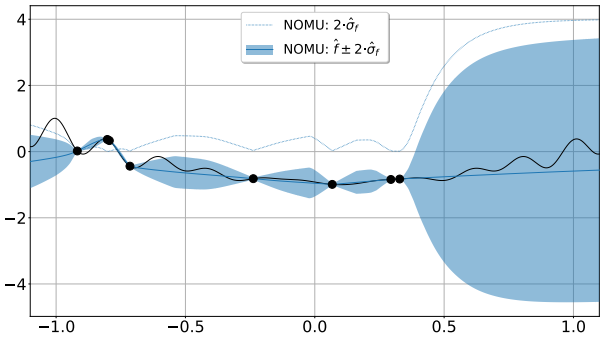


Figure 23. NOMU's UBs (c=2) for the generic loss hyperparameters from Section 4.1.1 in the main paper.

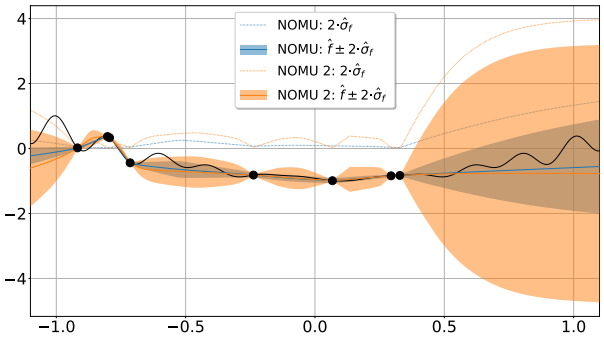


Figure 24. NOMU's UBs (c=2) for $\pi_{\text{sqr}} = 1e-4, \pi_{\text{exp}} = 1e-5$ (blue) and $\pi_{\text{sqr}} = 1, \pi_{\text{exp}} = 0.1$ (orange).

els, where we ceteris paribus de- and increase the hyperparameter's default value by factors $1/s$ and s, respectively. The multiplicative factors π_{sqr} and π_{exp} we treat jointly in our sensitivity analysis: First, we show the effect of ceteris paribus de- and increasing the default value of the product $\pi_{\text{exp}}\pi_{\text{sqr}}$ by factors $s_l = 0.001$ and $s_u = 10$. Second, we vary the ratio $\pi_{\text{exp}}/\pi_{\text{sqr}}$ in the same fashion, with scaling factors $s_l = 1/16$ and $s_u = 16$. Within all of the experiments in this section, we sample artificial input points D^{art} on an equidistant (deterministic) grid on $[-1.1, 1.1]$. This allows us to give a qualitative analysis of the hyperparameters' effects as follows.

Varying $\pi_{\text{exp}}\pi_{\text{sqr}}$ with Scaling Factors of $s_l = 0.001$ and $s_u = 10$. Decreasing $\pi_{\text{exp}}\pi_{\text{sqr}}$ by decreasing both π_{exp} and π_{sqr} leads to more tubular bounds by relaxing the desiderata D2 (In-Sample) and D3 (Out-of-Sample). This can be seen in Figure 24: NOMU's blue dashed uncertainty (corresponding to small $\pi_{\text{exp}}\pi_{\text{sqr}}$) is larger at data points than the orange one of NOMU 2 (corresponding to large $\pi_{\text{exp}}\pi_{\text{sqr}}$), and it is smaller further away from the training data points. NOMU's default hyperparameters are in a range where the loss is already at its limit[41] enforcing the desiderata D2 (In-Sample) and D3 (Out-of-Sample). Therefore, further increasing $\pi_{\text{exp}}\pi_{\text{sqr}}$ (while keeping their ratio and the other hyperparameters fixed) barely causes the UBs to change as can be seen for the orange UBs in Figure 24. Increasing $\pi_{\text{exp}}\pi_{\text{sqr}}$ too much, can lead to numerical instabilities.

Varying $\pi_{\text{exp}}/\pi_{\text{sqr}}$ with Scaling Factors of $s_l = 1/16$ and $s_u = 16$. Increasing the ratio of $\pi_{\text{exp}}/\pi_{\text{sqr}}$ (while keeping their product and all other hyperparameters fixed) simply causes NOMU's UBs to uniformly widen across the entire domain. Indeed, in Figure 25, the orange UBs of NOMU 2 (corresponding to large $\pi_{\text{exp}}/\pi_{\text{sqr}}$) are blown up and cover the blue UBs (corresponding to small $\pi_{\text{exp}}/\pi_{\text{sqr}}$) throughout the input space.

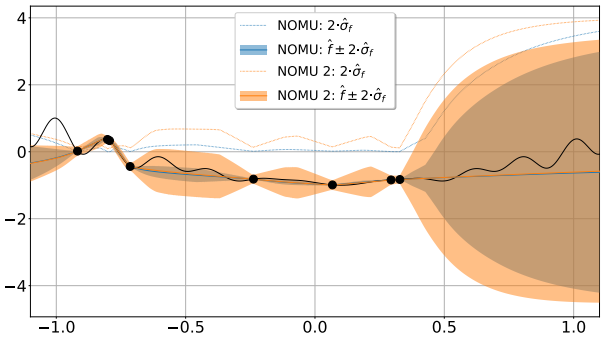


Figure 25. NOMU's UBs (c=2) for $\pi_{\text{exp}} = 0.0025, \pi_{\text{sqr}} = 0.4$ (blue) and $\pi_{\text{exp}} = 0.04, \pi_{\text{sqr}} = 0.025$ (orange).

Varying c_{exp} with a Scaling Factor of $s = 2$. Increasing the hyperparameter c_{exp} causes UBs to shrink in areas of large uncertainty and to widen in areas of small uncertainty. This effect is visualized in Figure 26: the orange dashed uncertainty line of NOMU 2 (large $c_{\text{exp}} = 60$) lies above the blue one of NOMU (small $c_{\text{exp}} = 15$) at data points; and in regions of large uncertainty, the orange UBs (corresponding to large c_{exp}) turn out to be more narrow than the blue ones (corresponding to small c_{exp}). Thus, increasing c_{exp} causes NOMU's UBs to be more tubular.

Varying D^{art} with a Scaling Factor of $s = 8$. Finally, we assess the effect of changing the number of artificial data

[41] For NOMU's default parameters the ratio $\pi_{\text{exp}}\pi_{\text{sqr}}/\lambda$ is already very large such that the explicit regularization via $\lambda \|\theta\|_2^2$ is almost negligible. Thus, the UBs are only prevented from having even larger curvature by implicit regularization, i.e., within a given number of epochs the training algorithm cannot reach a function with more curvature, because increasing the amplitude of the loss is (partially) compensated by the adaptivity of the ADAM algorithm. Only in ranges where $\pi_{\text{exp}}\pi_{\text{sqr}}/\lambda$ is small enough for the explicit regularization to actually matter, the UBs become sensitive to the ratio $\pi_{\text{exp}}\pi_{\text{sqr}}/\lambda$. Then the regularization of $\lambda \|\theta\|_2^2$ keeps the curvature of $\hat{\sigma}_f$ low, i.e., the UBs become more tubular.

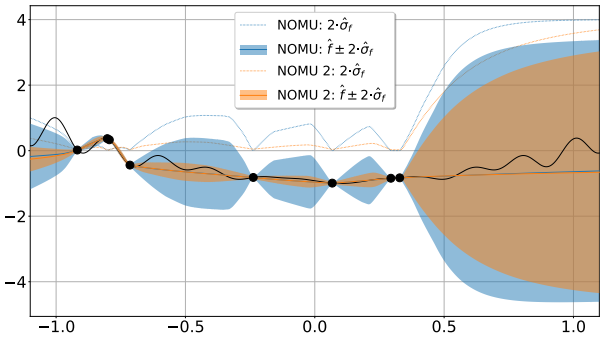


Figure 26. NOMU's UBs (c=2) for $c_{\text{exp}} = 15$ (blue) and $c_{\text{exp}} = 60$ (orange).

points D^{art} used to approximate the integral (c) of NOMU's loss function defined in Equation (4). As expected, the UBs behave overall very similarly. However, for very small gaps in between input training points, D^{art} can have an influence on the estimated UBs. For example, in the gap between the training input point at $x \approx -0.77$ and the one at $x \approx -0.73$, the $\hat{\sigma}_f$ obtained from the smaller D^{art} (blue) vanishes (because of a lack of artificial data points falling in this gap), while $\hat{\sigma}_f$ obtained from the larger D^{art} also estimates non-zero model uncertainty in this small gap.

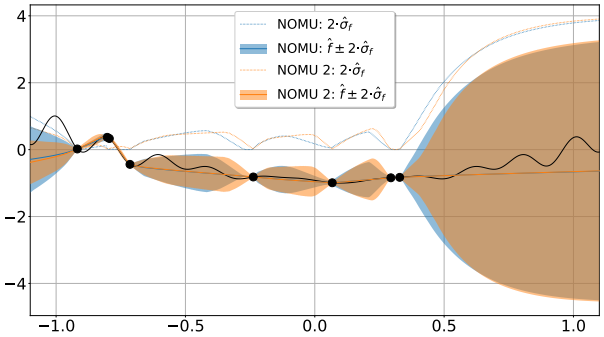


Figure 27. NOMU's UBs (c=2) for $D^{\text{art}} = 16$ (blue) and $D^{\text{art}} = 1024$ (orange).

Varying the architecture and L2-regularization In Appendix D.4.1, we visualize how certain changes to the architecture and different L2-regularization parameters λ for different parts of the network influence NOMU's model uncertainty estimate $\hat{\sigma}_f$. In particular, we show in Appendix D.4.1 how the choice of the architecture and the L2-regularization determine the degree to which NOMU fulfills desiderata D4 (Metric Learning).

Finally, in Appendix H.2, we empirically show that NOMU is robust with respect to its hyperparameters within a certain range.

H.2. Quantitative Sensitivity Analysis

We now present an extensive quantitative sensitivity analysis of NOMU's loss hyperparameters: π_{sqr}, π_{exp}, c_{exp}, and D^{art} in the generative test-bed setting (see Section 4.1.2 for details on this setting). Additionally, we also consider in this analysis the hyperparameters corresponding to the readout map, i.e., ℓ_{min} and ℓ_{max}. We decided to perform the quantitative sensitivity analysis in the generative test-bed setting, since it offers a particularly rich variety of different test functions and thus exposes each hyperparameter selection to hundreds of different test functions.

Setting We use the same default hyperparameters as in Section 4.1.2. This includes NOMU's loss hyperparameters: π_{sqr}, π_{exp}, c_{exp}, and D^{art}, the hyperparameters of the readout map: ℓ_{min} and ℓ_{max} as well as all other hyperparameters. For the following sensitivity analysis, we then vary at each time only a *single* hyperparameter, i.e., one of π_{sqr}, π_{exp}, c_{exp}, D^{art}, ℓ_{min}, and ℓ_{max}, and set all other hyperparameters to their default values (i.e., we perform a *ceteris paribus* analysis as in Appendix H.1).

In Table 15, we present for each considered hyperparameter a grid of five different values which we use to test its sensitivity. The **NOMU** column in Table 15 corresponds to the NOMU's default hyperparameters used in the generative test-bed setting (Section 4.1.2). The columns **NOMU1** to **NOMU4** in Table 15 then correspond to deviations from these original hyperparameters.

Results In Table 16, Table 17, and Table 18 we present for each of the ceteris paribus runs average NLL values for input dimensions 1D, 2D, and 5D, respectively. Each cell in those tables represents a single hyperparameter selection where we use NOMU's default hyperparameters except for the hyperparameter of the corresponding row which we choose according to the cell's column, e.g., to obtain the result for the cell (π_{exp}, **NOMU3**), we use the default NOMU hyperparameters from Section 4.1.2 *except* for π_{exp}

Table 15. Grid selection for each hyperparameter (HP). The column NOMU corresponds to NOMU's default hyperparameters used in the generative test-bed setting. NOMU1 to NOMU4 correspond to deviations from these default hyperparameters. For D^{art}, d denotes the input dimension.

HP	NOMU1	NOMU2	NOMU	NOMU3	NOMU4
π_{SQR}	0.01	0.02	0.1	0.5	1
π_{EXP}	0.001	0.002	0.01	0.05	0.1
c_{EXP}	10	20	30	45	90
D^{ART}	$\frac{100 \cdot d}{4}$	$\frac{100 \cdot d}{2}$	$100 \cdot d$	$2 \cdot (100 \cdot d)$	$4 \cdot (100 \cdot d)$
ℓ_{MIN}	0.01	0.05	0.1	0.15	0.20
ℓ_{MAX}	0.50	0.75	1	2.0	4.0

which we set according to column **NOMU3** in Table 15 to $\pi_{\text{exp}} := 0.05$. Overall, we can make the following three main observations:

1. The majority of all cells in Table 16, Table 17, and Table 18 are marked in grey. This shows that, their corresponding hyperparameters lead to average NLL results which are statistically on par with the results obtained via NOMU's default hyperparameters in the **NOMU** columns. This highlights NOMU's robustness with respect to all considered hyperparameters within the chosen grids. Furthermore, it confirms our claim from the main paper that using generic hyperparameters for NOMU often works well without much hyperparameter-tuning.

2. Since NOMU with the default hyperparameters already outperforms *all* other considered benchmark methods in this setting, i.e., each **NOMU** column represents the winning method among benchmarks (see Table 2), we see that all grey marked deviations of hyperparameters lead to results that outperform all other considered benchmark methods too. Moreover, all except for one (5D (ℓ_{min}, **NOMU1**)) NOMU models corresponding to cells which are not marked in grey, i.e., with hyperparameters that lead to statistically worse results than NOMU's default hyperparameters, are as good or better than the best benchmark methods from Table 2.

3. By varying NOMU's hyperparameters, we can even obtain better results (i.e., with a smaller average NLL) than the ones reported in Table 2 of the main paper, e.g., in 1D with $\ell_{\text{min}} := 0.01$ the average NLL $= -1.83 < -1.65$. While these improvements are not statistically significant, these results still suggest that systematic hyperparameter-tuning could improve the performance of NOMU even further.

Table 16. Sensitivity analysis for 1D generative test-bed setting. We present for each hyperparameter (HP) and its five corresponding grid-points the average NLL (without const. $\ln(2\pi)/2$) and a 95% CI over 200 BNN samples. Results which are statistically on par with NOMU's default HPs, i.e., the column **NOMU**, are marked in grey. Note that, the best benchmark method for this experiment is GP with NLL $= -1.08 \pm 0.22$ (see Table 2).

HP	NOMU1	NOMU2	NOMU	NOMU3	NOMU4
π_{SQR}	-1.59±0.11	-1.63±0.10	-1.65±0.10	-1.55±0.13	-1.54±0.14
π_{EXP}	-1.64±0.09	-1.67±0.09	-1.65±0.10	-1.62±0.10	-1.60±0.10
c_{EXP}	-1.77±0.12	-1.73±0.10	-1.65±0.10	-1.49±0.12	-1.11±0.13
D^{ART}	-1.65±0.09	-1.62±0.11	-1.65±0.10	-1.63±0.12	-1.65±0.10
ℓ_{MIN}	-1.83±0.09	-1.72±0.10	-1.65±0.10	-1.58±0.10	-1.47±0.12
ℓ_{MAX}	-1.49±0.12	-1.60±0.10	-1.65±0.10	-1.63±0.16	-1.66±0.14

Table 17. Sensitivity analysis for 2D generative test-bed setting. We present for each hyperparameter (HP) and its five corresponding grid-points the average NLL (without const. $\ln(2\pi)/2$) and a 95% CI over 200 BNN samples. Results which are statistically on par with NOMU's default HPs, i.e., the column **NOMU**, are marked in grey. Note that, the best benchmark method for this experiment is DE with NLL $= -0.77 \pm 0.07$ (see Table 2).

HP	NOMU1	NOMU2	NOMU	NOMU3	NOMU4
π_{SQR}	-1.18±0.04	-1.18±0.04	-1.16±0.05	-1.15±0.04	-1.15±0.04
π_{EXP}	-1.15±0.04	-1.15±0.05	-1.16±0.05	-1.18±0.04	-1.18±0.04
c_{EXP}	-1.17±0.04	-1.19±0.04	-1.16±0.05	-1.11±0.05	-1.00±0.05
D^{ART}	-1.16±0.05	-1.16±0.05	-1.16±0.05	-1.16±0.04	-1.15±0.05
ℓ_{MIN}	-1.07±0.05	-1.17±0.04	-1.16±0.05	-1.14±0.05	-1.12±0.05
ℓ_{MAX}	-1.13±0.05	-1.15±0.05	-1.16±0.05	-1.16±0.04	-1.16±0.04

Table 18. Sensitivity analysis for 5D generative test-bed setting. We present for each hyperparameter (HP) and its five corresponding grid-points the average NLL (without const. $\ln(2\pi)/2$) and a 95% CI over 200 BNN samples. Results which are statistically on par with NOMU's default HPs, i.e., the column **NOMU**, are marked in grey. Note that, the best benchmark method for this experiment is GP with NLL $= -0.33 \pm 0.02$ (see Table 2).

HP	NOMU1	NOMU2	NOMU	NOMU3	NOMU4
π_{SQR}	-0.37±0.03	-0.37±0.02	-0.37±0.02	-0.36±0.02	-0.35±0.02
π_{EXP}	-0.33±0.02	-0.34±0.02	-0.37±0.02	-0.40±0.02	-0.40±0.02
c_{EXP}	-0.39±0.02	-0.38±0.02	-0.37±0.02	-0.37±0.02	-0.34±0.05
D^{ART}	-0.37±0.02	-0.37±0.02	-0.37±0.02	-0.37±0.02	-0.37±0.02
ℓ_{MIN}	-0.21±0.03	-0.33±0.02	-0.37±0.02	-0.38±0.02	-0.39±0.02
ℓ_{MAX}	-0.39±0.02	-0.37±0.02	-0.37±0.02	-0.35±0.02	-0.33±0.02

I. Details on our Notation

In Section 2, we are using a slightly overloaded notation, where we use the same symbol f for different mathematical objects. Sometimes, we use f for a function-valued random variable $F : (\Omega, \Sigma, \mathbb{P}) \to Y^X$. and sometimes we use f for the specific unknown ground truth function $f_{\text{true}} := F(\omega)$ (i.e., $\forall x \in X : f_{\text{true}}(x) = (F(\omega))(x)$). While we used this slightly overloaded notation for the sake of readability in the main paper, we will now introduce our Bayesian uncertainty framework in its full mathematical detail:

In practice, there exists an unknown ground truth function f_{true}. In the classical Bayesian paradigm, one assumes that everything unknown (i.e., here f_{true}) was sampled from a prior distribution. Specifically, $f_{\text{true}} := F(\omega)$ is a realization of a random variable $F : (\Omega, \Sigma, \mathbb{P}) \to Y^X$ distributed according to a prior belief, i.e., a prior distribution. Using this notation, one can mathematically describe in a rigorous way the full Bayesian data generating process as follows:

Let $F : (\Omega, \Sigma, \mathbb{P}) \to Y^X$ be a function-valued random variable distributed according to a prior belief, i.e., a prior

distribution. Moreover, let $(\mathcal{X}_i, \mathcal{E}_i)_{i\in\{1,\dots,n^{\text{train}}\}}$ denote the random variable representing the input data points and the corresponding data noise, i.e., a family off i.i.d random variables independent of F, where each random variable $(\mathcal{X}_i, \mathcal{E}_i) : (\Omega, \Sigma, \mathbb{P}) \to X \times \mathbb{R}$ fulfills $\forall x \in X : (\mathcal{E}_i|\mathcal{X}_i = x) \sim \mathcal{N}(0, \sigma_{\text{n}}(x))$. Finally, let $\mathcal{Y}_i$ be the random variable associated to the targets, which we define as $\mathcal{Y}_i : (\Omega, \Sigma, \mathbb{P}) \to Y, \omega \mapsto \mathcal{Y}_i(\omega) := F(\omega)(\mathcal{X}_i(\omega)) + \mathcal{E}_i(\omega)$.

With this notation in place, the objects used in the main paper in Section 2: $x_i = \mathcal{X}_i(\omega)$, $y_i = \mathcal{Y}_i(\omega)$ and $\varepsilon_i = \mathcal{E}_i(\omega)$ are the values one actually observe in practice as training data, i.e., the realizations of the data generating process.

Therefore, $\sigma_f(x)$ from Equation (1) should be interpreted for all $x \in X$ as

$$\sigma_f(x) = \sqrt{\mathbb{V}\left[F(\cdot)(x)|\forall i \in \{1,\dots,n^{\text{train}}\} : (\mathcal{X}_i, \mathcal{Y}_i) = (x_i, y_i)\right]}, \tag{94}$$

which, for all $x \in X$, is defined mathematically even more rigorously via the conditional variance as follows:

$$\sqrt{\mathbb{V}\left[F(\cdot)(x)\middle|(\mathcal{X}_i, \mathcal{Y}_i)_{i\in\{1,\dots,n^{\text{train}}\}}\right](\omega)}. \tag{95}$$

Throughout the paper it should always be clear from the context if x_i refers to the random variable $\mathcal{X}_i$ or its realization $\mathcal{X}_i(\omega)$ where the same holds true for y_i, ε_i and f.

Importantly, note that in the setting which we consider in this paper, one only observes data coming from a *single* function $f = F(\omega)$ and one does not have access to more functions f_i. Specifically, we neither have access to other samples of the random variable F nor do we consider multiple i.i.d random variables F_i in contrast to the setting considered for neural processes as described in Appendix F.

J. Visualization of the Readout Map

In this section, we provide in Figure 28 a visualization of the readout map

$$\varphi(z) = \ell_{\text{max}}\left(1 - \exp\left(-\frac{\max(0, z) + \ell_{\text{min}}}{\ell_{\text{max}}}\right)\right), \tag{96}$$

for $\ell_{\text{min}} \geq 0$ and $\ell_{\text{max}} > 0$. The readout map φ was introduced in Section 3.2 to transform the *raw* model uncertainty output $\hat{r}_f$ into NOMU's *model uncertainty prediction* $\hat{\sigma}_f(x) = \varphi(\hat{r}_f(x))$ (see Equation (2)).

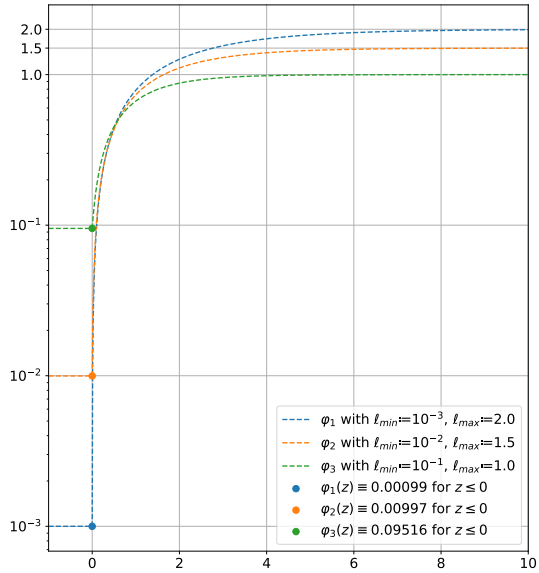


Figure 28. Visualization of the readout map $\varphi(z)$ for three different $(\ell_{\text{min}}, \ell_{\text{max}})$-pairs.

Figure 28 shows how the readout map φ interpolates between its minimal value $\ell_{\text{max}}(1-\exp(-\frac{\ell_{\text{min}}}{\ell_{\text{max}}})) \approx \ell_{\text{max}}(1-1+\frac{\ell_{\text{min}}}{\ell_{\text{max}}})) = \ell_{\text{min}}$ (shown in Figure 28 for three different pairs of $(\ell_{\text{min}}, \ell_{\text{max}})$ as the three dots) and its asymptotic maximal value $\ell_{\text{max}} = \lim_{z\to\infty} \varphi(z)$. Specifically, we designed the readout map φ such that it has a relatively steep increase starting from 0 and flattens when asymptotically converging to its maximal value ℓ_{max}. The parameter ℓ_{min} is used for numerical stability to prevent 0 model uncertainty outputs, which would be a very extreme statements, causing metrics such as NLL or AUC to result in infinitely bad values for arbitrarily small numerical deviations of the model predictions. The parameter ℓ_{max} defines the maximal model uncertainty far away from input training points (similarly to the prior variance for GPs with RBF kernel). As discussed in Remark 3.2, the readout map φ can be modified depending on the subsequent use of the estimated UBs or it can even be learned by parameterizing φ by a NN and minimizing a valid metric (such as the NLL) on a validation set.

5 Monotone-Value Neural Networks: Exploiting Preference Monotonicity in Combinatorial Assignment

The content of this chapter has previously appeared in

Monotone-Value Neural Networks: Exploiting Preference Monotonicity in Combinatorial Assignment.
Jakob Weissteiner*, Jakob Heiss*, Julien Siems* and Sven Seuken.
In *Proceedings of the Thirty-first International joint Conference on Artificial Intelligence* (IJCAI'22), Vienna, AUT, July 2022.

For its full updated version including appendix, please see

Monotone-Value Neural Networks: Exploiting Preference Monotonicity in Combinatorial Assignment.
Jakob Weissteiner*, Jakob Heiss*, Julien Siems* and Sven Seuken.
Working paper, March 2023.
URL: arxiv.org/pdf/2109.15117.pdf

*These authors contributed equally.

Monotone-Value Neural Networks: Exploiting Preference Monotonicity in Combinatorial Assignment*

Jakob Weissteiner[1,3†], **Jakob Heiss**[2,3†], **Julien Siems**[1†] and **Sven Seuken**[1,3]
[1]University of Zurich
[2]ETH Zurich
[3]ETH AI Center
weissteiner@ifi.uzh.ch, jakob.heiss@math.ethz.ch, juliensiems@gmail.com, seuken@ifi.uzh.ch

Abstract

Many important resource allocation problems involve the *combinatorial assignment* of items, e.g., auctions or course allocation. Because the bundle space grows exponentially in the number of items, preference elicitation is a key challenge in these domains. Recently, researchers have proposed ML-based mechanisms that outperform traditional mechanisms while reducing preference elicitation costs for agents. However, one major shortcoming of the ML algorithms that were used is their disregard of important prior knowledge about agents' preferences. To address this, we introduce *monotone-value neural networks (MVNNs)*, which are designed to capture combinatorial valuations, while enforcing *monotonicity* and *normality*. On a technical level, we prove that our MVNNs are universal in the class of monotone and normalized value functions, and we provide a mixed-integer linear program (MILP) formulation to make solving MVNN-based winner determination problems (WDPs) practically feasible. We evaluate our MVNNs experimentally in spectrum auction domains. Our results show that MVNNs improve the prediction performance, they yield state-of-the-art allocative efficiency in the auction, and they also reduce the run-time of the WDPs. Our code is available on GitHub: https://github.com/marketdesignresearch/MVNN.

1 Introduction

Many important economic problems involve the *combinatorial assignment* of multiple indivisible items to multiple agents. In domains *with money*, prominent examples include *combinatorial auctions (CAs)* and *combinatorial exchanges (CEs)*. In CAs, heterogeneous items are allocated amongst a set of bidders, e.g., for the sale of spectrum licenses [Cramton, 2013]. In CEs, a set of items is allocated between multiple agents who can be sellers *and* buyers at the same time, e.g., for the reallocation of catch shares [Bichler *et al.*, 2019]. In domains *without money*, a popular example is *combinatorial course allocation*, where course seats are allocated to students at large business schools [Budish, 2011].

What all of these domains have in common is that the agents can report their values on *bundles* of items rather than only on individual items. This allows them to express more complex preferences, i.e., an agent's valuation of a bundle is not simply the sum of each individual item's value, but it can be more (*complementarity*) or less (*substitutability*). However, since the bundle space grows exponentially in the number of items, agents cannot report values for all bundles in settings with more than a modest number of items. Thus, parsimonious *preference elicitation* is key for the design of practical combinatorial assignment mechanisms.

In this paper, we present a new machine learning approach that exploits prior (structural) knowledge about agents' preferences and can be integrated well into iterative market mechanisms. Our contribution applies to any combinatorial assignment problem. However, we present our algorithms in the context of a CA specifically, to simplify the notation and because there exist well-studied preference generators for CAs that we can use for our experimental evaluation.

For CAs with general valuations, Nisan and Segal [2006] have shown that exponential communication in the number of items is needed in the worst case to find an optimal allocation of items to bidders, i.e., to ensure full efficiency. Thus, for general valuations, practical CAs cannot provide efficiency guarantees in large domains. In practice, *iterative combinatorial auctions (ICAs)* are employed, where the auctioneer interacts with bidders over multiple rounds, eliciting a *limited* amount of information, aiming to find a highly efficient allocation. ICAs are widely used, e.g., for the sale of licenses to build offshore wind farms [Ausubel and Cramton, 2011]. The provision of spectrum licenses via the *combinatorial clock auction (CCA) [Ausubel* et al.*, 2006]* has generated more than $20 billion in total revenue [Ausubel and Baranov, 2017]. Therefore, increasing the efficiency of such real-world ICAs by only 1% point translates into monetary gains of hundreds of millions of dollars.

1.1 ML-based Auction Design

In recent years, researchers have successfully integrated machine learning (ML) algorithms into the design of CAs to im-

*This paper is the slightly updated version of Weissteiner *et al.* [2022a] published at IJCAI'22 including the appendix.

†These authors contributed equally to this paper.

prove their performance. Dütting *et al.* [2019] and Rahme *et al.* [2021] used neural networks (NNs) to learn entire auction mechanisms from data, following the automated mechanism design paradigm. Brero *et al.* [2019] studied a Bayesian ICA using probabilistic price updates to achieve faster convergence to an efficient allocation.

Most related to this paper is the work by Brero *et al.* [2018; 2021], who developed a value-query-based ICA that achieves even higher efficiency than the widely used CCA. In follow-up work, Weissteiner and Seuken [2020] extended their work by integrating neural networks (NN) in their mechanisms and could further increase the efficiency. Finally, Weissteiner *et al.* [2022b] used Fourier transforms (FTs) to leverage different notions of sparsity of value functions in preference elicitation. However, despite these advances, it remains a challenging problem to find the efficient allocation while keeping the elicitation cost for bidders low. Even state-of-the-art approaches suffer from significant efficiency losses, highlighting the need for better preference elicitation algorithms.

We show in this paper that these limitations can be partially explained due to the usage of poor, non-informative ML-algorithms, which either do not include important prior domain knowledge or make too restrictive assumptions about the bidders' value functions. Brero *et al.* [2018; 2021] used support vector regressions (SVRs) with quadratic kernels which can only learn up to two way interactions between items and do not account for an important monotonicity property of bidders' value functions. While the fully-connected feed-forward NNs used by Weissteiner and Seuken [2020] are more expressive, they also do not account for this monotonicity property. In particular when operating with only a small number of data points (which is the standard in market mechanisms, because preference elicitation is costly), this can cause significant efficiency losses.

Over the last decade, major successes in ML were made by specialized NN architectures (e.g., Convolutional Neural Networks) that incorporate domain-specific prior knowledge to improve generalization [Bronstein *et al.*, 2017]. With this paper, we follow the same paradigm by incorporating prior knowledge about monotone preferences into an NN architecture to improve generalization, which is key for a well-functioning preference elicitation algorithm.

Several other approaches for incorporating monotonicity into NNs have previously been proposed. However, for these architectures, it is not known how the NN-based winner determination problem (WDP) could be solved quickly or they have other limitations. Sill [1998] proposes only a shallow architecture which violates the normalization property. You *et al.* [2017] propose a complicated non-standard architecture, where no computationally feasible MILP formulation of the corresponding WDP is known. Wehenkel and Louppe [2019] implement monotonicity by representing the target function as an integral of an NN and thus the WDP would result in a computationally infeasible MILP. Liu *et al.* [2020] train NNs with successively higher regularization until a MILP based verification procedure guarantees monotonicity. The repeated retraining and verification lead to high computational cost.

In contrast, our approach is particularly well suited for combinatorial assignment, because (i) our NN-based WDP can be formulated as a succinct MILP and thus solved quickly in practice and (ii) we propose a generic fully-connected feed-forward architecture with arbitrary number of hidden layers which can be trained efficiently.

1.2 Our Contribution

We make the following contributions:

1. We introduce *monotone-value neural networks (MVNNs)*, a new class of NNs. For our MVNNs, we use as activation function *bounded ReLU (bReLU)* and enforce constraints on the parameters such that they are normalized and fulfill a monotonicity property (Section 3). These MVNNs are specifically suited to model monotone (combinatorial) value functions in economic settings.
2. On a technical level, we provide the following two theorems (Section 3.1): First, we prove that MVNNs are universal in the class of monotone and normalized combinatorial value functions, i.e., one can represent *any* value function with arbitrarily complex substitutabilities and complementarities exactly as an MVNN. Second, we show how to formulate the MVNN-based WDP as a MILP, which is key to calculate optimal allocations in practice.
3. We experimentally evaluate the learning performance of MVNNs vs. NNs in four different spectrum CA domains and show that MVNNs are significantly better at modelling bidders' combinatorial value functions (Section 4).
4. Finally, we experimentally investigate the performance of MVNNs vs. plain NNs when integrated into an existing ML-based ICA (MLCA) and compare them also to the FT-based method by Weissteiner *et al.* [2022b]. We show that MVNNs lead to substantially smaller efficiency losses than existing state-of-the-art mechanisms (Section 5).

2 Preliminaries

In this section, we present our formal model and review the ML-based ICA by Brero *et al.* [2021].

2.1 Formal Model for ICAs

We consider a CA with n bidders and m indivisible items. Let $N = \{1, \dots, n\}$ and $M = \{1, \dots, m\}$ denote the set of bidders and items, respectively. We denote with $x \in \mathcal{X} = \{0,1\}^m$ a bundle of items represented as an indicator vector, where $x_j = 1$ iff item $j \in M$ is contained in x. Bidders' true preferences over bundles are represented by their (private) value functions $v_i : \mathcal{X} \to \mathbb{R}_+,\ i \in N$, i.e., $v_i(x)$ represents bidder i's true value for bundle x.

By $a = (a_1, \dots, a_n) \in \mathcal{X}^n$ we denote an allocation of bundles to bidders, where a_i is the bundle bidder i obtains. We denote the set of *feasible* allocations by $\mathcal{F} = \left\{a \in \mathcal{X}^n : \sum_{i \in N} a_{ij} \leq 1,\ \forall j \in M\right\}$. We assume that bidders' have quasilinear utilities u_i, i.e., for payments $p \in \mathbb{R}^n_+$, $u_i(a,p) = v_i(a_i) - p_i$. This implies that the (true) *social welfare* $V(a)$ of an allocation a is equal to the sum of all bidders' values, i.e., $V(a) = \sum_{i \in N} u_i(a_i, p_i) + u_{\text{auctioneer}}(p) = \sum_{i \in N} v_i(a_i) - p_i + \sum_{i \in N} p_i = \sum_{i \in N} v_i(a_i)$. We let $a^* \in \operatorname{argmax}_{a \in \mathcal{F}} V(a)$ be a social-welfare maximizing, i.e., *efficient*, allocation. The *efficiency* of any allocation $a \in \mathcal{F}$ is measured as $V(a)/V(a^*)$.

An ICA *mechanism* defines how the bidders interact with the auctioneer and how the final allocation and payments are determined. We denote a bidder's (possibly untruthful) reported value function by $\hat{v}_i : \mathcal{X} \to \mathbb{R}_+$. In this paper, we consider ICAs that ask bidders to iteratively report their values $\hat{v}_i(x)$ for particular bundles x selected by the mechanism. A set of $L \in \mathbb{N}$ reported bundle-value pairs of bidder i is denoted as $R_i = \left\{\left(x^{(l)}, \hat{v}_i(x^{(l)})\right)\right\}_{l=1}^{L}$, $x^{(l)} \in \mathcal{X}$. Let $R = (R_1, \ldots, R_n)$ denote the tuple of reported bundle-value pairs obtained from all bidders. We define the *reported social welfare* of an allocation a given R as $\widehat{V}(a|R) := \sum_{i \in N:\, (a_i, \hat{v}_i(a_i)) \in R_i} \hat{v}_i(a_i)$, where $(a_i, \hat{v}_i(a_i)) \in R_i$ ensures that only values for reported bundles contribute. The final optimal allocation $a^*_R \in \mathcal{F}$ and payments $p(R) \in \mathbb{R}^n_+$ are computed based on the elicited reports R *only*. More formally, the optimal allocation $a^*_R \in \mathcal{F}$ given the reports R is defined as

$$a^*_R \in \operatorname*{argmax}_{a \in \mathcal{F}} \widehat{V}(a|R). \tag{1}$$

As the auctioneer can generally only query each bidder i a limited number of bundles $|R_i| \leq Q^{\max}$ (e.g., $Q^{\max} = 100$), the mechanism needs a sophisticated preference elicitation algorithm, with the goal of finding a highly efficient final allocation a^*_R with a limited number of value queries.

2.2 A Machine Learning-powered ICA

We now review the *machine learning-powered combinatorial auction (MLCA)* by Brero *et al.* [2021]. Interested readers are referred to Appendix A, where we present MLCA in detail.

MLCA proceeds in rounds until a maximum number of value queries per bidder $Q^{\max}$ is reached. In each round, a generic ML algorithm $\mathcal{A}_i$ is trained for every bidder $i \in N$ on the bidder's reports R_i. Next, MLCA generates new value queries $q^{\text{new}} = (q_i^{\text{new}})_{i=1}^n$ with $q_i^{\text{new}} \in \mathcal{X} \setminus R_i$ by solving a ML-based WDP $q^{\text{new}} \in \operatorname*{argmax}_{a \in \mathcal{F}} \sum_{i \in N} \mathcal{A}_i(a_i)$.

The idea is the following: if $\mathcal{A}_i$ are good surrogate models of the bidders' true value functions then q^{new} should be a good proxy of the efficient allocation a^* and thus provide the auctioneer with valuable information. Additionally, in a real-world MLCA, bidders' are always allowed to report values of further bundles that they deem potentially useful ("push"-bids). This mitigates the risk of bidders not getting asked the right queries. In our experiments, MLCA achieves already state-of-the-art results without making use of any "push"-bids (mathematically additional "push"-bids can only improve the results further).

At the end of each round, MLCA receives reports R^{new} from all bidders for the newly generated queries q^{new}, and updates the overall elicited reports R. When $Q^{\max}$ is reached, MLCA computes an allocation a^*_R that maximizes the *reported* social welfare (see Equation (1)) and determines VCG payments $p(R)$ based on the reported values (see Appendix Definition B.1).

Remark 1 (IR, No-Deficit, and Incentives of MLCA)**.** *Brero* et al. *[2021] showed that MLCA satisfies* individual rationality (IR) *and* no-deficit*, with any ML algorithm $\mathcal{A}_i$. Furthermore, they studied the incentive properties of MLCA; this is important, given that opportunities for manipulations might lower efficiency. Like all deployed spectrum auctions (including the CCA [Ausubel and Baranov, 2017]) MLCA is not strategyproof. However, Brero* et al. *[2021] argued that it has good incentives in practice; and given two additional assumptions, bidding truthfully is an ex-post Nash equilibrium in MLCA. Their analyses apply to MLCA using any ML algorithm, and therefore also to an MVNN-based MLCA. We present a more detailed summary of their incentive analysis in Appendix B.*

3 Monotone-Value Neural Networks

In combinatorial assignment problems, value functions are used to model each agent's (reported) value for a bundle of items ($\hat{v}_i : \{0,1\}^m \to \mathbb{R}_+$). However, while the bundle space grows exponentially with the number of items, the agents' value functions often exhibit useful structure that can be exploited. A common assumption is monotonicity:

(M) Monotonicity (*"additional items increase value"*):
For $A, B \in 2^M$: if $A \subseteq B$ it holds that $\hat{v}_i(A) \leq \hat{v}_i(B)$.[1]

This property is satisfied in many market domains. For example, in many CAs, bidders can freely dispose of unwanted items; in combinatorial course allocation, students can just drop courses they have been assigned. However, prior work on ML-based market design [Weissteiner and Seuken, 2020; Weissteiner *et al.*, 2022b; Brero *et al.*, 2021] has not taken this property into account, which negatively affects the performance (see Sections 4 and 5).

For ease of exposition, we additionally assume that the value functions are normalized:

(N) Normalization (*"no value for empty bundle"*):
$\hat{v}_i(\emptyset) = \hat{v}_i((0, \ldots, 0)) := 0$

Note that this property is not required by our method and can be easily adapted to be any other fixed value, or to be a learned parameter. In the following, we denote with

$$\mathcal{V} := \{\hat{v}_i : \mathcal{X} \to \mathbb{R}_+ |\, \text{satisfy } \textbf{(N)} \text{ and } \textbf{(M)}\} \tag{2}$$

the set of all value functions, that satisfy the *normalization* and *monotonicity* property. Next, we introduce *Monotone-Value Neural Networks (MVNNs)* and show that they span the entire set $\mathcal{V}$. Thus, MVNNs are specifically suited to all applications with monotone value functions.

Definition 1 (MVNN)**.** *An MVNN $\mathcal{N}_i^\theta : \mathcal{X} \to \mathbb{R}_+$ for bidder $i \in N$ is defined as*

$$\mathcal{N}_i^\theta(x) = W^{i,K_i} \varphi_{0,t}\left(\ldots \varphi_{0,t}(W^{i,1}x + b^{i,1}) \ldots\right), \tag{3}$$

- *$K_i + 1 \in \mathbb{N}$ is the number of layers ($K_i - 1$ hidden layers),*
- *$\varphi_{0,t}$ is the* bounded ReLU (bReLU)[2] *activation function with cutoff $t > 0$:*

$$\varphi_{0,t}(z) := \min(t, \max(0, z)) \tag{4}$$

[1] We slightly abused the notation here by using sets instead of their corresponding indicator vectors as arguments of $\hat{v}_i$.

[2] bReLUs have been widely used in practice, e.g., to enhance training stability in visual pattern recognition [Liew *et al.*, 2016].

- *$W^i := (W^{i,k})_{k=1}^{K_i}$ with $W^{i,k} \geq 0$ and $b^i := (b^{i,k})_{k=1}^{K_i}$ with $b^{i,k} \leq 0$ denote a tuple of non-negative weights and non-positive biases of dimensions $d^{i,k} \times d^{i,k-1}$ and $d^{i,k}$, whose parameters are stored in $\theta = (W^i, b^i)$.*[3]

For implementation details of MVNNs we refer to Appendix C.6. Unless explicitly stated, we consider from now on a cutoff of $t = 1$ for the bReLU, i.e., $\varphi(z) := \varphi_{0,1}(z) = \min(1, \max(0, z))$. Next, we discuss the choice of bReLU.

Choice of bReLU (i) The constraints on the weights and biases enforce monotonicity of MVNNs (in fact for *any* monotone activation). (ii) For universality (see Theorem 1) we need a *bounded* monotone non-constant activation, i.e., with ReLUs and our constraints one cannot express substitutabilities. (iii) for the MILP (see Theorem 2), we need a *piecewise linear* activation (e.g., with sigmoids we could not formulate a MILP). Taking all together, bReLU is the simplest bounded, monotone, non-constant, piecewise-linear activation (see Appendix Remarks C.2 and C.3 for a detailed discussion).

Remark 2. *For applications where value functions are expected to be "almost" (but not completely) monotone, one can adapt MVNNs to only have* soft monotonicity constraints *by implementing the constraints on the weights and biases via regularization, e.g., $\sum_{i,k,j,l} \max(0, -W^{i,k}_{j,l})$. This results in soft-MVNNs that can model non-monotone changes in some items if the data evidence is strong.*

3.1 Theoretical Analysis and MILP-Formulation

Next, we provide a theoretical analysis of MVNNs and present a MILP formulation of the MVNN-based WDP.

Lemma 1. *Let $\mathcal{N}_i^\theta : \mathcal{X} \to \mathbb{R}_+$ be an MVNN (Definition 1). Then it holds that $\mathcal{N}_i^{(W^i,b^i)} \in \mathcal{V}$ for all $W^i \geq 0$ and $b^i \leq 0$.*

We provide the proof for Lemma 1 in Appendix C.1. Next, we state our main theorem about MVNNs.

Theorem 1 (Universality)**.** Any *value function $\hat{v}_i : \mathcal{X} \to \mathbb{R}_+$ that satisfies **(N)** and **(M)** can be represented exactly as an MVNN $\mathcal{N}_i^\theta$ from Definition 1, i.e.,*

$$\mathcal{V} = \left\{ \mathcal{N}_i^{(W^i,b^i)} : W^i \geq 0, b^i \leq 0 \right\}. \tag{5}$$

We present a constructive proof for Theorem 1 in Appendix C.3. In the proof, we consider an arbitrary $(\hat{v}_i(x))_{x \in \mathcal{X}} \in \mathcal{V}$ for which we construct a two hidden layer MVNN $\mathcal{N}_i^\theta$ of dimensions $[m, 2^m - 1, 2^m - 1, 1]$ with parameters $\theta = (W^i_{\hat{v}_i}, b^i_{\hat{v}_i})$ such that $\mathcal{N}_i^\theta(x) = \hat{v}_i(x) \, \forall x \in \mathcal{X}$.

Note that for q queries, it is always possible to construct MVNN of size $[m, q, q, 1]$ that perfectly fits through the q data-points (see Appendix Corollary C.1). Thus, the worst-case required architecture size grows only linearly with the number of queries. In our experiments we already achieve very good performance with even smaller architectures. Example C.1 in the Appendix nicely illustrates how exactly MVNNs capture complementarities, substitutabilities and independent items.

A key step in combinatorial assignment mechanisms is finding the social welfare-maximizing allocation, i.e., solving the *Winner Determination Problem* (WDP). To use MVNNs in such mechanisms, we need to be able to solve MVNN-based WDPs in a practically feasible amount of time. To this end, we present a MILP formulation of the MVNN-based WDP which can be (approximately) solved in practice for reasonably-sized NNs (see Section 5.3). The key idea is to rewrite the bReLU $\varphi(z)$ as $-\max(-1, -\max(0, z))$ and encode for each bidder i, hidden layer k and neuron j both $\max(\cdot, \cdot)$ operators with two binary decision variables $y^{i,k}_j, \mu^{i,k}_j$. First, we show how to encode one *single* hidden layer of an MVNN as multiple linear constraints. We provide the proof in Appendix C.4.

Lemma 2. *Fix bidder $i \in N$, let $k \in \{1, \ldots, K_i - 1\}$ and denote the pre-activated output of the kth layer as $o^{i,k} := W^{i,k} z^{i,k-1} + b^{i,k}$ with $W^{i,k} \in \mathbb{R}^{d^{i,k} \times d^{i,k-1}}, b^{i,k} \in \mathbb{R}^{d^{i,k}}$. Then the output of the kth layer $z^{i,k} := \varphi(o^{i,k}) = \min(1, \max(0, o^{i,k})) = -\max(-1, -\eta^{i,k})$, with $\eta^{i,k} := \max(0, o^{i,k})$ can be equivalently expressed by the following linear constraints:*

$$o^{i,k} \leq \eta^{i,k} \leq o^{i,k} + y^{i,k} \cdot L_1^{i,k} \tag{6}$$

$$0 \leq \eta^{i,k} \leq (1 - y^{i,k}) \cdot L_2^{i,k} \tag{7}$$

$$\eta^{i,k} - \mu^{i,k} \cdot L_3^{i,k} \leq z^{i,k} \leq \eta^{i,k} \tag{8}$$

$$1 - (1 - \mu^{i,k}) \cdot L_4^{i,k} \leq z^{i,k} \leq 1 \tag{9}$$

$$y^{i,k} \in \{0,1\}^{d^{i,k}}, \quad \mu^{i,k} \in \{0,1\}^{d^{i,k}}, \tag{10}$$

where $L_1^{i,k}, L_2^{i,k}, L_3^{i,k}, L_4^{i,k} \in \mathbb{R}_+$ are large enough constants for the respective big-M *constraints.*[4]

Finally, we formulate the MVNN-based WDP as a MILP.

Theorem 2 (MILP)**.** *The MVNN-based WDP $\max_{a \in \mathcal{F}} \sum_{i \in N} \mathcal{N}_i^{(W^i,b^i)}(a_i)$ can be equivalently formulated as the following MILP*[5]*:*

$$\max_{a \in \mathcal{F}, z^{i,k}, \mu^{i,k}, \eta^{i,k}, y^{i,k}} \left\{ \sum_{i \in N} W^{i,K_i} z^{i,K_i - 1} \right\} \tag{11}$$

s.t. for $i \in N$ and $k \in \{1, \ldots, K_i - 1\}$

$$z^{i,0} = a_i \tag{12}$$

$$W^{i,k} z^{i,k-1} + b^{i,k} \leq \eta^{i,k} \tag{13}$$

$$\eta^{i,k} \leq W^{i,k} z^{i,k-1} + b^{i,k} + y^{i,k} \cdot L_1^{i,k} \tag{14}$$

$$0 \leq \eta^{i,k} \leq (1 - y^{i,k}) \cdot L_2^{i,k} \tag{15}$$

$$\eta^{i,k} - \mu^{i,k} \cdot L_3^{i,k} \leq z^{i,k} \leq \eta^{i,k} \tag{16}$$

$$1 - (1 - \mu^{i,k}) \cdot L_4^{i,k} \leq z^{i,k} \leq 1 \tag{17}$$

$$y^{i,k} \in \{0,1\}^{d^{i,k}}, \qquad \mu^{i,k} \in \{0,1\}^{d^{i,k}} \tag{18}$$

[3] We apply a linear readout map to the last hidden layer, i.e, no $\varphi_{0,t}$ and $b^{i,K_i} := 0$. By setting $b^{i,K_i} \neq 0$ with *trainable=False*, the MVNN can model any other value than zero in the normalization property. By not restricting $b^{i,k} \leq 0$ and setting $b^{i,K_i} \neq 0$ with *trainable=True* one can also learn the value for the empty bundle.

[4] To account for a general cutoff $t \neq 1$ in the bReLU, one needs to adjust (9) by replacing the left- and rightmost 1 with t.

[5] To account for a general cutoff $t \neq 1$ in the bReLU, one needs to adjust (17) by replacing the left- and rightmost 1 with t.

Proof. The proof follows by iteratively applying Lemma 2 for each bidder and all her respective hidden MVNN layers. □

Fact 1. *One can significantly reduce the solve time for the MILP by tightening the bounds of each neuron. In Appendix C.5, we present bound tightening via* interval arithmetic (IA) *[Tjeng* et al.*, 2019] for MVNNs. For a plain ReLU NN, these IA bounds are not tight and calculating tighter bounds in a computationally efficient manner is very challenging. In contrast, the MVNN-IA bounds are always* perfectly tight*, because of their encoded monotonicity property. The upper and lower bound of a neuron is the value the neuron outputs for the input* $(1, \ldots, 1)$ *and* $(0, \ldots, 0)$*. This is a big advantage of MVNNs compared to plain (ReLU) NNs.*

Remark 3 (MVNNs as Preference Generator)**.** *To experimentally evaluate a new mechanism one needs to generate many different possible value functions. Preference generators like the Spectrum Auction Test Suite [Weiss* et al.*, 2017] are carefully designed to capture the essential properties of spectrum auctions, but they are not available for every application. Instead of using such a domain-specific preference generator, one can also use MVNNs with randomly initialized weights to generate possible value functions. An advantage of random MVNNs is that they are universal (see Theorem 1) and hence come with a diversity rich enough to sample any possible monotone value function with arbitrarily complex substitutabilities and complementarities (the distribution of supplements and complements can be controlled via the cutoff* t*, where the smaller/larger* t *the more substitutabilities/complementarities). Future work could investigate this distribution, i.e., how representative it is for real-world valuations. These types of generative test beds become increasingly important to avoid overfitting on specific simulation engines and/or real data sets [Osband* et al.*, 2021].*

4 Prediction Performance of MVNNs

In this section, we show that in all considered CA domains, MVNNs are significantly better at capturing bidders' complex value functions than plain (ReLU) NNs, which allows them to extrapolate much better in the bundle space.

4.1 Experimental Setup - Prediction Performance

CA Domains In our experiments we use simulated data from the Spectrum Auction Test Suite (SATS) version 0.7.0 [Weiss *et al.*, 2017]. We consider the following four domains:

- **Global Synergy Value Model (GSVM)** [Goeree and Holt, 2010] has 18 items, 6 *regional* and 1 *national bidder*.
- **Local Synergy Value Model (LSVM)** [Scheffel *et al.*, 2012] has 18 items, 5 *regional* and 1 *national bidder*. Complementarities arise from spatial proximity of items.
- **Single-Region Value Model (SRVM)** [Weiss *et al.*, 2017] has 29 items and 7 bidders (categorized as *local*, *high frequency regional*, or *national*) and models large UK 4G spectrum auctions.
- **Multi-Region Value Model (MRVM)** [Weiss *et al.*, 2017] has 98 items and 10 bidders (*local*, *regional*, or *national*) and models large Canadian 4G spectrum auctions.

When simulating bidders, we follow prior work (e.g., [Brero *et al.*, 2021]) and assume truthful bidding (i.e., $\hat{v}_i = v_i$). Details on how we collect the data and the train/val/test split can be found in Appendix D.1.

HPO To efficiently optimize the hyperparameters and fairly compare MVNNs and plain NNs for best generalization across different instances of each SATS domain, we frame the *hyperparameter optimization (HPO)* problem as an algorithm configuration problem and use the well-established *sequential model-based algorithm configuration (SMAC)* [Hutter *et al.*, 2011]. SMAC quickly discards hyperparameters which already perform poorly on a few SATS instances and proposes more promising ones via Bayesian optimization. It is flexible enough for the parameterization of NNs as it naturally handles a mixture of categorical, integer and float hyperparameters. Further details on the setting including hyperparameter ranges can be found in Appendix D.2.

4.2 Prediction Performance Results

For ease of exposition, we only present our results for the MVNN-RELU-PROJECTED implementation of our MVNNs (termed MVNN in the following). Results for other MVNN implementations can be found in the Appendix D.3. In Table 1, we compare the prediction performance of the winning models that the HPO found for different amounts of training data (T) on the test data. We see that, compared to plain

			R^2 ↑		KT ↑	
DOMAIN	T	BIDDER	MVNN	NN	MVNN	NN
GSVM	10	NAT	0.686 ±0.061	0.534 ±0.040	0.668 ±0.027	0.583 ±0.021
		REG	0.618 ±0.068	0.504 ±0.062	0.633 ±0.038	0.557 ±0.033
	20	NAT	0.923 ±0.016	0.818 ±0.032	0.849 ±0.017	0.752 ±0.029
		REG	0.940 ±0.018	0.880 ±0.022	0.882 ±0.020	0.815 ±0.021
	50	NAT	0.992 ±0.001	0.988 ±0.001	0.962 ±0.003	0.953 ±0.003
		REG	0.997 ±0.001	0.988 ±0.001	0.974 ±0.002	0.953 ±0.003
LSVM	10	NAT	0.248 ±0.069	0.137 ±0.031	0.693 ±0.011	0.710 ±0.023
		REG	0.563 ±0.049	0.348 ±0.067	0.605 ±0.031	0.504 ±0.025
	50	NAT	0.616 ±0.020	0.199 ±0.031	0.753 ±0.009	0.678 ±0.035
		REG	0.921 ±0.015	0.872 ±0.012	0.860 ±0.017	0.812 ±0.013
	100	NAT	0.677 ±0.014	0.396 ±0.033	0.813 ±0.005	0.706 ±0.018
		REG	0.965 ±0.010	0.936 ±0.010	0.918 ±0.015	0.857 ±0.012
SRVM	10	H.F.	0.538 ±0.044	-2.123 ±0.268	0.626 ±0.020	0.607 ±0.012
		LO	0.381 ±0.045	0.267 ±0.042	0.559 ±0.030	0.489 ±0.032
		NAT	0.389 ±0.063	0.341 ±0.038	0.560 ±0.026	0.535 ±0.012
		REG	0.422 ±0.051	0.372 ±0.036	0.562 ±0.023	0.544 ±0.014
	50	H.F.	0.860 ±0.015	0.773 ±0.034	0.853 ±0.013	0.803 ±0.020
		LO	0.895 ±0.020	0.588 ±0.031	0.902 ±0.000	0.771 ±0.030
		NAT	0.988 ±0.004	0.828 ±0.015	0.918 ±0.005	0.801 ±0.009
		REG	0.989 ±0.004	0.872 ±0.047	0.931 ±0.004	0.823 ±0.022
	100	H.F.	0.911 ±0.008	0.849 ±0.011	0.908 ±0.006	0.896 ±0.006
		LO	0.948 ±0.014	0.723 ±0.005	0.903 ±0.000	0.900 ±0.002
		NAT	0.998 ±0.000	0.913 ±0.008	0.952 ±0.003	0.841 ±0.008
		REG	0.996 ±0.001	0.945 ±0.004	0.948 ±0.021	0.895 ±0.012
MRVM	10	LO	-0.055 ±0.058	-0.018 ±0.050	0.262 ±0.017	0.200 ±0.015
		NAT	0.182 ±0.045	-0.556 ±1.355	0.365 ±0.018	0.414 ±0.008
		REG	0.036 ±0.085	-0.048 ±0.092	0.322 ±0.022	0.255 ±0.038
	100	LOCAL	0.831 ±0.023	0.493 ±0.027	0.786 ±0.019	0.545 ±0.012
		NAT	0.778 ±0.022	0.560 ±0.019	0.726 ±0.014	0.581 ±0.010
		REG	0.832 ±0.028	0.447 ±0.053	0.779 ±0.027	0.572 ±0.018
	300	LOCAL	0.944 ±0.006	0.871 ±0.009	0.883 ±0.010	0.819 ±0.009
		NAT	0.868 ±0.016	0.855 ±0.029	0.814 ±0.009	0.808 ±0.013
		REG	0.917 ±0.017	0.847 ±0.010	0.851 ±0.019	0.809 ±0.012

Table 1: Prediction performance measured via R-squared (R^2) and Kendall tau (KT) with a 95%-CI in four SATS domains with corresponding bidder types: high frequency (H.F.), local (LO), regional (REG) and national (NAT), averaged over 30 auction instances. Both MVNNs and plain NNs are trained on T and evaluated on $209\,715 - T$ random bundles. Winners are marked in grey.

Domain	Q^{MAX}	Efficiency Loss in % ↓				T-Test for Efficiency:	
		MVNN	NN	FT	RS	$\mathcal{H}_0 : \mu_{\text{NN}} \leq \mu_{\text{MVNN}}$	$\mathcal{H}_0 : \mu_{\text{FT}} \leq \mu_{\text{MVNN}}$
GSVM	100	00.00 ± 0.00	00.00 ± 0.00	01.77± 0.96	30.34 ± 1.61		$p_{\text{VAL}} = 3\text{e}{-6}$
LSVM	100	00.70 ± 0.40	02.91 ± 1.44	01.54± 0.65	31.73 ± 2.15	$p_{\text{VAL}} = 2\text{e}{-03}$	$p_{\text{VAL}} = 5\text{e}{-3}$
SRVM	100	00.23 ± 0.06	01.13 ± 0.22	00.72± 0.16	28.56 ± 1.74	$p_{\text{VAL}} = 5\text{e}{-10}$	$p_{\text{VAL}} = 2\text{e}{-8}$
MRVM	100	08.16 ± 0.41	09.05 ± 0.53	10.37± 0.57	48.79 ± 1.13	$p_{\text{VAL}} = 9\text{e}{-03}$	$p_{\text{VAL}} = 1\text{e}{-7}$

Table 2: Efficiency loss of MVNN vs plain NNs, the Fourier Transform (FT) benchmark and random search (RS). Shown are averages and a 95% CI on a test set of 50 CA instances. Winners based on a (paired) t-test with significance level of 1% are marked in grey.

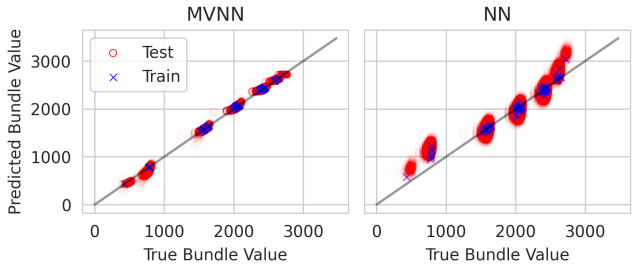


Figure 1: Prediction performance of MVNNs vs plain NNs in SRVM (national bidder). The identity is shown in grey.

NNs, MVNNs provide both a significantly better fit in terms of R-squared R^2 as well as a better Kendall Tau rank correlation **KT** (i.e., a better ordinal ranking of the predicted test bundle values). Thus, enforcing the monotonicity property in MVNNs significantly improves the learning performance.

Figure 1 illustrates our findings by providing a visual comparison of the prediction performance for the highly non-unimodal SRVM.[6] We see that the MVNN fits the training data exactly (blue crosses), although the HPO only optimized generalization performance on the validation data. This is a strong indication that MVNNs correspond to a more realistic prior, since for a realistic prior, it is optimal to exactly fit the training data in noiseless settings [Heiss *et al.*, 2022, Proposition D.2.a]. In contrast, the HPO has selected hyperparameters for the plain NNs that result in a worse fit of the training data (otherwise generalization to unseen data would be even worse). Moreover, the plain NNs show a particularly bad fit on the less frequent lower and higher valued bundles.

5 MVNN-powered Iterative CA

To evaluate the performance of MVNNs when used inside a combinatorial market mechanism, we have integrated MVNNs into MLCA (see Section 2.2), yielding an *MVNN-powered Iterative CA*. In this section, we compare the economic efficiency of our MVNN-based MLCA against the previously proposed NN-based MLCA. For solving the MVNN-based WDPs in MLCA, we use our MILP from Theorem 2.

5.1 Experimental Setup - MLCA

To generate synthetic CA instances, we use the same four SATS domains as in Section 4. SATS also gives us access to the true optimal allocation a^*, which we use to measure the *efficiency loss*, i.e., $1 - V(a^*_R)/V(a^*)$ and *relative revenue* $\sum_{i \in N} p(R)_i / V(a^*)$ of an allocation $a^*_R \in \mathcal{F}$ and payments $p(R) \in \mathbb{R}^n_+$ determined by MLCA when eliciting reports R. Due to the long run-time of a single evaluation of MLCA, we perform a restricted HPO, which, for each domain, only uses the winners of the prediction performance experiment (Table 1) for the three amounts of training data T. This is a reasonable choice, because in the prediction performance we optimize the generalization performance for bidders' value functions that is also key in MLCA.

For each domain, we use $Q^{\text{init}} = 40$ initial random queries and set the query budget to $Q^{\text{max}} = 100$. We terminate MLCA in an intermediate iteration if it already found an efficient allocation (i.e., with 0 efficiency loss).

5.2 Efficiency Results

In Table 2, we present the efficiency results of MVNN-based MLCA and NN-based MLCA, averaged over 50 auction instances. We focus on efficiency rather than revenue, since spectrum auctions are government-run auctions with a mandate to maximize efficiency and not revenue [Cramton, 2013]. In Appendices E.2 and E.3, we also present and discuss detailed revenue results.

For each domain, we present results corresponding to the best MVNNs and NNs amongst the three incumbents obtained from the prediction performance experiments. We present results for all three incumbents in Appendix E.2. Overall, we see that the better prediction performance of MVNNs (Table 1) translates to smaller efficiency losses in MLCA. In LSVM and SRVM, MVNNs significantly outperform NNs and have a more than four times lower efficiency loss. In MRVM, MVNN's average efficiency loss is approximately 1% point smaller than the NN's loss. Given that in the 2014 Canadian 4G auction the total revenue was on the order of 5 billion USD [Ausubel and Baranov, 2017], an efficiency loss decrease of 1% point in MRVM can translate to welfare gains on the order of 50 million USD. Finally, in GSVM, the simplest domain, where bidders' value functions have at most two-way interactions between items, both MVNNs and plain NNs incur no efficiency loss. As further baselines, we evaluate the Fourier transform (FT) auction [Weissteiner *et al.*, 2022b] using their proposed optimal hyperparameters and random search (RS). We do not compare to SVRs [Brero *et al.*, 2021] since they were already outperformed by plain NNs in [Weissteiner and Seuken, 2020]. We observe that RS incurs efficiency losses of 30–50% illustrating the need for smart preference elicitation. Moreover, we see that MVNNs also significantly outperform FTs in all domains.

[6] We provide corresponding plots for the other domains and bidder types in Appendix D.3; the results are qualitatively similar.

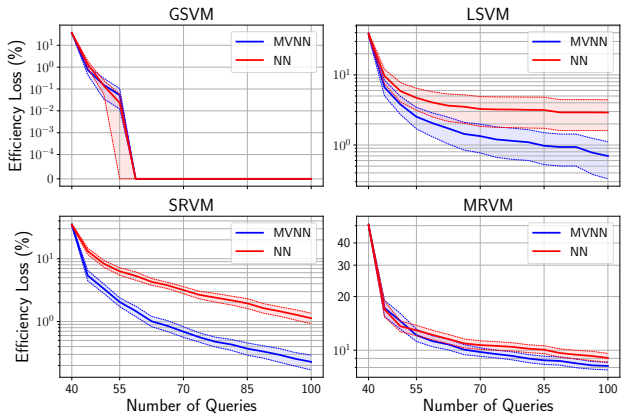


Figure 2: Efficiency loss paths of MLCA with MVNNs vs plain NNs. Shown are averages with 95% CIs over 50 auction instances.

In Figure 2, we present the efficiency loss path of MVNNs vs NNs (i.e., the regret curve) corresponding to Table 2. We see that in LSVM and SRVM, MVNNs lead to a smaller (average) efficiency loss for *every* number of queries. In MRVM, the same holds true for 50 and more queries. In GSVM, both networks have no efficiency loss in every instance after only 56 queries. Since a single query can be very costly in real-world CAs, it makes sense to ask few queries. Figure 2 shows that MVNNs consistently outperform plain NNs also in settings with a small number of queries (i.e., reduced $Q^{\max}$)

5.3 MILP Runtime Analysis

When integrating MVNNs into MLCA or another iterative combinatorial assignment mechanism, one needs to solve the MVNN-based WDP multiple times in one full run. Thus, the key computational challenge when integrating MVNNs in such mechanisms is to make solving the MVNN-based WDP practically feasible. In Theorem 2, we have shown how to encode the MVNN-based WDP as a succinct MILP, which can be (approximately) solved in practice for reasonably-sized NNs. However, due to the bReLU, i.e., the two cutoffs at 0 and $t > 0$, the MVNN-based MILP has twice the number of binary variables ($y^{i,k}$ and $\mu^{i,k}$) than the MILP encoding of a plain NN with ReLUs [Weissteiner and Seuken, 2020].

Figure 3 presents MILP runtimes of MVNNs vs plain ReLU NNs for selected architectures. We observe two effects: First, even though the MVNN-based MILPs have twice the number of binary variables, they can be solved faster than the plain NN-based MILPs. Second, the deeper the architecture or the more neurons, the larger this difference becomes. One hypothesis to explain these effects is that (i) MVNNs are more regular functions than plain NNs (due to their monotonicity property) and (ii) the constraints on their parameters yield structure that might be exploited by the MILP solver.

6 Conclusion

In this paper, we have introduced MVNNs, a new class of NNs that is specifically designed to model normalized and monotone value functions in combinatorial assignment problems. We have experimentally evaluated the performance of

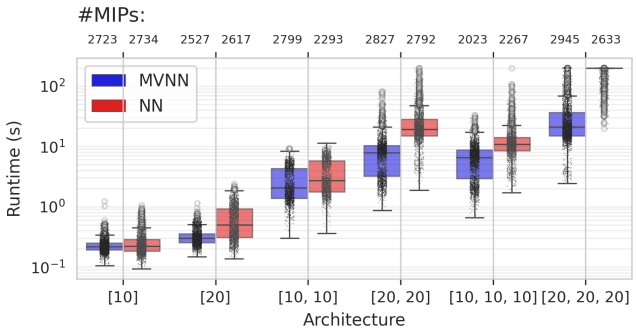


Figure 3: MILP runtime (200s time limit) of MVNNs vs plain NNs in MLCA on 10 LSVM instances for a selection of architectures.

MVNNs in four combinatorial spectrum auction domains and shown that MVNNs outperform plain NNs with respect to prediction performance, economic efficiency, and runtime. Overall, our experiments suggest that MVNNs are the best currently available model for preference elicitation in combinatorial assignment (also compared to FTs and SVRs). Thus, incorporating important structural knowledge in the ML algorithm plays an important role in combinatorial assignment.

MVNNs enable us to incorporate an informative *prior* into a market mechanism. Future work could use such informative priors and enhance existing mechanisms (e.g., MLCA) by also using the *posterior* estimates in a more principled way than just the mean prediction. For example, one could frame an ICA as a (combinatorial) Bayesian optimization task and integrate a well-defined notion of posterior uncertainty to foster exploration [Heiss *et al.*, 2022].[7] Finally, it would be interesting to also evaluate the performance of MVNNs in other combinatorial assignment problems such as course allocation.[8]

Acknowledgments

We thank the anonymous reviewers for helpful comments. This paper is part of a project that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (Grant agreement No. 805542).

References

[Ausubel and Baranov, 2017] Lawrence M Ausubel and Oleg Baranov. A practical guide to the combinatorial clock auction. *Economic Journal*, 127(605):F334–F350, 2017. 1, 3, 6

[Ausubel and Cramton, 2011] Lawrence Ausubel and Peter Cramton. Auction design for wind rights. *Report to Bureau of Ocean Energy Management, Regulation and Enforcement*, 2011. 1

[7] After the publication of the present paper, Weissteiner *et al.* [2023] introduced uncertainty-based exploration for ML-powered ICAs using the method by Heiss *et al.* [2022].

[8] After the publication of the present paper, Soumalias *et al.* [2023] used MVNNs to learn students' preferences in the course allocation domain.

[Ausubel *et al.*, 2006] Lawrence M Ausubel, Peter Cramton, and Paul Milgrom. The clock-proxy auction: A practical combinatorial auction design. In Peter Cramton, Yoav Shoham, and Richard Steinberg, editors, *Combinatorial Auctions*, pages 115–138. MIT Press, 2006. 1

[Bichler *et al.*, 2019] Martin Bichler, Vladimir Fux, and Jacob K Goeree. Designing combinatorial exchanges for the reallocation of resource rights. *Proceedings of the National Academy of Sciences*, 116(3):786–791, 2019. 1

[Brero *et al.*, 2018] Gianluca Brero, Benjamin Lubin, and Sven Seuken. Combinatorial auctions via machine learning-based preference elicitation. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 2018. 2

[Brero *et al.*, 2019] Gianluca Brero, Sébastien Lahaie, and Sven Seuken. Fast iterative combinatorial auctions via bayesian learning. In *Proceedings of the 33rd AAAI Conference of Artificial Intelligence*, 2019. 2

[Brero *et al.*, 2021] Gianluca Brero, Benjamin Lubin, and Sven Seuken. Machine learning-powered iterative combinatorial auctions. *arXiv preprint arXiv:1911.08042*, Jan 2021. 2, 3, 5, 6, 10, 15

[Bronstein *et al.*, 2017] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017. 2

[Budish, 2011] Eric Budish. The combinatorial assignment problem: Approximate competitive equilibrium from equal incomes. *Journal of Political Economy*, 119(6):1061–1103, 2011. 1

[Cramton, 2013] Peter Cramton. Spectrum auction design. *Review of Industrial Organization*, 42(2):161–190, 2013. 1, 6

[Dütting *et al.*, 2019] Paul Dütting, Zhe Feng, Harikrishna Narasimhan, David C Parkes, and Sai Srivatsa Ravindranath. Optimal auctions through deep learning. In *Proceedings of the 36th International Conference on Machine Learning*, 2019. 2

[Goeree and Holt, 2010] Jacob K Goeree and Charles A Holt. Hierarchical package bidding: A paper & pencil combinatorial auction. *Games and Economic Behavior*, 70(1):146–169, 2010. 5

[Heiss *et al.*, 2022] Jakob M Heiss, Jakob Weissteiner, Hanna S Wutte, Sven Seuken, and Josef Teichmann. NOMU: Neural optimization-based model uncertainty. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 8708–8758. PMLR, 17–23 Jul 2022. 6, 7

[Hutter *et al.*, 2011] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *International Conference on learning and intelligent optimization*, pages 507–523. Springer, 2011. 5

[Liew *et al.*, 2016] Shan Sung Liew, Mohamed Khalil-Hani, and Rabia Bakhteri. Bounded activation functions for enhanced training stability of deep neural networks on visual pattern recognition problems. *Neurocomputing*, 216:718–734, 2016. 3

[Liu *et al.*, 2020] Xingchao Liu, Xing Han, Na Zhang, and Qiang Liu. Certified monotonic neural networks. *Advances in Neural Information Processing Systems*, 33:15427–15438, 2020. 2

[Loshchilov and Hutter, 2017] I. Loshchilov and F. Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017. 15

[Nisan and Segal, 2006] Noam Nisan and Ilya Segal. The communication requirements of efficient allocations and supporting prices. *Journal of Economic Theory*, 129(1):192–224, 2006. 1

[Osband *et al.*, 2021] Ian Osband, Zheng Wen, Mohammad Asghari, Morteza Ibrahimi, Xiyuan Lu, and Benjamin Van Roy. Epistemic neural networks. *arXiv preprint arXiv:2107.08924*, 2021. 5

[Rahme *et al.*, 2021] Jad Rahme, Samy Jelassi, Joan Bruno, and S. Matthew Weinberg. A permutation-equivariant neural network architecture for auction design. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*, 2021. 2

[Scheffel *et al.*, 2012] Tobias Scheffel, Georg Ziegler, and Martin Bichler. On the impact of package selection in combinatorial auctions: an experimental study in the context of spectrum auction design. *Experimental Economics*, 15(4):667–692, 2012. 5

[Sill, 1998] Joseph Sill. Monotonic networks. *Advances in Neural Information Processing Systems*, 10:661–667, 1998. 2

[Soumalias *et al.*, 2023] Ermis Soumalias, Behnoosh Zamanlooy, Jakob Weissteiner, and Sven Seuken. Machine learning-powered course allocation. *arXiv preprint arXiv:2210.00954*, 2023. 7

[Tjeng *et al.*, 2019] Vincent Tjeng, Kai Y. Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. In *International Conference on Learning Representations*, 2019. 5

[Wehenkel and Louppe, 2019] Antoine Wehenkel and Gilles Louppe. Unconstrained monotonic neural networks. *Advances in Neural Information Processing Systems*, 32:1545–1555, 2019. 2

[Weiss *et al.*, 2017] Michael Weiss, Benjamin Lubin, and Sven Seuken. Sats: A universal spectrum auction test suite. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pages 51–59, 2017. 5

[Weissteiner and Seuken, 2020] Jakob Weissteiner and Sven Seuken. Deep learning—powered iterative combinatorial auctions. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(02):2284–2293, Apr. 2020. 2, 3, 6, 7

[Weissteiner *et al.*, 2022a] Jakob Weissteiner, Jakob Heiss, Julien Siems, and Sven Seuken. Monotone-value neural networks: Exploiting preference monotonicity in combinatorial assignment. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 541–548. International Joint Conferences on Artificial Intelligence Organization, 7 2022. Main Track. 1

[Weissteiner *et al.*, 2022b] Jakob Weissteiner, Chris Wendler, Sven Seuken, Ben Lubin, and Markus Püschel. Fourier analysis-based iterative combinatorial auctions. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 549–556. International Joint Conferences on Artificial Intelligence Organization, 7 2022. Main Track. 2, 3, 6

[Weissteiner *et al.*, 2023] Jakob Weissteiner, Jakob Heiss, Julien Siems, and Sven Seuken. Bayesian optimization-based combinatorial assignment. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37, 2023. 7

[You *et al.*, 2017] Seungil You, David Ding, Kevin Canini, Jan Pfeifer, and Maya Gupta. Deep lattice networks and partial monotonic functions. *Advances in neural information processing systems*, 30, 2017. 2

Appendix

A A Machine Learning-powered ICA

In this section, we present in detail the *machine learning-powered combinatorial auction (MLCA)* by Brero *et al.* [2021].

At the core of MLCA is a *query module* (Algorithm 1), which, for each bidder $i \in I \subseteq N$, determines a new value query q_i. First, in the *estimation step* (Line 1), an ML algorithm $\mathcal{A}_i$ is used to learn bidder i's valuation from reports R_i. Next, in the *optimization step* (Line 2), an *ML-based WDP* is solved to find a candidate q of value queries. In principle, any ML algorithm $\mathcal{A}_i$ that allows for solving the corresponding ML-based WDP in a fast way could be used. Finally, if q_i has already been queried before (Line 4), another, more restricted NN-based WDP (Line 6) is solved and q_i is updated correspondingly. This ensures that all final queries q are new.

Algorithm 1: NEXTQUERIES(I, R) (Brero et al. 2021)

Inputs: Index set of bidders I and reported values R
1 **foreach** $i \in I$ **do** Fit $\mathcal{A}_i$ on R_i: $\mathcal{A}_i[R_i]$ ▷ `Estimation step`
2 Solve $q \in \operatorname{argmax}_{a \in \mathcal{F}} \sum_{i \in I} \mathcal{A}_i[R_i](a_i)$ ▷ `Optimization step`
3 **foreach** $i \in I$ **do**
4 | **if** $(q_i, \hat{v}_i(q_i)) \in R_i$ **then** ▷ `Bundle already queried`
5 | | Define $\mathcal{F}' = \{a \in \mathcal{F} : a_i \neq x, \forall (x, \hat{v}_i(x)) \in R_i\}$
6 | | Re-solve $q' \in \operatorname{argmax}_{a \in \mathcal{F}'} \sum_{l \in I} \mathcal{A}_l[R_l](a_l)$
7 | | Update $q_i = q'_i$
8 | **end**
9 **end**
10 **return** *Profile of new queries* $q = (q_1, \ldots, q_n)$

In Algorithm 2, we present MLCA. In the following, let $R_{-i} = (R_1, \ldots, R_{i-1}, R_{i+1}, \ldots, R_n)$. MLCA proceeds in rounds until a maximum number of queries per bidder Q^{max} is reached. In each round, it calls Algorithm 1 $(Q^{\text{round}} - 1)n + 1$ times: for each bidder $i \in N$, $Q^{\text{round}} - 1$ times excluding a different bidder $j \neq i$ (Lines 5–10, sampled *marginal economies*) and once including all bidders (Line 11, *main economy*). In total each bidder is queried Q^{round} bundles per round in MLCA. At the end of each round, the mechanism receives reports R^{new} from all bidders for the newly generated queries q^{new}, and updates the overall elicited reports R (Lines 12–14). In Lines 16–17, MLCA computes an allocation a^*_R that maximizes the *reported* social welfare (see Equation (1)) and determines VCG payments $p(R)$ based on the reported values R (see Appendix Definition B.1).

B Incentives of MLCA

In this section, we briefly review the key arguments by Brero *et al.* [2021] why MLCA has good incentives in practice. First, we define VCG-payments given bidder's reports.

Definition B.1. (VCG PAYMENTS FROM REPORTS) *Let $R = (R_1, \ldots, R_n)$ denote an elicited set of reported bundle-value pairs from each bidder obtained from* MLCA *(Algorithm 2) and let*

Algorithm 2: MLCA$(Q^{\text{init}}, Q^{\text{max}}, Q^{\text{round}})$ (Brero et al. 2021)

Params: $Q^{\text{init}}, Q^{\text{max}}, Q^{\text{round}}$ initial, max and #queries/round
1 **foreach** $i \in N$ **do**
2 | Receive reports R_i for Q^{init} randomly drawn bundles
3 **end**
4 **for** $k = 1, \ldots, \lfloor (Q^{max} - Q^{init})/Q^{round} \rfloor$ **do** ▷ `Round iterator`
5 | **foreach** $i \in N$ **do** ▷ `Marginal economy queries`
6 | | Draw uniformly without replacement $(Q^{\text{round}}-1)$ bidders from $N \setminus \{i\}$ and store them in $\tilde{N}$
7 | | **foreach** $j \in \tilde{N}$ **do**
8 | | | $q^{\text{new}} = q^{\text{new}} \cup \mathit{NextQueries}(N \setminus \{j\}, R_{-j})$
9 | | **end**
10 | **end**
11 | $q^{\text{new}} = \mathit{NextQueries}(N, R)$ ▷ `Main economy queries`
12 | **foreach** $i \in N$ **do**
13 | | Receive reports R_i^{new} for q_i^{new}, set $R_i = R_i \cup R_i^{\text{new}}$
14 | **end**
15 **end**
16 Given elicited reports R compute a^*_R as in Equation (1)
17 Given elicited reports R compute VCG-payments $p(R)$
18 **return** *Final allocation* a^*_R *and payments* $p(R)$

$R_{-i} := (R_1, \ldots, R_{i-1}, R_{i+1}, \ldots, R_n)$. We then calculate the VCG payments $p(R) = (p(R)_1 \ldots, p(R)_n) \in \mathbb{R}^n_+$ as follows:

$$p(R)_i := \sum_{j \in N \setminus \{i\}} \hat{v}_j \left(\left(a^*_{R_{-i}} \right)_j \right) - \sum_{j \in N \setminus \{i\}} \hat{v}_j \left((a^*_R)_j \right). \tag{19}$$

where $a^*_{R_{-i}}$ maximizes the reported social welfare when excluding bidder i, i.e.,

$$a^*_{R_{-i}} \in \operatorname*{argmax}_{a \in \mathcal{F}} \widehat{V}(a|R_{-i}) = \operatorname*{argmax}_{a \in \mathcal{F}} \sum_{\substack{j \in N \setminus \{i\}: \\ (a_j, \hat{v}_j(a_j)) \in R_j}} \hat{v}_j(a_j), \tag{20}$$

and a^*_R is a reported-social-welfare-maximizing allocation (including all bidders), i.e,

$$a^*_R \in \operatorname*{argmax}_{a \in \mathcal{F}} \widehat{V}(a|R) = \operatorname*{argmax}_{a \in \mathcal{F}} \sum_{i \in N: (a_i, \hat{v}_i(a_i)) \in R_i} \hat{v}_i(a_i). \tag{21}$$

Therefore, when using VCG, bidder i's utility is:

$$\begin{aligned} u_i =& v_i((a^*_R)_i) - p(R)_i \\ =& \underbrace{v_i((a^*_R)_i) + \sum_{j \in N \setminus \{i\}} \hat{v}_j((a^*_R)_j)}_{\text{(a) Reported SW of main economy}} - \underbrace{\sum_{j \in N \setminus \{i\}} \hat{v}_j((a^*_{R_{-i}})_j)}_{\text{(b) Reported SW of marginal economy}}. \end{aligned}$$

Any beneficial misreport must increase the difference (a) − (b).

MLCA has two features that mitigate manipulations. First, MLCA explicitly queries each bidder's marginal economy (Algorithm 2, Line 5), which implies that (b) is practically independent of bidder i's bid (Section 7.3 in [Brero *et al.*, 2021] provides experimental support for this). Second, MLCA enables bidders to "push" information to the auction which they deem useful. This mitigates certain manipulations that target (a), as it allows bidders to increase (a) with truthful information. Brero *et al.* [2021] argue that any remaining manipulation would be implausible as it would require almost complete information.

If we are willing to make two assumptions, we also obtain a theoretical incentive guarantee. Assumption 1 requires that, if all bidders bid truthfully, then MLCA finds an efficient allocation (we show in Appendix E.2 that in two of our domains: GSVM, LSVM, we indeed find the efficient allocation in the majority of cases). Assumption 2 requires that, for all bidders i, if all other bidders report truthfully, then the social welfare of bidder i's marginal economy is independent of his value reports. If both assumptions hold, then bidding truthfully is an ex-post Nash equilibrium in MLCA.

C Monotone-Value Neural Networks

In this section, we provide proofs for all mathematical claims made in Section 3.1 (Appendix C.1, C.3 and C.4), provide a toy example on MVNNs (Appendix C.2) and give an overview of bounds tightening via *interval arithmetic* for the MVNN-based MILP (Appendix C.5).

C.1 Proof of Lemma 1

Lemma 1. *Let $\mathcal{N}_i^\theta : \mathcal{X} \to \mathbb{R}_+$ be an MVNN from Definition 1. Then it holds that $\mathcal{N}_i^{(W^i,b^i)} \in \mathcal{V}$ for all $W^i \geq 0$ and $b^i \leq 0$.*

Proof.

1. **Monotonicity (M)**:
   This property immediately follows, since (component wise) the weights $W^{i,k} \geq 0$ for all $k \in \{1\dots,K_i\}$ and $\varphi_{0,t}(z)$ is monotonically increasing.
2. **Normalization (N)**:
   Since $\varphi_{0,t}(z) = 0$ for $z \leq 0$ and the biases fulfill (component wise) $b^{i,k} \leq 0$, we can conclude that $\mathcal{N}_i^\theta(\underbrace{(0,\dots,0)}_{m\text{-times}}) = 0$.

□

Remark C.1 (The role of the activation-function in Lemma 1)**.** *For Lemma 1 it would be sufficient to only assume that the activation function is monotonically increasing to get (M) and maps negative numbers to zero to get (N). So ReLU would also be a valid activation function for Lemma 1, but not for Theorem 1 (see Remark C.2).*

C.2 Example MVNNs

The following example illustrates how we can capture complementarities, substitutabilities and independent items via an MVNN.

Example C.1 (MVNN)**.** *Consider the set of items $M = \{x_1, x_2, x_3\}$ and the associated (reported) value function $\hat{v}_i$ shown in Table 3 (where we use 001 as a shorthand notation for $(0,0,1)$):*

	000	100	010	001	110	101	011	111
$\hat{v}_i$	0	1	1	1	1	3	2	4

Table 3: Example on flexibility of MVNNs.

In this example, x_1 and x_2 are substitutes*, i.e., $2 = \hat{v}_i(\{x_1\}) + \hat{v}_i(\{x_2\}) > \hat{v}_i(\{x_1,x_2\}) = 1$; x_1 and x_3 are* complements*, i.e., $2 = \hat{v}_i(\{x_1\}) + \hat{v}_i(\{x_3\}) < \hat{v}_i(\{x_1,x_3\}) = 3$; and x_2 and x_3 are* independent*, i.e., $2 = \hat{v}_i(\{x_2\}) + \hat{v}_i(\{x_3\}) = \hat{v}_i(\{x_2,x_3\}) = 2$. This reported value function can be exactly captured by an MVNN $\mathcal{N}_i^\theta(x)$ in the following way:*

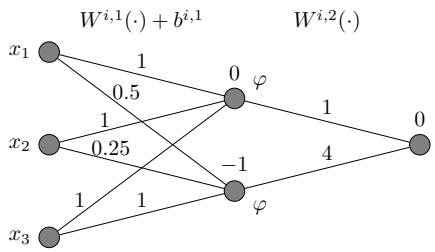


Figure 4: MVNN $\mathcal{N}_i^\theta$ with $\mathcal{N}_i^\theta(x) = \hat{v}_i(x) \quad \forall x \in \mathcal{X}$.

Biases are marked at the top of each neuron and weights are marked above the corresponding connections. A missing connection denotes a weight of 0. We see that the second kink (at $t = 1$) of the bReLU *together with the 0 bias of the top neuron in the hidden layer implements the substitutability between x_1 and x_2. Furthermore, the complementarity between x_1 and x_3 is implemented by the bottom neuron in the hidden layer via the negative bias -1 and the first kink (at 0) of the* bReLU.

C.3 Proof of Theorem 1

Theorem 1 (Universality)**.** Any *value function $\hat{v}_i : \mathcal{X} \to \mathbb{R}_+$ that satisfies **(N)** and **(M)** can be represented exactly as an MVNN $\mathcal{N}_i^\theta$ from Definition 1, i.e.,*

$$\mathcal{V} = \left\{\mathcal{N}_i^{(W^i,b^i)} : W^i \geq 0, b^i \leq 0\}\right\}. \tag{22}$$

Proof.

1. $\mathcal{V} \supseteq \left\{\mathcal{N}_i^{(W^i,b^i)} : W^{i,k} \geq 0, b^{i,k} \leq 0 \ \forall k \in \{1,\dots,K_i\}\right\}$

   This direction follows immediately from Lemma 1.

2. $\mathcal{V} \subseteq \left\{\mathcal{N}_i^{(W^i,b^i)} : W^{i,k} \geq 0, b^{i,k} \leq 0 \ \forall k \in \{1,\dots,K_i\}\right\}$

   Let $(\hat{v}_i(x))_{x\in\mathcal{X}} \in \mathcal{V}$. For the reverse direction, we give a constructive proof, i.e., we construct an MVNN $\mathcal{N}_i^\theta$ with $\theta = (W^i_{\hat{v}_i}, b^i_{\hat{v}_i})$ such that $\mathcal{N}_i^\theta(x) = \hat{v}_i(x)$ for all $x \in \mathcal{X}$.
   Let $(w_j)_{j=1}^{2^m}$ denote the values corresponding to $(\hat{v}_i(x))_{x\in\mathcal{X}}$ sorted in increasing order, i.e, let $x_1 = (0,\dots,0)$ with
   $$w_1 := \hat{v}_i(x_1) = 0, \tag{23}$$
   let $x_{2^m} = (1,\dots,1)$ with
   $$w_{2^m} := \hat{v}_i(x_{2^m}), \tag{24}$$
   and $x_j, x_l \in \mathcal{X} \setminus \{x_1, x_{2^m}\}$ for $1 < l \leq j \leq 2^m - 1$ with
   $$w_j := \hat{v}_i(x_j) \leq w_l := \hat{v}_i(x_l). \tag{25}$$

In the following, we slightly abuse the notation and write for $x_l, x_j \in \mathcal{X}$ $x_l \subseteq x_j$ iff for the corresponding sets $A_j, A_l \in 2^M$ it holds that $A_j \subseteq A_l$. Furthermore, we denote by $\langle \cdot, \cdot \rangle$ the Euclidean scalar product on $\mathbb{R}^m$. Then, for all $x \in \mathcal{X}$:

$$\hat{v}_i(x) = \sum_{l=1}^{2^m-1} (w_{l+1} - w_l) \, \mathbb{1}_{\{\forall j \in \{1,\dots,l\} \,:\, x \not\subseteq x_j\}} \tag{26}$$

$$= \sum_{l=1}^{2^m-1} (w_{l+1} - w_l) \, \varphi_{0,1} \left(\sum_{j=1}^{l} \varphi_{0,1} \left(\langle 1 - x_j, x \rangle \right) - (l-1) \right), \tag{27}$$

where the second equality follows since

$$x \not\subseteq x_j \iff \langle 1 - x_j, x \rangle \geq 1 \tag{28}$$

$$\iff \varphi_{0,1} \left(\langle 1 - x_j, x \rangle \right) = 1, \tag{29}$$

which implies that

$$\forall j \in \{1, \dots, l\} : x \not\subseteq x_j \iff \sum_{j=1}^{l} \varphi_{0,1} \left(\langle 1 - x_j, x \rangle \right) = l, \tag{30}$$

and

$$\mathbb{1}_{\{\forall j \in \{1,\dots,l\} \,:\, x \not\subseteq x_j\}} = \varphi_{0,1} \left(\sum_{j=1}^{l} \varphi_{0,1} \left(\langle 1 - x_j, x \rangle \right) - (l-1) \right) \tag{31}$$

Finally, Equation (27) can be equivalently written in matrix notation as

$$\underbrace{\begin{bmatrix} w_2 - w_1 \\ w_3 - w_2 \\ \vdots \\ w_{2^m} - w_{2^m-1} \end{bmatrix}^T}_{(W^{i,3}_{\hat{v}_i})^T \in \mathbb{R}^{2^m-1}_{\geq 0}} \varphi_{0,1} \left(W^{i,2}_{\hat{v}_i} \varphi_{0,1} \left(\underbrace{\begin{bmatrix} 1 - x_1 \\ 1 - x_2 \\ \vdots \\ 1 - x_{2^m-1} \end{bmatrix}}_{W^{i,1}_{\hat{v}_i} \in \mathbb{R}^{(2^m-1)\times m}_{\geq 0}} x \right) + \underbrace{\begin{bmatrix} 0 \\ -1 \\ \vdots \\ -(2^m-2) \end{bmatrix}}_{b^{i,2}_{\hat{v}_i} \in \mathbb{R}^{2^m-1}_{\leq 0}} \right)$$

with $W^{i,2}_{\hat{v}_i} \in \mathbb{R}^{(2^m-1)\times(2^m-1)}_{\geq 0}$ a lower triangular matrix of ones, i.e.,

$$W^{i,2}_{\hat{v}_i} := \begin{bmatrix} 1 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & 0 \\ 1 & \dots & \dots & 1 \end{bmatrix}.$$

From that, we can see that the last term is indeed an MVNN $\mathcal{N}^\theta_i(x) = W^{i,3}_{\hat{v}_i} \varphi_{0,1} \left(W^{i,2}_{\hat{v}_i} \varphi_{0,1} \left(W^{i,1}_{\hat{v}_i} x \right) + b^{i,2}_{\hat{v}_i} \right)$ with four layers in total (i.e., two hidden layers) and respective dimensions $[m, 2^m - 1, 2^m - 1, 1]$.

□

Remark C.2 (The role of the activation-function in Theorem 1)**.** *If the classical ReLU-activation function was used, we would not get universality, because of the non-negativity constraints of the weights: A ReLU-network with non-negative weights can only express convex functions.*[9] *This means that it could only express complementarities, but no substitutabilities. For approximate universality any monotonically increasing, bounded, non-constant activation-function would be sufficient (e.g., bReLU or sigmoid). However, sigmoid would not be a valid activation function for Theorem 1, because it would not allow* exact *normalization for a non-zero function. (Firstly, for an activation function such as sigmoid that does not map negative number to zero, networks that fulfil the sign constraints for the weights and biases could be arbitrarily far away from being normalized. Secondly, for universality, one could only approximate the true function up to an arbitrarily small epsilon but never exactly for a normalized non-constant true function.) While these problems of sigmoid-activation-functions for Theorem 1 might be rather theoretical, what is more important in practice is that non-piecewise-linear activations functions are not MILP-formalizable.*

Remark C.3 (Why bReLU?)**.** *Every monotonically increasing, non-constant, bounded activation function that maps negative numbers to 0 would be a valid choice, since it would fulfill Lemma 1 (see Remark C.1), Theorem 1 (see Remark C.2) and be MILP-formalizable (cp. Theorem 2). The complexity of the MILP grows with the number of kinks in the activation function. In this sense, bReLU is the simplest monotonically increasing, non-constant, bounded activation function that maps negative numbers to 0.*[10] *In addition, bReLU shares many properties of ReLU and sigmoid, which are both popular choices for activation-functions. Remark C.2 explains why ReLU or sigmoid would not be valid choices.*

Corollary C.1. *For every dataset* $((x_j), \hat{v}_i(x_j))_{j \in \{1\dots,q\}}$ *there exist an MVNN of dimensions* $[m, q, q, 1]$ *that perfectly fits the dataset, i.e.* $\mathcal{N}^\theta_i(x_j) = \hat{v}_i(x_j) \; \forall j \in \{1 \dots, q\}$.

Proof. This interpolating MVNN can be explicitly constructed such as the MVNN in the proof of Theorem 1 by replacing the collection of all bundles and their values by the dataset $((x_j), \hat{v}_i(x_j))_{j \in \{0\dots,q\}}$, where x_0 is the empty bundle and $\hat{v}_i(x_0) = 0$, i.e., Equation (27) has to be replaced by:

$$\mathcal{N}^\theta_i(x) = \sum_{l=1}^{q} (w_l - w_{l-1}) \, \varphi_{0,1} \left(\sum_{j=0}^{l-1} \varphi_{0,1} \left(\langle 1 - x_j, x \rangle \right) - (l-1) \right), \tag{32}$$

[9] A ReLU-MVNN could only express convex functions, since ReLU is convex and non-decreasing, and linear combinations of convex non-decreasing functions are convex and non-decreasing if all the coefficients are non-negative, and compositions of convex non-decreasing functions are convex (and non-decreasing). Thus any ReLU-MVNN with our weight constraints could never express any non-convex functions. However, agent's value functions are very often non-convex, i.e., they typically admit substitutabilities. E.g. Example C.1 could not be expressed by a ReLU-MVNN.

[10] One can easily see that any bounded piece-wise linear non-constant activation function has to have at least two kinks, so at least 3 linear segments. bReLU has two kinks (one at zero and one at the cut-off t), so there is no other suitable activation function with less kinks.

where we again assume that the bundles are sorted by their values and for $0 < l \leq j \leq q$ we define:

$$w_j := \hat{v}_i(x_j) \;\leq\; w_l := \hat{v}_i(x_l). \tag{33}$$

□

Remark C.4 (Number of neurons needed)**.** *$[m, 2^m - 1, 2^m - 1, 1]$ is just an upper bound for the size of the network to learn the true value function exactly on $\mathcal{X}$. For NNs on a continuous domain no finite upper bound exist, but they are still widely used in practice. Most value functions could be expressed with significantly fewer nodes.*

More importantly, in an ICA (e.g., MLCA) one doesn't need to learn the value function everywhere perfectly but only approximately where the precision mainly matters for relevant bundles. Corollary C.1 states that for q queries, there is an MVNN with size $[m, q, q, 1]$ that can perfectly fit through these q queries. So we have a mathematical guarantee that the number of nodes only grows linearly *with the number of queries in the worst case. This is the much more relevant growth rate in practice, because we will typically not be able to do $2^m - 1$ queries anyway and thus the MILP will also never be exponentially large. Thus, for MVNNs with size $[m, 100, 100, 1]$, we have a mathematical guarantee that we can always get a perfect monotonic normalized interpolation of all available 100 queries even in the worst cases. However, our experiments show that for data coming from realistic value functions, significantly smaller architectures are sufficient to (almost) perfectly interpolate the training data most of the time and even often almost perfectly approximate the true value function on $\mathcal{X}$ (see Tables 7 and 8) leading to better than state-of-the-art performance in MLCA (see Tables 2 and 5). This allows very fast MILP-solving times in practice (see Table 5, where the average run-times of full auctions are given in hours or Figure 3, where the average run-times per MILP are given in seconds).*

C.4 Proof of Lemma 2

In this section, we proove Lemma 2 from Section 3.

Proof. For all $j \in \{1, \ldots, d^{i,k}\}$ we distinguish the following three cases:

Case 1: $o_j^{i,k} \in (-\infty, 0] \Rightarrow \varphi(o^{i,k})_j = 0$

$$(6) \;\&\; (7) \Rightarrow y_j^{i,k} = 1 \Rightarrow \eta_j^{i,k} = 0$$

$$(8) \;\&\; (9) \Rightarrow \mu_j^{i,k} = 0 \Rightarrow z_j^{i,k} = \eta_j^{i,k} = 0$$

Case 2: $o_j^{i,k} \in (0, 1] \Rightarrow \varphi(o^{i,k})_j = o_j^{i,k}$

$$(6) \;\&\; (7) \Rightarrow y_j^{i,k} = 0 \Rightarrow \eta_j^{i,k} = o_j^{i,k}$$

$$(8) \;\&\; (9) \Rightarrow \mu_j^{i,k} = 0 \Rightarrow z_j^{i,k} = \eta_j^{i,k} = o_j^{i,k}$$

Case 3: $o_j^{i,k} \in (1, +\infty] \Rightarrow \varphi(o^{i,k})_j = 1$

$$(6) \;\&\; (7) \Rightarrow y_j^{i,k} = 0 \Rightarrow \eta_j^{i,k} = o^{i,k}$$

$$(8) \;\&\; (9) \Rightarrow \mu_j^{i,k} = 1 \Rightarrow z_j^{i,k} = 1$$

Thus, in total $z^{i,k} = \varphi(o^{i,k})$. □

C.5 Interval Arithmetic Bounds Tightening for MVNNs

In this section, we consider a bReLU $\varphi_{0,t}$ with cutoff $t > 0$ and mark it in red. This helps when implementing the MILP, i.e. it particularly shows more clearly where the cutoff t propagates in the respective equations. First, we recall Lemma 1 for a general cutoff t, where for the sake of readability we remove the bidder index $i \in N$ from all variables.

Lemma 2. *Let $k \in \{1, \ldots, K-1\}$ and let the pre-activated output of the kth layer be given as $W^k z^{k-1} + b^k$ with $W^k \in \mathbb{R}^{d^k \times d^{k-1}}, b^k \in \mathbb{R}^{d^k}$. Then the output of the kth layer $z^k := \varphi_{0,t}(W^k z^{k-1} + b^k) = \min(t, \max(0, (W^k z^{k-1} + b^k)) = -\max(-t, -\eta^k)$, with $\eta^k := \max(0, W^k z^{k-1} + b^k)$ can be equivalently expressed by the following linear constraints:*

$$W^k z^{k-1} + b^k \leq \eta^k \leq W^k z^{k-1} + b^k + y^k L_1^k \tag{34}$$

$$0 \leq \eta^k \leq (1 - y^k) L_2^k \tag{35}$$

$$\eta^k - \mu^k L_3^k \leq z^k \leq \eta^k \tag{36}$$

$$t - (1 - \mu^k) L_4^k \leq z^k \leq t \tag{37}$$

where $y^k \in \{0,1\}^{d^k}$, $\mu^k \in \{0,1\}^{d^k}$, and $L_1^k, L_2^k, L_3^k, L_4^k \in \mathbb{R}_+$ are large enough constants for the respective big-M *constraints.*

Interval Arithmetic (IA) for Single Neurons

For any $k \in \{1, \ldots, K-1\}$ let $W^k \geq 0$ and $b^k \leq 0$ be the weights of its affine linear transformation. Furthermore, let z^{k-1} be the output of the previous layer. and, let

$$z^k = \varphi_{0,t}\left(W^k z^{k-1} + b^k\right) \tag{38}$$

be the output of the current layer.

Given already computed IA bounds for z_i^{k-1}, i.e., $[L(z_l^{k-1}), U(z_l^{k-1})] \subseteq [0, t]$, we can then calculate the IA bounds $L(z_l^k), U(z_l^k)$ for z_l^k such that

$$z^k \in \prod_{l=1}^{d^k} [L(z_l^k), U(z_l^k)] \tag{39}$$

as follows:

1. **Upper bound (pre-activated):**

$$U^{\text{pre}}(z^k) = \tag{40}$$

$$= \max_{z^{k-1} \in \prod_{l=1}^{d^{k-1}} [L(z_l^{k-1}), U(z_l^{k-1})]} \left\{ W^k z^{k-1} + b^k \right\} \tag{41}$$

$$= \left(\sum_l W_{j,l}^k \cdot U(z_l^{k-1}) \right)_{j=1}^{d^k} + b^k \tag{42}$$

2. **Upper bound:**

$$U(z^k) = \varphi_{0,t}\left(U^{\text{pre}}(z^k)\right) \tag{43}$$

3. **Lower bound (pre-activated):**

$$L^{\text{pre}}(z^k) = \tag{44}$$

$$= \min_{z^{k-1} \in \prod_{l=1}^{d^{k-1}} [L(z_l^{k-1}), U(z_l^{k-1})]} \left\{ W^k z^{k-1} + b^k \right\} \tag{45}$$

$$= \left(\sum_l W_{j,l}^k \cdot L(z_l^{k-1}) \right)_{j=1}^{d^k} + b^k \tag{46}$$

Note, that the preactivated lower bound $L^{\text{pre}}(z^k)$ is always non-positive, i.e. $L^{\text{pre}}(z^k) \leq 0$. This can be seen as follows: start with a $L(z^1) = \mathbf{0}_{d^1}$. Then since all biases are non-positive $L^{\text{pre}}(z^2) \leq 0 \implies L(z^2) = \mathbf{0}_{d^2}$, and so forth. Thus, we get for the lower bounds:

4. **Lower bound:**

$$L(z^k) = \varphi_{0,t}\left(L^{\text{pre}}(z^k)\right) = \mathbf{0}_{d^k} \tag{47}$$

Removing Constraints with IA

In the following cases, we can remove the constraints and corresponding variables in Lemma 2:

1. **Case:**
If $U^{\text{pre}}(z_l^k) \leq 0 \implies z_l^k = 0$ and one can remove the lth components from all constraints (34) – (37) and the corresponding variables for layer k.
2. **Case:**
If $L^{\text{pre}}(z_l^k) \in [0, t)$ and $U^{\text{pre}}(z_l^k) \in (0, t] \implies z_l^k = \left(W^k z^{k-1} + b^k\right)_l$ and one and can remove the lth components from all constraints (34) – (37) and the corresponding variables for layer k.
3. **Case:**
If $L^{\text{pre}}(z_l^k) = 0$ and $U^{\text{pre}}(z_l^k) > t \implies \eta_l^k = \left(W^k z^{k-1} + b^k\right)_l$ and one and can remove the lth components from all constraints (34) – (35) and the corresponding variables for layer k.
4. **Case:**
If $L^{\text{pre}}(z_l^k) < 0$ and $U^{\text{pre}}(z_l^k) \in (0, t] \implies z_l^k = \eta_l^k$ and one and can remove the lth components from all constraints (36) – (37) and the corresponding variables for layer k.[11]

Fact C.1. *For a plain ReLU NN, the above calculated IA bounds are not tight and calculating tighter bounds in a computationally efficient manner is very challenging and an open research question. However, for MVNNs, the IA bounds are always* perfectly tight*, because of their encoded monotonicity property. This is a big advantage of MVNN-based MILPs compared to plain (ReLU) NN-based MILPs.*

Interval Arithmetic for the Four Big-M Constraints

Recall, that $L(z^k) = \mathbf{0}_{d^k}$ for all $k \in \{1, \ldots, K-1\}$. Let $m =: d^0$ denote the number of items to be allocated.

We present standard IA bounds where one starts for $z^0 \in \{0,1\}^m$ with $L(z^0) = \mathbf{0}_m$ and $U(z^0) = \mathbf{1}_m$ (per definition) and iteratively for $k \in \{1, \ldots, K-1\}$ *propagates* through the network for given bounds on z^{k-1}, i.e., $z^{k-1} \in \mathcal{Z}^k := \prod_{l=1}^{d^{k-1}} [0, U(z_l^{k-1})] \subset [0, t]^{d^{k-1}}$.

1. L_1^k only appears when $y^k = 1$, which implies that $\eta^k = 0$ and $W^k z^{k-1} + b^k \leq 0$. Thus, Equation (34) implies that $L_1^k + \min_{z^{k-1} \in \mathcal{Z}^{k-1}} \left\{W^k z^{k-1} + b^k\right\} \geq 0$ and we get

$$L_1^k = \max\left\{0, -\min_{z^{k-1} \in \mathcal{Z}^{k-1}} \left\{W^k z^{k-1} + b^k\right\}\right\} \tag{48}$$

$$= \max\left\{0, \left(\sum_{l: W_{j,l}^k < 0} |W_{j,l}^k| \cdot U(z_l^{k-1})\right)_{j=1}^{d^k} - b^k\right\} \tag{49}$$

$$= \max\left\{0, -b^k\right\} \tag{50}$$

where the last equality follows since per definition any MVNN only has positive weights.

2. L_2^k only appears when $y^k = 0$ which implies that $\eta^k = W^k z^{k-1} + b^k \geq 0$. Thus, Equation (35) implies that $L_2^k \geq \eta^k = W^k z^{k-1} + b^k$ and we get

$$L_2^k = \max_{z^{k-1} \in \mathcal{Z}^{k-1}} \left\{W^k z^{k-1} + b^k\right\} \tag{51}$$

$$= \max\left\{0, \left(\sum_{l: W_{j,l}^k > 0} W_{j,l}^k \cdot U(z_l^{k-1})\right)_{j=1}^{d^k} + b^k\right\} \tag{52}$$

$$= \max\left\{0, \left(\sum_{l} W_{j,l}^k \cdot U(z_l^{k-1})\right)_{j=1}^{d^k} + b^k\right\}, \tag{53}$$

where the last equality follows since per definition any MVNN only has positive weights.

3. L_3^k only appears when $\mu^k = 1$ which implies that $z^k = t$ and $\eta^k = W^k z^{k-1} + b^k \geq t$. Thus, Equation (36) implies that $t = z^k \geq \eta^k - L_3^k \iff L_3^k \geq \eta^k - t \iff L_3^k \geq W^k z^{k-1} + b^k - t$ and we get

$$L_3^k = \max_{z^{k-1} \in \mathcal{Z}^{k-1}} \left\{W^k z^{k-1} + b^k - t\right\} \tag{54}$$

$$= \max\left\{0, L_2^k - t\right\} \tag{55}$$

4. For L_4^k only appears when $\mu^k = 0$. In this case, we get from Equation (37) that $t - L_4^k \leq z^k$ and we get

$$L_4^k = \max_{z^k \in \mathcal{Z}^k} \{t - z^k\} = t \tag{56}$$

C.6 Implementation Details - MVNNs

Recall that the three key building blocks of MVNNs are the bReLU φ, the (element-wise) non-negative weights W^i, and the (element-wise) non-positive biases b^i. The bReLU φ can be straightforwardly implemented as a custom activation function in PyTorch 1.8.1. However, the constraints on the weights and biases to be element-wise positive and negative respectively, can be implemented in several different ways. We first describe the method we call MVNN-ReLU-Projected, which we have found experimentally to perform best. Finally, we describe the other options we have explored since they may be of independent interest to some readers.

MVNN-ReLU-Projected In this implementation, we project W^i and b^i prior to every forward pass to be non-negative and non-positive, respectively, using ReLU $z \mapsto$

[11] In all experiments presented in this paper, we have not taken advantage of Case 3 and 4. We believe that we can further improve the computational performance of MVNN by incorporating these cases too.

$\pm \max(0, \pm z)$. Thus, (in contrast to some of the other methods we explored), gradient descent (GD) updates are performed on the already transformed weights and biases and we *do not differentiate* through the ReLUs. After the last GD step, we apply *post-processing* and project W^i and b^i again via $z \mapsto \pm \max(0, \pm z)$.

MVNN-ABS and MVNN-RELU For these methods, we add an additional node to the computational graph and element-wisely transform the weights W^i via the absolute value $z \mapsto |z|$ or ReLU $z \mapsto max(0, z)$ to ensure non-negativity. For the biases b^i we analogously use $z \mapsto -|z|$ or $z \mapsto -max(0, -z)$ to ensure non-positivity. Importantly, we *differentiate* through $z \mapsto \pm|z|$ or $z \mapsto \pm \max(0, \pm z)$ in every gradient descent (GD) step. We refer to these implementation variants as MVNN-ABS and MVNN-RELU. After the last GD step we apply *post-processing* and project W^i and b^i again via $z \mapsto \pm|z|$ and $z \mapsto \pm \max(0, \pm z)$.

MVNN-ABS-PROJECTED We do not consider a MVNN-ABS-PROJECTED implementation, since one can prove that in the classic GD algorithm this version is mathematically equivalent to MVNN-ABS.

D Details Prediction Performance

D.1 Data Generation - Prediction Performance

Train/Val/Test-Split For every domain, bidder type and considered amount of training data T we create the data in the following way: For each seed we let SATS create a value function and uniformly at random select T different bundles from the bidder-specific-feasible (using the SATS method *get_uniform_random_bids*) bundle space $\mathcal{X}$ (training set). We measure the metrics of a method trained on the training data of a seed based on randomly selected different bundles from the same bundle space $\mathcal{X}$ (approx. 52 000 for the HPO seeds (validation set) and approx. 210 000 for the test seeds (test set)). We use HPO seeds 0-20 only for the HPO (validation sets) and test seeds 21-50 only for reporting the values in this paper (test sets).

D.2 HPO - Prediction Performance

Table 4 shows the hyperparameter ranges from our HPO. Cosine Annealing [Loshchilov and Hutter, 2017] was used to decay the learning to zero within the number of training epochs. Occasionally, the neural networks (both MVNNs and plain NNs) diverged during training (determined if *the Pearson correlation coefficient* $r_{xy} < 0.9$ on the train set) hence we added 20 retries to the training to make sure that we obtain a valid network.

Experimentally, we found that MVNNs need the training data to be normalized to be within [0, 1], as a result of the bReLU activation functions bounded output. On the other hand, plain NNs worked better if the data was normalized to be within [0, 500] for all considered SATS domains.

The computational budget of the HPO was 12 hours. All experiments were conducted on a compute cluster running Debian GNU/Linux 10 (buster) with Intel Xeon E5-2650 v4 2.20GHz processors with 24 cores and 128GB RAM and Intel E5 v2 2.80GHz processors with 20 cores and 128GB RAM and Python 3.7.10.

Hyperparameters	Type	Range	Scale
Optimizer	Categorical	[Adam, SGD]	
Batch Size	Integer	[1, 4]	
Num. Hidden Layers	Integer	[1, 3]	
Total Num. Neurons	Integer	[1, 64]	
L2	Float	[1e-10, 1e-6]	log
Learning Rate	Float	[1e-4, 1e-2]	log
Epochs	Integer	[50, 400]	
Loss Function	Categorical	[MSE, MAE]	

Table 4: HPO space used in SMAC.

D.3 Detailed Results - Prediction Performance

Table 7 shows the detailed prediction performance results including all optimized hyperparameters for the MVNN-ABS and the MVNN-RELU-PROJECTED implementation of MVNNs. The MVNN-RELU implementation led to similar results as the MVNN-RELU-PROJECTED implementation, and therefore we do not present them in this table.

In Table 8, we visualize for all other SATS domains the prediction performance capabilities of MVNNs vs plain NNs. Overall, Table 7 and 8 show that MVNNs have a superior generalization performance across all SATS domains and bidder types.

E Details MVNN-based Iterative CA

E.1 Details Experimental Setup - MVNN-based Iterative CA

We used 50 auction instances for evaluation with seeds (for generating the SATS instances) 10001-10050 which do not intersect with the seeds used in Appendix D.1 for prediction performance. All experiments were conducted on the same compute cluster as in Appendix D.2.

Following prior work [Brero *et al.*, 2021] we set $Q^{\text{round}} = 4$ (see Algorithm 2), i.e., in each iteration of MLCA we ask each bidder 1 query in the main economy (including all bidders) and 3 queries from randomly sampled marginal economies (excluding one bidder). This choice ensures a trade-off between efficiency (more main economy queries decrease the efficiency loss) and revenue (more marginal queries increase revenue). Note that we add the empty bundle to the initial elicited bundles Q^{init}, as we know its value to be 0 a priori. This has no impact for MVNNs as they estimate empty bundles to have zero value by definition but adds extra prior information for plain NNs.

For all SATS domains we specified a minimum relative gap of 1e-2 and a timeout of 300s.

E.2 Detailed Results - MVNN-based Iterative CA

In Table 5, we present detailed results of MLCA with MVNNs vs MLCA with plain NNs. The revenue and runtime should be considered with care. The revenue is influenced by our choice to use early stopping whenever we already found an efficient allocation at an intermediary iteration as discussed in the main paper. The shown runtime is not comparable between MVNNs and NNs as they do not use the same architecture, but the one found during HPO. See Section 5.3 in the main paper for a fair runtime comparison.

					Efficiency Loss in % ↓		Revenue in % ↑		Runtime in Hrs.	
Domain	**T**	Q^{INIT}	Q^{ROUND}	Q^{MAX}	MVNN	NN	MVNN	NN	MVNN	NN
GSVM	10	40	4	100	00.00 ± 0.00	00.01 ± 0.02	60.11 ± 3.86	58.59 ± 4.35	00.14 ± 0.03	00.15 ± 0.05
	20	40	4	100	00.00 ± 0.00	*00.00 ± 0.00	59.07 ± 3.84	55.71 ± 4.46	00.08 ± 0.01	00.06 ± 0.01
	50	40	4	100	*00.00 ± 0.00	00.00 ± 0.00	52.77 ± 5.04	56.09 ± 4.57	00.06 ± 0.01	00.07 ± 0.01
LSVM	10	40	4	100	01.63 ± 0.75	03.19 ± 01.59	70.14 ± 4.23	65.23 ± 3.83	00.93 ± 0.36	01.16 ± 0.23
	50	40	4	100	*00.70 ± 0.40	03.11 ± 01.52	70.70 ± 4.63	64.07 ± 4.24	00.54 ± 0.15	01.28 ± 0.32
	100	40	4	100	01.27 ± 0.55	*02.91 ± 01.44	71.09 ± 4.35	65.10 ± 3.90	00.52 ± 0.12	00.44 ± 0.10
SRVM	10	40	4	100	00.67 ± 0.10	01.27 ± 0.23	50.86 ± 2.31	46.93 ± 2.07	01.86 ± 0.10	01.59 ± 0.09
	50	40	4	100	00.49 ± 0.07	01.15 ± 0.31	50.70 ± 2.44	45.59 ± 2.72	20.72 ± 0.52	02.17 ± 0.21
	100	40	4	100	*00.23 ± 0.06	*01.13 ± 0.22	51.18 ± 2.52	46.30 ± 2.87	20.12 ± 1.77	00.68 ± 0.06
MRVM	10	40	4	100	*08.16 ± 0.41	10.96 ± 0.76	34.45 ± 2.06	41.68 ± 1.34	02.59 ± 0.13	06.08 ± 0.40
	100	40	4	100	08.67 ± 0.43	10.19 ± 0.89	35.04 ± 1.91	41.91 ± 1.39	01.96 ± 0.12	03.33 ± 0.37
	300	40	4	100	09.96 ± 0.49	*09.05 ± 0.53	32.72 ± 2.17	39.66 ± 1.20	00.96 ± 0.04	01.84 ± 0.24

Table 5: Efficiency loss, relative revenue and runtime of MLCA with MVNNs vs MLCA with plain NNs. Shown are averages including a 95% CI on a test set of 50 auction instances in all four SATS domains. The best MVNN and plain NN per domain based on the lowest efficiency loss are marked with a star (if the final efficiency loss is the same for multiple incumbents we selected the incumbent that reached 0% with the fewest number of queries).

For the presentation in the main paper (i.e., Table 2 in the main paper), we selected the best MVNN and plain NN per domain, i.e. the ones marked with a star. In Figures 5, 6, 7 and 8 we present a detailed boxplot version of the efficiency loss path plots for these best models per domain.

E.3 Detailed Revenue Analysis - MVNN-based Iterative CA

Recall the VCG-payments from Definition B.1, i.e., for a set of elicited reports R bidder i's VCG-payment is given as:

$$p_i(R) := \underbrace{\sum_{j \in N \setminus \{i\}} \hat{v}_j \left(\left(a^*_{R_{-i}} \right)_j \right)}_{=:\text{Marg}_{-i}} - \underbrace{\sum_{j \in N \setminus \{i\}} \hat{v}_j \left((a^*_R)_j \right)}_{=:\text{Main}_{-i}},$$

where Marg_{-i} is for a given set of reports R the optimal social welfare (SCW) in the *marginal* economy when excluding bidder $i \in N$ and Main_{-i} denotes the SCW of the *main* economy when excluding bidder i. The larger the differences $\text{Marg}_{-i} - \text{Main}_{-i}, i \in N$ the more revenue is generated in the auction. In Table 6, we print the normalized average marginal SCW $(\frac{1}{n} \sum_{i \in N} \text{Marg}_{-i})/V(a^*)$ (**Avg. Marg. SCWs (%)**) and the normalized average main SCW when excluding one bidder $(\frac{1}{n} \sum_{i \in N} \text{Main}_{-i})/V(a^*)$ (**Avg. Main SCWs (%)**) averaged over the 50 auction instances corresponding to Table 5. Using this notation, the average total relative revenue of the auction is then the difference of the normalized average

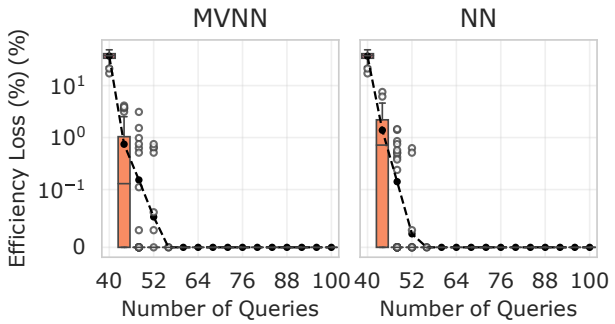


Figure 5: Efficiency loss path of MVNN vs plain NN in GSVM with the corresponding best MVNN ($T = 50$) and plain NN ($T = 20$) from Table 5. Averages are shown as black dots. We use a semi-logarithmic scale with linear range [0,1e-2].

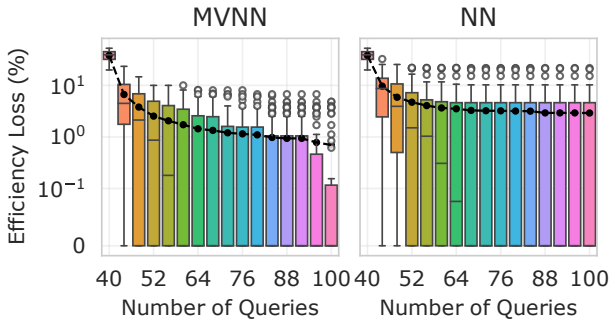


Figure 6: Efficiency loss path of MVNN vs plain NN in LSVM with the corresponding best MVNN ($T = 50$) and plain NN ($T = 100$) from Table 5. Averages are shown as black dots. We use a semi-logarithmic scale with linear range [0,1e-1].

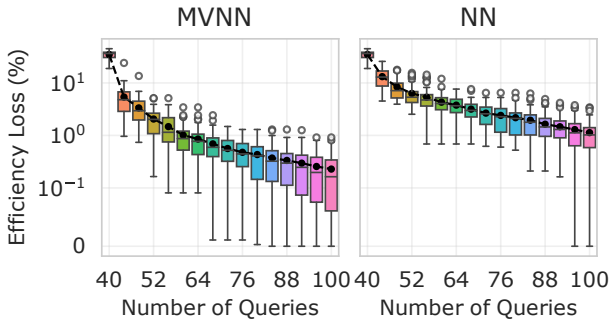


Figure 7: Efficiency loss path of MVNN vs plain NN in SRVM with the corresponding best MVNN ($T = 100$) and best plain NN ($T = 100$) from Table 5. Averages are shown as black dots. We use a semi-logarithmic scale with linear range [0,1e-1].

		Number of Queries			Avg. Marg. SCWs (%)			Avg. Main SCWs (%)			Revenue in % ↑		
Domain	**T**	MVNN	NN	RS	MVNN	NN	RS	MVNN	NN	RS	MVNN	NN	RS
GSVM	10	59.1	52.6	100.0	94.31	94.09	67.05	85.71	85.71	59.90	60.11 ± 3.86	58.59 ± 4.35	52.19 ± 2.39
	20	54.3	47.8	100.0	94.16	93.69	67.05	85.71	85.71	59.90	59.07 ± 3.84	55.71 ± 4.46	52.19 ± 2.39
	50	47.2	48.3	100.0	93.25	93.74	67.05	85.71	85.71	59.90	52.77 ± 5.04	56.09 ± 4.57	52.19 ± 2.39
LSVM	10	74.2	75.5	100.0	93.58	91.41	64.57	81.95	80.62	55.67	70.14 ± 4.23	65.23 ± 3.83	53.58 ± 1.84
	50	68.6	73.2	100.0	94.48	91.28	64.57	82.74	80.69	55.67	70.70 ± 4.63	64.07 ± 4.24	53.58 ± 1.84
	100	73.2	73.0	100.0	94.04	91.63	64.57	82.26	80.86	55.67	71.09 ± 4.35	65.10 ± 3.90	53.58 ± 1.84
SRVM	10	100.0	100.0	100.0	92.53	91.34	69.43	85.13	84.62	62.12	50.86 ± 2.31	46.93 ± 2.07	51.56 ± 2.07
	50	99.4	99.9	100.0	92.66	91.35	69.43	85.29	84.72	62.12	50.70 ± 2.44	45.59 ± 2.72	51.56 ± 2.07
	100	98.9	99.8	100.0	92.98	91.47	69.43	85.52	84.74	62.12	51.18 ± 2.52	46.30 ± 2.87	51.56 ± 2.07
MRVM	10	100.0	100.0	100.0	86.12	84.39	50.39	82.68	80.22	46.03	34.45 ± 2.06	41.68 ± 1.34	43.58 ± 0.65
	100	100.0	100.0	100.0	85.70	85.04	50.39	82.22	80.83	46.03	35.04 ± 1.91	41.91 ± 1.39	43.58 ± 0.65
	300	100.0	100.0	100.0	84.29	85.86	50.39	81.03	81.87	46.03	32.72 ± 2.17	39.66 ± 1.20	43.58 ± 0.65

Table 6: Normalized average social welfare in the marginal economies and main economies when excluding one bidder for MVNNs, NNs and random search (RS). Additionally, we print the average number of queries, that can differ between MVNN and NN incumbents due to early termination of MLCA. Shown are averages over the same test set of 50 auction instances as in Table 5.

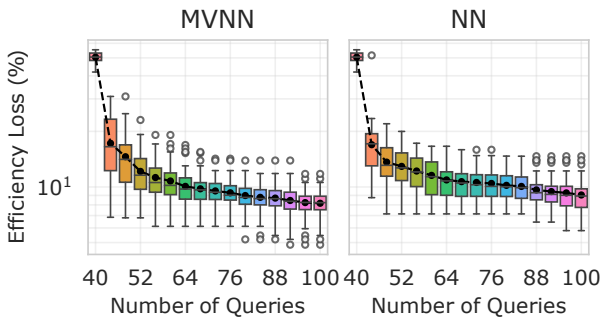


Figure 8: Efficiency loss path of MVNN vs plain NN in MRVM with the corresponding best MVNN ($T = 10$) and plain NN ($T = 300$) from Table 5. Averages are shown as black dots. Only logarithmic scale.

marginal SCW and the normalized average main SCW when excluding one bidder times the number of bidders, i.e.,

$$n\left((\frac{1}{n}\sum_{i\in N}\text{Marg}_{-i})/V(a^*) - (\frac{1}{n}\sum_{i\in N}\text{Main}_{-i})/V(a^*)\right).$$

Recall, that we terminate MLCA in an intermediate iteration, if it already found an efficient allocation (to save computational costs), i.e., incurred no efficiency loss. Due to this early termination the revenue can be worse off since fewer bundles are elicited in the marginal economies. If one runs a full auction without early termination, the revenue can only improve after the efficient allocation was already found (because Main_{-i} cannot increase anymore after the efficient allocation was already found, but Marg_{-i} can still improve).

In Table 6, we present a detailed revenue analysis for all domains. Table 6 shows that overall the MVNN's achieved (normalized) SCWs are larger both in the marginal economies as well as in the main economies when excluding one bidder.

In GSVM, we find the efficient allocation very early (see Figure 5 or Table 6), so the revenue is mainly determined by the number of queries. Without early determination, 60% revenue should be easily achievable for MVNNs. For LSVM and SRVM, we see that even with less queries MVNNs can outperform plain NNs consistently in terms of revenue. Without early termination, we expect this margin to be even higher (because this would add more queries for MVNNs).

In MRVM, we see that the best MVNN ($T = 10$) achieves an average difference of $86.12 - 82.68 = 3.44$ while the best plain NN (T=300) achieves an average difference of $85.86 - 81.87 = 3.98$. This implies the larger revenue ($\approx 5\% = 10 \cdot 0.5\%$) of plain NNs in MRVM from Table 2 in the main paper. A possible explanation for the larger revenue of plain NNs in MRVM is that, with each query MVNNs much more exploit the economy for that we solve the WDP, while plain NNs are more random and thus the query that solves the WDP for the main economy is not perfectly specialized for the main economy, but only slightly more helpful for the main economy than for the other economies. Recall that MLCA asks in each iteration each bidder Q^{round}-many queries: one main economy query and $Q^{\text{round}} - 1$ marginal economy queries. Thus, if $Q^{\text{round}} < n$, MLCA generates more queries for the main economy than for each marginal economy and MVNNs improve the main economy even more than they improve the marginal economies, since their queries are highly specialized and exploiting. This effect is the strongest for MRVM as it has the largest ratio of the number of bidders n and Q^{round}.

However, if the objective would be to maximize revenue rather than efficiency, MVNNs could achieve as good or better revenue than plain NNs when we set $Q^{\text{round}} = n$, since then both main and marginal economies would be equally improved by the advantages of MVNNs (again with $Q^{\text{round}} = 4$ the main economy profits more from the advantages of the MVNNs than the marginal economies; see Table 6).

As expected, for random search (RS) the (normalized) SCWs in both the marginal economies and the main economies when excluding one bidder are much lower compared to MVNNs and NNs. However, since *each* economies' SCW is bad, their differences and thus the revenue does not have to be worse. Moreover, since RS treats all economies equally and particularly is not specialized towards the main economies (in contrast to MVNN-MLCA or NN-MLCA with $Q^{\text{round}} < n$), RS's revenue can be even higher compared to MVNNs and NNs (e.g., in SRVM and MRVM).

			r_{xy}			KT			MAE			ARCHITECTURE			BATCH SIZE			L2-REG.			OPTIMIZER			LEARNING RATE			LOSS			EPOCHS		
			NN	MVNN	MVNN	NN	MVNN	MVNN	NN	MVNN	MVNN	NN	MVNN	MVNN	NN	MVNN	MVNN	NN	MVNN	MVNN	NN	MVNN	MVNN	NN	MVNN	MVNN	NN	MVNN	MVNN	NN	MVNN	MVNN
DOMAIN	T	BIDDER	PLAIN	ABS	RELU-PROJ.	PLAIN	ABS	RELU-PROJ.	PLAIN	ABS	RELU-PROJ.	PLAIN	ABS	RELU-PROJ.	PLAIN	ABS	RELU-PROJ.	PLAIN	ABS	RELU-PROJ.	PLAIN	ABS	RELU	PLAIN	ABS	RELU-PROJ.	PLAIN	ABS	RELU-PROJ.	PLAIN	ABS	RELU-PROJ.
GSVM	10	NAT	0.782 ±0.020	0.861 ±0.017	0.856 ±0.022	0.583 ±0.021	0.672 ±0.022	0.668 ±0.027	2.197e+1 ±1.713e+0	1.731e+1 ±1.605e+0	1.776e+1 ±1.960e+0	[16, 16]	[62]	[63]	1	4	1	4.844e−10	1.160e−7	1.002e−10	ADAM	ADAM	ADAM	3.558e−3	9.945e−3	4.297e−3	MAE	MAE	MAE	219	55	60
		REG	0.746 ±0.033	0.799 ±0.032	0.816 ±0.034	0.557 ±0.033	0.614 ±0.034	0.633 ±0.038	1.376e+1 ±9.002e−1	1.268e+1 ±1.077e+0	1.185e+1 ±1.095e+0	[5, 5, 5]	[11]	[61]	2	2	1	3.817e−7	1.124e−7	5.884e−8	ADAM	ADAM	ADAM	9.973e−3	6.599e−3	6.587e−3	MAE	MAE	MAE	321	181	114
	20	NAT	0.911 ±0.017	0.966 ±0.008	0.965 ±0.007	0.752 ±0.029	0.865 ±0.020	0.849 ±0.017	1.347e+1 ±1.616e+0	8.479e+0 ±1.354e+0	8.568e+0 ±1.089e+0	[44]	[21, 21, 21]	[53]	3	1	3	3.864e−9	8.866e−8	1.344e−7	SGD	ADAM	ADAM	2.935e−3	3.584e−3	5.982e−3	MAE	MSE	MAE	398	338	117
		REG	0.944 ±0.011	0.965 ±0.009	0.973 ±0.009	0.815 ±0.021	0.878 ±0.018	0.882 ±0.020	6.594e+0 ±6.773e−1	4.844e+0 ±7.184e−1	4.324e+0 ±6.252e−1	[57]	[21, 21, 21]	[61]	4	1	2	2.300e−10	1.137e−8	4.712e−7	SGD	ADAM	ADAM	2.945e−3	1.415e−3	6.956e−3	MAE	MAE	MAE	390	381	138
	50	NAT	0.995 ±0.000	0.998 ±0.001	0.997 ±0.000	0.953 ±0.003	0.978 ±0.003	0.962 ±0.003	3.294e+0 ±2.693e−1	1.540e+0 ±1.774e−1	2.506e+0 ±2.261e−1	[61]	[20, 20, 20]	[61]	1	1	2	4.465e−8	2.117e−7	1.653e−7	SGD	ADAM	ADAM	8.275e−3	7.401e−4	2.246e−3	MAE	MAE	MAE	223	247	223
		REG	0.995 ±0.001	0.998 ±0.001	0.999 ±0.000	0.953 ±0.003	0.978 ±0.002	0.974 ±0.002	1.936e+0 ±1.204e−1	7.172e−1 ±9.339e−2	8.704e−1 ±8.185e−2	[48]	[21, 21, 21]	[62]	4	3	2	3.013e−7	5.333e−7	3.351e−7	SGD	ADAM	ADAM	6.250e−3	1.676e−3	2.002e−3	MAE	MAE	MAE	348	392	355
LSVM	10	NAT	0.701 ±0.013	0.803 ±0.010	0.598 ±0.019	0.710 ±0.023	0.712 ±0.013	0.693 ±0.011	3.225e+1 ±1.366e+0	2.811e+1 ±1.447e+0	3.263e+1 ±1.256e+0	[24]	[47]	[29, 29]	4	2	1	7.358e−8	1.224e−8	5.951e−8	ADAM	SGD	SGD	3.919e−3	9.634e−3	2.398e−4	MAE	MSE	MAE	213	372	321
		REG	0.679 ±0.027	0.766 ±0.026	0.785 ±0.028	0.504 ±0.025	0.585 ±0.028	0.605 ±0.031	2.917e+1 ±4.131e+0	2.457e+1 ±3.299e+0	2.385e+1 ±3.382e+0	[59]	[54]	[44]	1	4	1	6.406e−7	2.702e−7	1.287e−8	ADAM	ADAM	SGD	9.853e−3	9.752e−3	1.323e−3	MAE	MAE	MAE	113	121	273
	50	NAT	0.681 ±0.024	0.793 ±0.011	0.796 ±0.012	0.678 ±0.035	0.765 ±0.011	0.753 ±0.009	3.052e+1 ±1.116e+0	2.327e+1 ±7.769e−1	2.292e+1 ±6.508e−1	[4]	[11, 11, 11]	[20, 20, 20]	1	1	1	5.092e−7	6.468e−8	2.702e−10	ADAM	ADAM	SGD	8.593e−3	1.084e−3	3.355e−3	MAE	MAE	MAE	92	306	220
		REG	0.938 ±0.006	0.964 ±0.006	0.963 ±0.007	0.812 ±0.013	0.859 ±0.015	0.860 ±0.017	1.274e+1 ±2.013e+0	9.379e+0 ±2.020e+0	9.511e+0 ±2.114e+0	[26]	[24, 24]	[62]	1	2	1	1.997e−10	4.041e−7	6.958e−7	SGD	ADAM	ADAM	1.067e−3	1.994e−3	1.917e−3	MAE	MAE	MAE	158	281	258
	100	NAT	0.719 ±0.015	0.825 ±0.008	0.834 ±0.008	0.706 ±0.018	0.817 ±0.005	0.813 ±0.005	2.805e+1 ±7.422e−1	1.994e+1 ±6.458e−1	1.998e+1 ±6.605e−1	[9, 9, 9]	[32, 32]	[24, 24]	4	3	1	9.546e−10	7.747e−7	8.972e−10	ADAM	ADAM	ADAM	5.013e−3	1.780e−3	2.406e−4	MAE	MAE	MAE	79	250	391
		REG	0.969 ±0.005	0.983 ±0.004	0.983 ±0.005	0.857 ±0.012	0.912 ±0.014	0.918 ±0.015	9.082e+0 ±1.764e+0	6.003e+0 ±1.649e+0	5.952e+0 ±1.719e+0	[14]	[22, 22]	[55]	4	2	4	6.641e−10	7.392e−9	2.273e−9	SGD	ADAM	ADAM	2.109e−3	1.450e−3	3.640e−3	MAE	MAE	MAE	392	387	204
MRVM	10	LO	0.296 ±0.021	0.438 ±0.018	0.383 ±0.023	0.200 ±0.015	0.303 ±0.012	0.262 ±0.017	7.447e+6 ±1.383e+6	7.093e+6 ±1.330e+6	7.511e+6 ±1.445e+6	[4]	[9, 9]	[21]	2	1	4	3.510e−10	1.161e−7	4.206e−8	SGD	ADAM	ADAM	1.222e−4	5.775e−4	2.028e−3	MSE	MSE	MSE	377	276	271
		NAT	0.597 ±0.009	0.588 ±0.016	0.529 ±0.022	0.414 ±0.008	0.409 ±0.014	0.365 ±0.018	3.708e+8 ±1.415e+8	2.754e+8 ±3.316e+7	2.863e+8 ±2.287e+7	[2, 2]	[29]	[32]	4	3	1	4.675e−8	1.400e−7	3.966e−8	ADAM	ADAM	ADAM	1.681e−3	1.320e−3	2.529e−4	MSE	MAE	MAE	330	121	352
		REG	0.368 ±0.052	0.482 ±0.028	0.459 ±0.031	0.255 ±0.038	0.340 ±0.020	0.322 ±0.022	2.809e+7 ±6.838e+6	2.455e+7 ±6.173e+6	2.599e+7 ±6.790e+6	[6]	[13]	[28]	2	1	2	6.042e−9	4.163e−8	1.333e−7	SGD	ADAM	ADAM	1.360e−4	7.934e−4	2.344e−3	MSE	MSE	MAE	86	285	184
	100	LO	0.729 ±0.012	0.928 ±0.010	0.918 ±0.012	0.545 ±0.012	0.793 ±0.017	0.786 ±0.019	5.177e+6 ±9.344e+5	2.582e+6 ±3.289e+5	2.735e+6 ±3.360e+5	[1, 1]	[32]	[13, 13]	1	1	1	1.515e−8	6.048e−8	3.342e−7	ADAM	ADAM	ADAM	1.405e−3	9.018e−4	4.855e−3	MAE	MAE	MSE	225	157	61
		NAT	0.776 ±0.010	0.894 ±0.012	0.887 ±0.011	0.581 ±0.010	0.737 ±0.015	0.726 ±0.014	2.094e+8 ±1.582e+7	1.338e+8 ±1.393e+7	1.390e+8 ±1.339e+7	[6, 6]	[13, 13]	[23]	2	2	4	6.091e−7	1.455e−8	2.330e−8	ADAM	ADAM	ADAM	2.654e−4	2.368e−4	5.983e−4	MAE	MAE	MAE	393	399	245
		REG	0.747 ±0.016	0.915 ±0.013	0.917 ±0.015	0.572 ±0.018	0.777 ±0.023	0.779 ±0.027	1.880e+7 ±3.774e+6	8.807e+6 ±1.401e+6	8.504e+6 ±1.121e+6	[1, 1]	[15, 15]	[13, 13]	1	1	1	5.212e−7	5.562e−8	1.336e−8	ADAM	ADAM	ADAM	2.176e−3	3.952e−4	9.247e−4	MAE	MAE	MAE	117	130	189
	300	LO	0.934 ±0.005	0.968 ±0.004	0.972 ±0.003	0.819 ±0.009	0.874 ±0.011	0.883 ±0.010	2.417e+6 ±4.241e+5	1.616e+6 ±3.194e+5	1.484e+6 ±2.932e+5	[11, 11]	[30]	[16, 16]	4	4	2	7.798e−9	2.857e−8	3.990e−8	SGD	ADAM	ADAM	2.904e−3	1.571e−3	1.195e−3	MAE	MSE	MAE	93	228	161
		NAT	0.927 ±0.014	0.933 ±0.008	0.933 ±0.008	0.808 ±0.013	0.818 ±0.008	0.814 ±0.009	9.967e+7 ±1.006e+7	9.519e+7 ±8.113e+6	9.584e+7 ±8.548e+6	[1, 1]	[4, 4]	[9]	3	2	4	5.880e−9	3.681e−9	6.226e−9	SGD	ADAM	ADAM	3.670e−4	1.346e−3	6.650e−4	MAE	MAE	MAE	292	149	191
		REG	0.923 ±0.005	0.963 ±0.007	0.958 ±0.009	0.809 ±0.012	0.862 ±0.018	0.851 ±0.019	9.405e+6 ±1.883e+6	5.257e+6 ±6.323e+5	5.501e+6 ±5.956e+5	[16]	[32]	[9, 9]	1	1	3	1.246e−10	7.778e−7	8.111e−8	SGD	ADAM	ADAM	4.827e−3	5.867e−4	1.064e−3	MAE	MAE	MAE	103	292	298
SRVM	10	H.F.	0.762 ±0.011	0.805 ±0.010	0.795 ±0.018	0.607 ±0.012	0.642 ±0.014	0.626 ±0.020	4.268e+1 ±2.940e+0	1.359e+1 ±9.009e−1	1.436e+1 ±9.693e−1	[2, 2, 2]	[29]	[30, 30]	4	2	2	5.455e−9	1.884e−7	3.639e−7	ADAM	ADAM	ADAM	3.995e−3	9.079e−3	9.808e−3	MSE	MAE	MSE	394	206	253
		LO	0.570 ±0.030	0.678 ±0.027	0.660 ±0.029	0.489 ±0.032	0.571 ±0.027	0.559 ±0.030	3.298e+0 ±3.677e−1	2.708e+0 ±2.180e−1	2.756e+0 ±2.047e−1	[18]	[18, 18, 18]	[27, 27]	1	2	3	2.506e−9	1.564e−7	9.703e−7	SGD	ADAM	ADAM	1.369e−4	3.551e−3	5.858e−3	MSE	MAE	MAE	89	203	290
		NAT	0.636 ±0.018	0.692 ±0.028	0.676 ±0.032	0.535 ±0.012	0.579 ±0.023	0.560 ±0.026	4.082e+2 ±2.859e+1	3.582e+2 ±2.826e+1	3.691e+2 ±2.935e+1	[2, 2, 2]	[25, 25]	[55]	3	2	1	5.466e−9	1.471e−7	1.295e−10	ADAM	ADAM	ADAM	4.268e−3	7.093e−3	5.177e−3	MAE	MAE	MAE	400	59	135
		REG	0.675 ±0.020	0.704 ±0.029	0.603 ±0.027	0.544 ±0.014	0.572 ±0.024	0.562 ±0.023	2.234e+2 ±1.753e+1	1.994e+2 ±1.590e+1	2.042e+2 ±1.624e+1	[10, 10, 10]	[35]	[44]	2	2	1	3.237e−9	2.667e−8	6.439e−9	ADAM	ADAM	ADAM	9.590e−3	3.521e−3	3.023e−3	MSE	MAE	MAE	320	219	396
	50	H.F.	0.895 ±0.012	0.945 ±0.006	0.933 ±0.007	0.803 ±0.020	0.870 ±0.011	0.853 ±0.013	9.761e+0 ±1.216e+0	5.789e+0 ±5.648e−1	6.451e+0 ±6.152e−1	[2, 2]	[21, 21]	[21, 21, 21]	1	4	2	3.989e−8	8.670e−9	8.080e−9	SGD	ADAM	ADAM	7.067e−4	1.847e−3	2.820e−3	MAE	MAE	MAE	256	211	204
		LO	0.794 ±0.019	0.992 ±0.002	0.955 ±0.008	0.771 ±0.030	0.903 ±0.000	0.902 ±0.000	2.128e+0 ±2.477e−1	2.976e−1 ±4.422e−2	5.608e−1 ±8.987e−2	[2]	[20, 20, 20]	[20, 20, 20]	3	4	4	8.099e−7	7.348e−8	5.486e−8	SGD	ADAM	ADAM	1.196e−4	2.398e−3	3.818e−3	MAE	MAE	MAE	224	245	333
		NAT	0.916 ±0.008	0.996 ±0.001	0.995 ±0.002	0.801 ±0.009	0.925 ±0.007	0.918 ±0.005	2.014e+2 ±1.438e+1	3.280e+1 ±5.011e+0	3.528e+1 ±4.707e+0	[2]	[19, 19, 19]	[19, 19, 19]	1	2	2	9.253e−7	6.578e−8	4.919e−7	SGD	ADAM	ADAM	2.687e−4	2.340e−3	4.937e−3	MAE	MAE	MAE	112	220	225
		REG	0.945 ±0.017	0.995 ±0.001	0.995 ±0.001	0.823 ±0.022	0.939 ±0.006	0.931 ±0.004	9.217e+1 ±1.343e+1	2.119e+1 ±2.486e+0	2.206e+1 ±2.423e+0	[2, 2, 2]	[30, 30]	[19, 19, 19]	1	1	3	5.244e−10	2.352e−9	6.164e−8	ADAM	ADAM	ADAM	6.539e−3	1.920e−3	4.493e−3	MAE	MAE	MAE	230	208	297
	100	H.F.	0.927 ±0.005	0.964 ±0.003	0.956 ±0.004	0.896 ±0.006	0.917 ±0.005	0.908 ±0.006	6.204e+0 ±3.830e−1	3.867e+0 ±2.865e−1	4.337e+0 ±3.420e−1	[1]	[32, 32]	[20, 20, 20]	4	4	1	8.548e−8	2.419e−8	3.730e−9	SGD	ADAM	ADAM	5.262e−3	1.822e−3	7.410e−4	MAE	MAE	MAE	290	86	186
		LO	0.872 ±0.003	0.998 ±0.001	0.978 ±0.006	0.900 ±0.002	0.903 ±0.000	0.903 ±0.000	1.313e+0 ±1.255e−1	1.173e−1 ±1.749e−2	2.839e−1 ±4.946e−2	[1]	[21, 21, 21]	[20, 20, 20]	4	4	3	1.034e−9	6.892e−8	2.486e−8	SGD	ADAM	ADAM	5.180e−4	1.882e−3	1.684e−3	MAE	MAE	MAE	376	125	372
		NAT	0.959 ±0.004	0.999 ±0.000	0.999 ±0.000	0.841 ±0.008	0.958 ±0.002	0.952 ±0.003	1.293e+2 ±1.099e+1	1.342e+1 ±1.168e+0	1.502e+1 ±1.181e+0	[34]	[14, 14, 14]	[20, 20, 20]	1	2	4	6.302e−8	1.520e−7	3.698e−7	ADAM	ADAM	ADAM	5.768e−3	2.262e−3	4.292e−3	MAE	MAE	MAE	221	289	244
		REG	0.975 ±0.002	0.998 ±0.000	0.981 ±0.033	0.895 ±0.012	0.962 ±0.003	0.948 ±0.021	5.510e+1 ±4.958e+0	1.206e+1 ±1.335e+0	1.910e+1 ±1.299e+1	[5]	[23, 23]	[21, 21, 21]	1	3	2	1.617e−8	2.641e−9	3.085e−8	SGD	ADAM	ADAM	2.324e−4	1.790e−3	3.954e−3	MAE	MAE	MAE	330	381	76

Table 7: Prediction performance of the MVNN implementations MVNN-ABS and MVNN-RELU-PROJECTED measured via Pearson correlation coefficient (r_{xy}), Kendall tau (**KT**) and mean absolute error (**MAE**) with a 95%-CI in four SATS domains with corresponding bidder types: high frequency (H.F.), local (LO), regional (REG) and national (NAT). Both MVNNs and plain NNs are trained on T, validated on $0.2 \cdot 262,144$ and evaluated on $262,144 - T - 0.2 \cdot 262,144$ random bundles.

REGIONAL BIDDER	NATIONAL BIDDER	LOCAL BIDDER	HIGH FREQUENCY BIDDER

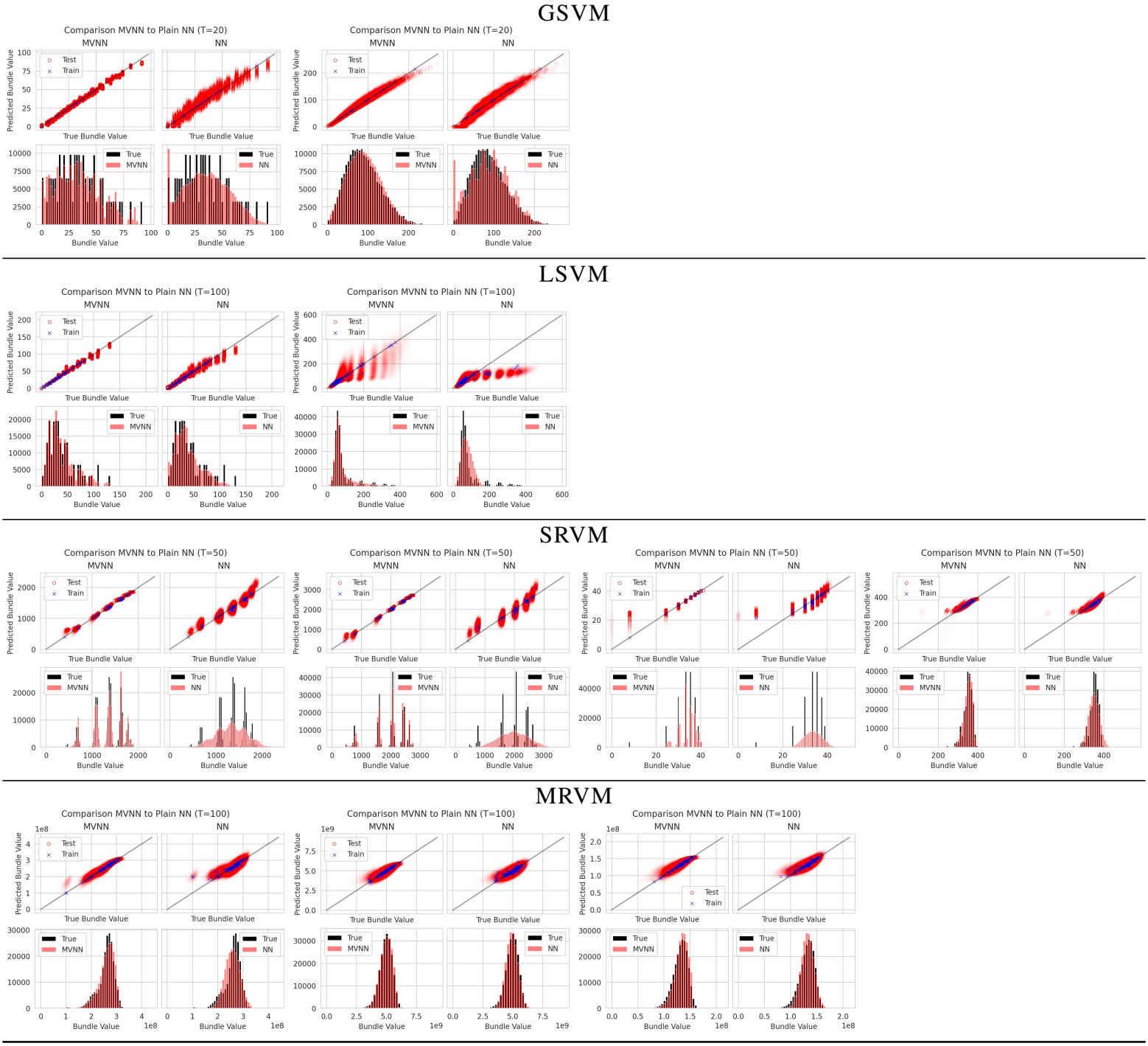


Table 8: Prediction performance comparison for selected MVNNs (MVNN-RELU-PROJECTED) and plain NNs from Table 1 across all domains and bidder types. The top plots compare the quality of the prediction (training data as blue crosses), while the bottom plots focus on how well the models capture the overall value distribution.

6 Bayesian Optimization-based Combinatorial Assignment

The content of this chapter has previously appeared in

Bayesian Optimization-based Combinatorial Assignment.
Jakob Weissteiner*, Jakob Heiss*, Julien Siems* and Sven Seuken.
In *Proceedings of the Thirty-seventh AAAI Conference on Artificial Intelligence* (AAAI'23), Washington, D.C., USA, February 2023.

For its full updated version including appendix, please see

Bayesian Optimization-based Combinatorial Assignment.
Jakob Weissteiner*, Jakob Heiss*, Julien Siems* and Sven Seuken.
Working paper, March 2023.
URL: arxiv.org/pdf/2208.14698.pdf

*These authors contributed equally.

Bayesian Optimization-Based Combinatorial Assignment*

Jakob Weissteiner[1,3†], **Jakob Heiss**[2,3†], **Julien Siems**[1†] and **Sven Seuken**[1,3]

[1]University of Zurich
[2]ETH Zurich
[3]ETH AI Center
weissteiner@ifi.uzh.ch, jakob.heiss@math.ethz.ch, juliensiems@gmail.com, seuken@ifi.uzh.ch

Abstract

We study the combinatorial assignment domain, which includes combinatorial auctions and course allocation. The main challenge in this domain is that the bundle space grows exponentially in the number of items. To address this, several papers have recently proposed *machine learning-based preference elicitation* algorithms that aim to elicit only the most important information from agents. However, the main shortcoming of this prior work is that it does not model a mechanism's uncertainty over values for not yet elicited bundles. In this paper, we address this shortcoming by presenting a *Bayesian optimization-based combinatorial assignment (BOCA)* mechanism. Our key technical contribution is to integrate a method for capturing model uncertainty into an iterative combinatorial auction mechanism. Concretely, we design a new method for estimating an *upper uncertainty bound* that can be used to define an acquisition function to determine the next query to the agents. This enables the mechanism to properly *explore* (and not just *exploit*) the bundle space during its preference elicitation phase. We run computational experiments in several spectrum auction domains to evaluate BOCA's performance. Our results show that BOCA achieves higher allocative efficiency than state-of-the-art approaches.

1 Introduction

Many economic problems require finding an efficient combinatorial assignment of multiple indivisible items to multiple agents. Popular examples include *combinatorial auctions (CAs)*, *combinatorial exchanges (CEs)*, and *combinatorial course allocation*. In CAs, heterogeneous items are allocated among a set of bidders, e.g., for the sale of spectrum licenses (Cramton 2013). In CEs, items are allocated among agents who can be sellers *and* buyers at the same time, e.g., for the reallocation of catch shares (Bichler, Fux, and Goeree 2019). In course allocation, course seats are allocated among students at universities (Budish 2011).

In all these domains, agents have preferences over *bundles* of items. In particular, agents' preferences may exhibit *complementarities* and *substitutabilities* among items. A mechanism that allows agents to report values for bundles rather than just for individual items can achieve significantly higher efficiency. However, this also implies that agents' preferences are exponentially-sized (i.e., for m items there are 2^m different bundles), which implies that agents cannot report values for all bundles. Therefore, the key challenge in combinatorial assignment is the design of a *preference elicitation (PE)* algorithm that is (i) *practically feasible* w.r.t. elicitation costs and (ii) *smart*, i.e., it should elicit the information that is "most useful" for achieving high efficiency.

1.1 Iterative Combinatorial Auctions (ICAs)

While the PE challenge is common to all combinatorial assignment problems, it has been studied most intensely in the CA domain (Sandholm and Boutilier 2006). In CAs with general valuations, the amount of communication needed to guarantee full efficiency is exponential in the number of items (Nisan and Segal 2006). Thus, practical CAs cannot provide efficiency guarantees. In practice, *iterative combinatorial auctions (ICAs)* are therefore employed, where the auctioneer interacts with bidders over rounds, eliciting a *limited* (and thus *practically feasible*) amount of information, aiming to find a highly efficient allocation. ICAs are widely used; e.g., for the sale of licenses to build offshore wind farms (Ausubel and Cramton 2011). The provision of spectrum licenses via the *combinatorial clock auction (CCA) (Ausubel, Cramton, and Milgrom 2006)* has generated more than $20 billion in total revenue (Ausubel and Baranov 2017). Thus, increasing the efficiency of such real-world ICAs by only 1% point translates into monetary gains of hundreds of millions of dollars.

1.2 ML-Powered Preference Elicitation

In recent years, researchers have used machine learning (ML) to design smart PE algorithms. Most related to this paper is the work by Brero, Lubin, and Seuken (2018, 2021), who developed the first practical ML-powered ICA that outperforms the CCA. The main idea of their mechanism is two-fold: first, they train a separate *support vector regression* model to learn each bidder's full value function from a small set of bids; second, they solve an *ML-based winner determination problem (WDP)* to determine the allocation with the highest predicted social welfare, and they use this allocation to generate the next set of queries to all bidders.

*This paper is the full version of Weissteiner et al. (2023) published at AAAI'23 including the appendix.

†These authors contributed equally.

This process repeats in an iterative fashion until a fixed number of queries has been asked. Thus, their ML-powered ICA can be interpreted as a form of combinatorial *Bayesian optimization (BO)* (see Appendix C).

In several follow-up papers, this work has been extended by developing more sophisticated ML methods for this problem. Weissteiner and Seuken (2020) integrated *neural networks (NN)* in their ICA and further increased efficiency. Weissteiner et al. (2022b) used *Fourier transforms (FTs)* to leverage different notions of sparsity of value functions. Finally, Weissteiner et al. (2022a) achieved state-of-the-art (SOTA) performance using *monotone-value NNs (MVNNs)*, which incorporate important prior domain knowledge.

The main shortcoming of this prior work is that all of these approaches are *myopic* in the sense that the resulting mechanisms simply query the allocation with the highest predicted welfare. In particular, the mechanisms do not have any model of *uncertainty* over bidders' values for not yet elicited bundles, although handling uncertainty in a principled manner is one of the key requirements of a smart PE algorithm (Bonilla, Guo, and Sanner 2010). Thus, the mechanisms cannot properly control the *exploration-exploitation trade-off* inherent to BO. For ML-based ICAs, this means that the mechanisms may get stuck in local minima, repeatedly querying one part of the allocation space while not exploring other, potentially more efficient allocations.

1.3 Our Contributions

In this paper, we address this main shortcoming of prior work and show how to integrate a notion of *model uncertainty* (i.e., epistemic uncertainty) over agents' preferences into iterative combinatorial assignment. Concretely, we design a *Bayesian optimization-based combinatorial assignment (BOCA)*[1] mechanism that makes use of model uncertainty in its query generation module. The main technical challenge is to design a new method for estimating an *upper uncertainty bound* that can be used to define an acquisition function to determine the next query. For this we combine *MVNNs* (Weissteiner et al. 2022a) with *neural optimization-based model uncertainty (NOMU)* (Heiss et al. 2022), a recently introduced method to estimate model uncertainty for NNs. In detail, we make the following contributions:

1. We present a modified NOMU algorithm (Section 3.1), tailored to CAs, exploiting monotonicity of agents' preferences and the discrete (finite) nature of this setting.
2. We show that generic parameter initialization for monotone NNs can dramatically fail and propose a new initialization method for MVNNs based on uniform mixture distributions (Section 3.2).
3. We present a more succinct mixed integer linear program for MVNNs to solve the ML-based WDP (Section 3.3).
4. We experimentally show that BOCA outperforms prior approaches in terms of efficiency (Section 4).

Although our contribution applies to any combinatorial assignment setting, we focus on CAs to simplify the notation and because there exist well-studied preference generators for CAs that we use for our experiments.

Our source code is publicly available on GitHub via: https://github.com/marketdesignresearch/BOCA.

[1]The acronym BOCA has also been used for a different method, namely for *Bayesian optimisation with continuous approximations* by Kandasamy et al. (2017).

1.4 Related Work on Model Uncertainty

Estimating model uncertainty for NNs is an active area of research in AI and ML, with a plethora of new methods proposed every year. Classic methods can be broadly categorized into *(i) ensemble methods*: training multiple different NNs to estimate model uncertainty (Gal and Ghahramani 2016; Lakshminarayanan, Pritzel, and Blundell 2017; Wenzel et al. 2020) and *(ii) Bayesian NNs (BNNs)*: assuming a prior distribution over parameters and then estimating model uncertainty by approximating the intractable posterior (Graves 2011; Blundell et al. 2015; Hernández-Lobato and Adams 2015; Ober and Rasmussen 2019). However, for ML-based iterative combinatorial assignment, a key requirement is to be able to efficiently solve the ML-based WDP based on these uncertainty estimates. As there is no known computationally tractable method to perform combinatorial optimization over ensembles or BNNs, we cannot use these approaches for ML-based ICAs. In contrast, NOMU (Heiss et al. 2022) enables the computationally efficient optimization over its uncertainty predictions, which is why we use it as a building block for BOCA.

2 Preliminaries

In this section, we present our formal model (Section 2.1), review the ML-based ICA by Brero, Lubin, and Seuken (2021) (Section 2.2), briefly review Bayesian optimization (BO) (Section 2.3), and review *monotone-value neural networks (MVNNs)* by Weissteiner et al. (2022a) (Section 2.4) as well as *neural optimization-based model uncertainty (NOMU)* by Heiss et al. (2022) (Section 2.5).

2.1 Formal Model for ICAs

We consider a CA with n bidders and m indivisible items. Let $N = \{1, \dots, n\}$ and $M = \{1, \dots, m\}$ denote the set of bidders and items. We denote by $x \in \mathcal{X} = \{0,1\}^m$ a bundle of items represented as an indicator vector, where $x_j = 1$ iff item $j \in M$ is contained in x. Bidders' true preferences over bundles are represented by their (private) value functions $v_i : \mathcal{X} \to \mathbb{R}_+,\ i \in N$, i.e., $v_i(x)$ represents bidder i's true value for bundle $x \in \mathcal{X}$.

By $a = (a_1, \dots, a_n) \in \mathcal{X}^n$ we denote an allocation of bundles to bidders, where a_i is the bundle bidder i obtains. We denote the set of *feasible* allocations by $\mathcal{F} = \left\{a \in \mathcal{X}^n : \sum_{i \in N} a_{ij} \leq 1,\ \forall j \in M\right\}$. We let $p \in \mathbb{R}^n_+$ denote the bidders' payments. We assume that bidders have quasilinear utility functions u_i of the form $u_i(a, p) = v_i(a_i) - p_i$. This implies that the (true) *social welfare* $V(a)$ of an allocation a is equal to the sum of all bidders' values $\sum_{i \in N} v_i(a_i)$. We let $a^* \in \operatorname{argmax}_{a \in \mathcal{F}} V(a)$ denote a social-welfare maximizing, i.e., *efficient*, allocation. The *efficiency* of any allocation $a \in \mathcal{F}$ is determined as $V(a)/V(a^*)$.

An ICA *mechanism* defines how the bidders interact with the auctioneer and how the allocation and payments are determined. We denote a bidder's (possibly untruthful) reported value function by $\hat{v}_i : \mathcal{X} \to \mathbb{R}_+$. In this paper, we consider ICAs that ask bidders to iteratively report their values $\hat{v}_i(x)$ for bundles x selected by the mechanism. A finite set of reported bundle-value pairs of bidder i is denoted as $R_i = \left\{\left(x^{(l)}, \hat{v}_i(x^{(l)})\right)\right\}$, $x^{(l)} \in \mathcal{X}$. Let $R = (R_1, \ldots, R_n)$ be the tuple of reported bundle-value pairs obtained from all bidders. We define the *reported social welfare* of an allocation a given R as $\widehat{V}(a|R) := \sum_{i \in N:\, (a_i, \hat{v}_i(a_i)) \in R_i} \hat{v}_i(a_i)$, where $(a_i, \hat{v}_i(a_i)) \in R_i$ ensures that only values for reported bundles contribute. The ICA's optimal allocation $a^*_R \in \mathcal{F}$ and payments $p(R) \in \mathbb{R}^n_+$ are computed based on the elicited reports R *only*. More formally, $a^*_R \in \mathcal{F}$ given reports R is defined as

$$a^*_R \in \operatorname*{argmax}_{a \in \mathcal{F}} \widehat{V}(a|R). \tag{1}$$

As the auctioneer can only query a limited number of bundles $|R_i| \leq Q^{\max}$ (e.g., $Q^{\max} = 100$), an ICA needs a practically feasible and smart PE algorithm.

2.2 A Machine Learning-Powered ICA

We now provide a high-level review of the *machine learning-powered combinatorial auction (MLCA)* by Brero, Lubin, and Seuken (2021) (please see Appendix A for further details). MLCA proceeds in rounds until a maximum number of value queries per bidder $Q^{\max}$ is reached. In each round, for every bidder i, an ML model $\mathcal{A}_i$ is trained on the bidder's reports R_i to learn an approximation of bidders' value functions. Next, MLCA generates new value queries by computing the allocation with the highest predicted social welfare. Concretely, it computes $q^{\text{new}} = (q_i^{\text{new}})_{i=1}^n$ with $q_i^{\text{new}} \in \mathcal{X} \setminus R_i$ by solving an ML-based WDP:

$$q^{\text{new}} \in \operatorname*{argmax}_{a \in \mathcal{F}} \sum_{i \in N} \mathcal{A}_i(a_i) \tag{2}$$

The idea is the following: if the $\mathcal{A}_i$'s are good surrogate models of the bidders' value functions, then the efficiency of q^{new} is likely to be high as well. Thus, in each round, bidders are providing value reports on bundles that are guaranteed to fit into a feasible allocation and that together are predicted to have high social welfare. Additionally, bidders are also allowed to submit "push-bids," enabling them to submit information to the auctioneer that they deem useful, even if they are not explicitly queried about it. At the end of each round, MLCA receives reports R^{new} from all bidders for the newly generated queries q^{new}, and updates the overall elicited reports R. When $Q^{\max}$ is reached, MLCA computes an allocation a^*_R that maximizes the *reported* social welfare (Equation (1)) and determines VCG payments $p(R)$ based on all reports (see Appendix Definition B.1).

Remark 1 (IR, No-Deficit, and Incentives of MLCA)**.** *Brero, Lubin, and Seuken (2021) showed that MLCA satisfies* individual rationality (IR) *and* no-deficit*, with* any *ML algorithm. They also studied MLCA's incentive properties; this is important, since manipulations may lower efficiency. Like all deployed ICAs (including the CCA), MLCA is not strategyproof. However, they argued that it has good incentives in practice; and given two additional assumptions, bidding truthfully is an ex-post Nash equilibrium. We present a detailed summary of their incentive analysis in Appendix B.*

2.3 Bayesian Optimization Background

In this section, we briefly review Bayesian optimization (BO). BO refers to a class of *machine learning-based gradient-free* optimization methods, which, for a given black-box objective function $f : X \to \mathbb{R}$, aim to solve

$$\max_{x \in X} f(x) \tag{3}$$

in an *iterative* manner. Specifically, given a budget of T queries (i.e., function evaluations of f), a BO algorithm generates queries $\{x^{(1)}, \ldots, x^{(T)}\}$ with the aim that

$$\max_{x \in \{x^{(1)}, \ldots, x^{(T)}\}} f(x) \approx \max_{x \in X} f(x). \tag{4}$$

In each BO step t, the algorithm selects a new input point $x^{(t)} \in X$ and observes a (potentially noisy) output

$$y^{(t)} = f(x^{(t)}) + \varepsilon^{(t)}, \tag{5}$$

where $\varepsilon^{(t)}$ is typically assumed to be i.i.d. Gaussian, i.e., $\varepsilon^{(t)} \sim \mathcal{N}(0, \sigma^2)$.[2] The BO algorithm's decision rule for selecting the query $x^{(t)}$ is based on

1. A *probabilistic model* representing an (approximate) posterior distribution over f (e.g., Gaussian processes, NOMU, ensembles, BNNs, etc.).
2. An *acquisition function* $\mathcal{A} : X \to \mathbb{R}$ that uses this probabilistic model to determine the next query $x^{(t)} \in \operatorname{argmax}_{x \in X} \mathcal{A}(x)$ by properly trading off *exploration* and *exploitation*. See Appendix C.3 for popular examples of acquisition functions including:
   - *Upper uncertainty bound* (uUB) (aka *upper confidence bound (UCB)*) (Srinivas et al. 2012)
   - *Expected improvement* (Frazier 2018, Section 4.1)
   - *Thompson sampling* (Chapelle and Li 2011)

Remark 2. *MLCA (Section 2.2) can be seen as a combinatorial BO algorithm with acquisition function* $\mathcal{A}(a) := \sum_{i \in N} \mathcal{A}_i(a_i)$ *(see Appendix C for a discussion).*

2.4 MVNNs: Monotone-Value Neural Networks

MVNNs (Weissteiner et al. 2022a) are a new class of NNs specifically designed to represent *monotone combinatorial* valuations. First, we reprint the definition of MVNNs and then discuss their desirable properties.

Definition 1 (MVNN, Weissteiner et al. (2022a))**.** *An MVNN* $\mathcal{M}_i^\theta : \mathcal{X} \to \mathbb{R}_+$ *for bidder* $i \in N$ *is defined as*

$$\mathcal{M}_i^\theta(x) := W^{i,K_i} \varphi_{0,t^{i,K_i-1}} \left(\ldots \varphi_{0,t^{i,1}} (W^{i,1} x + b^{i,1}) \ldots\right) \tag{6}$$

- $K_i + 1 \in \mathbb{N}$ *is the number of layers* ($K_i - 1$ *hidden layers),*

[2] In this paper, we assume that $\sigma^2 = 0$.

- $\{\varphi_{0,t^{i,k}}\}_{k=1}^{K_i-1}$ *are the MVNN-specific activation functions with cutoff* $t^{i,k} > 0$*, called* bounded ReLU (bReLU)*:*

$$\varphi_{0,t^{i,k}}(\cdot) := \min(t^{i,k}, \max(0, \cdot)) \tag{7}$$

- $W^i := (W^{i,k})_{k=1}^{K_i}$ *with* $W^{i,k} \geq 0$ *and* $b^i := (b^{i,k})_{k=1}^{K_i-1}$ *with* $b^{i,k} \leq 0$ *are the* non-negative *weights and* non-positive *biases of dimensions* $d^{i,k} \times d^{i,k-1}$ *and* $d^{i,k}$*, whose parameters are stored in* $\theta = (W^i, b^i)$.

MVNNs are particularly well suited for the design of combinatorial assignment mechanism for two reasons. First, MVNNs are *universal* in the set of monotone and normalized value functions (Weissteiner et al. 2022a, Theorem 1), i.e., *any* $\hat{v}_i : \mathcal{X} \to \mathbb{R}_+$ that satisfies the following two properties can be represented *exactly* as an MVNN $\mathcal{M}_i^\theta$:

1. **Monotonicity (M)** (*"additional items increase value"*): For $A, B \in 2^M$: if $A \subseteq B$ it holds that $\hat{v}_i(A) \leq \hat{v}_i(B)$
2. **Normalization (N)** (*"no value for empty bundle"*): $\hat{v}_i(\emptyset) = \hat{v}_i((0, \dots, 0)) := 0$,

Second, Weissteiner et al. (2022a) showed that an MVNN-based WDP, i.e., $\underset{a \in \mathcal{F}}{\operatorname{argmax}} \sum_{i \in N} \mathcal{M}_i^\theta(a_i)$, can be succinctly encoded as a MILP, which is key for the design of MVNN-based iterative combinatorial assignment mechanisms. Finally, Weissteiner et al. (2022a) experimentally showed that using MVNNs as $\mathcal{A}_i$ in MLCA leads to SOTA performance.

2.5 NOMU

Recently, Heiss et al. (2022) introduced a novel method to estimate model uncertainty for NNs: *neural optimization-based model uncertainty (NOMU)*. In contrast to other methods (e.g., ensembles), NOMU represents an *upper uncertainty bound (uUB)* as a *single* and *MILP-formalizable* NN. Thus, NOMU is particularly well suited for iterative combinatorial assignment, where uUB-based *winner determination problems (WDPs)* need to be solved hundreds of times to generate new informative queries. This, together with NOMU's strong performance in noiseless BO, is the reason why we build on it and define a modified NOMU algorithm tailored to iterative combinatorial assignment (Section 3.1).

3 Bayesian Optimization-Based ICA

In this section, we describe the design of our Bayesian optimization-based combinatorial assignment (BOCA) mechanism. While the design is general, we here present it for the CA setting, leading to a BO-based ICA. Recall that MLCA generates new value queries by solving the ML-based WDP $q^{\text{new}} \in \underset{a \in \mathcal{F}}{\operatorname{argmax}} \sum_{i \in N} \mathcal{A}_i(a_i)$ (see Section 2.2). For the design of BOCA, we integrate a proper notion of uncertainty into MLCA by using a bidder-specific *upper uncertainty bound (uUB)*, taking the role of the ML model $\mathcal{A}_i$, to define our acquisition function $\mathcal{A}(a) := \sum_{i \in N} \mathcal{A}_i(a_i)$. To define our uUB and make it amenable to MLCA, we proceed in three steps: First, we combine MVNNs with a modified NOMU algorithm that is tailored to the characteristics of combinatorial assignment (Section 3.1). Second, we highlight the importance of proper parameter initialization for MVNNs and propose a more robust method (Section 3.2). Third, we present a more succinct MILP for MVNNs (Section 3.3). In the remainder of the paper, we make the following assumption:

Assumption 1. *For all agents* $i \in N$*, the true and reported value functions* v_i *and* $\hat{v}_i$ *fulfill the* ***Monotonicity (M)*** *and* ***Normalization (N)*** *property (see Section 2.4).*

3.1 Model Uncertainty for Monotone NNs

We propose a modified NOMU architecture and loss that is specifically tailored to combinatorial assignment. Concretely, our algorithm is based on the following two key characteristics of combinatorial assignment: (i) since agents' value functions are monotonically increasing, the uUBs need to be monotonically increasing too, and (ii) due to the (finite) discrete input space, one can derive a closed-form expression of the 100%-uUB as an MVNN. Before we present our modified NOMU architecture and loss, we introduce the MVNN-based 100%-uUB.

Let $\mathcal{H}$ denote a *hypothesis class* of functions $f : X \to \mathbb{R}$ for some input space X and let $\mathcal{H}_{D^{\text{train}}} := \{f \in \mathcal{H} : f(x^{(l)}) = y^{(l)}, l = 1, \dots, n^{\text{train}}\}$ denote the set of all functions from $\mathcal{H}$ that fit exactly through training points $D^{\text{train}} = \left\{\left(x^{(l)}, f(x^{(l)})\right)\right\}_{l=1}^{n^{\text{train}}}$.

Definition 2 (100%-uUB)**.** *For a hypothesis class* $\mathcal{H}$ *and a training set* D^{train}*, we define the 100%-uUB as* $f^{100\%\text{-}uUB}(x) := \sup_{f \in \mathcal{H}_{D^{train}}} f(x)$ *for every* $x \in X$.

In the following, let

$$\mathcal{V} := \{\hat{v} : \mathcal{X} \to \mathbb{R}_+ | \text{ satisfy } \textbf{(N)} \text{ and } \textbf{(M)}\} \tag{8}$$

denote the set of all value functions that satisfy the *normalization* and *monotonicity* property. Next, we define the 100%-uUB. In Theorem 1, we show that for $\mathcal{H} = \mathcal{V}$ the 100%-uUB can be explicitly represented as an MVNN.

Theorem 1 (MVNN-based 100%-uUB)**.** *Let* $((1, \dots, 1), \hat{v}_i(1, \dots, 1)) \in D^{train}$*. Then for* $\mathcal{H} = \mathcal{V}$ *it holds that* $f^{100\%\text{-}uUB}(x) = \max_{f \in \mathcal{V}_{D^{train}}} f(x)$ *for all* $x \in \mathcal{X}$ *and* $f^{100\%\text{-}uUB} \in \mathcal{V}_{D^{train}}$ *can be represented as a two hidden layer MVNN with* n^{train} *neurons per layer, which we denote as* $\mathcal{M}_i^{100\%\text{-}uUB}$ *going forward.*[3]

Proof. The proof for Theorem 1 is provided in Appendix D.1. It follows a similar idea as the universality proof in (Weissteiner et al. 2022a, Theorem 1). In particular, Equation (27) in Appendix D.1 provides the closed-form expression of $f^{100\%\text{-uUB}}$ as MVNN $\mathcal{M}_i^{100\%\text{-uUB}}$. □

Using the MVNN-based 100%-uUB $\mathcal{M}_i^{100\%\text{-uUB}}$, we can now define our modified NOMU architecture and loss.

The Architecture. Towards defining the architecture, we first observe that if the true function is monotonically increasing, the corresponding uUB needs to be monotonically increasing as well (Propositions 1 and 2 in Appendix D.2).

[3] Note that $\mathcal{M}_i^{100\%\text{-uUB}}(\cdot)$ depends on a training set D^{train}, but we omit this dependency in our notation to improve readability.

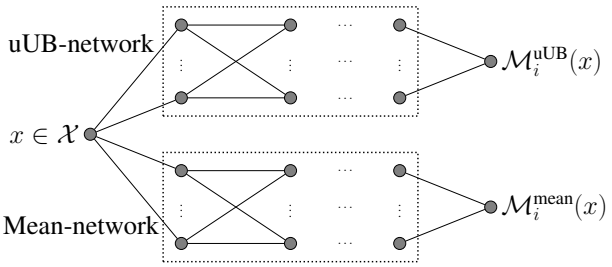


Figure 1: $\mathcal{M}_i^{\text{NOMU}}$: a modification of NOMU's original architecture for the combinatorial assignment domain.

Given that bidders' value functions are monotone (Assumption 1), this implies that our uUB must also be monotonically increasing. Thus, instead of the original NOMU architecture that outputs the (raw) uncertainty (i.e., an estimate of the posterior standard deviation) which is *not* monotone, we can modify NOMU's architecture and directly output the monotone uUB. Given this, we propose the following architecture $\mathcal{M}_i^{\text{NOMU}}$ to estimate the uUB for bidder $i \in N$. $\mathcal{M}_i^{\text{NOMU}}$ consists of two sub-MVNNs with two outputs: the mean prediction $\mathcal{M}_i^{\text{mean}} : \mathcal{X} \to \mathbb{R}$ and the estimated uUB $\mathcal{M}_i^{\text{uUB}} : \mathcal{X} \to \mathbb{R}$. In Figure 1, we provide a schematic representation of $\mathcal{M}_i^{\text{NOMU}}$ (see Appendix D.2 for details).

The Loss. Next, we formulate a new NOMU loss function L^π tailored to combinatorial assignment. Since we have a closed-form expression of the 100%-uUB as MVNN $\mathcal{M}_i^{\text{100\%-uUB}}$ (Theorem 1), we are able to enforce that $\mathcal{M}_i^{\text{mean}} \leq \mathcal{M}_i^{\text{uUB}} \leq \mathcal{M}_i^{\text{100\%-uUB}}$ via the design of our new loss function. Let $\mathcal{M}_i^{\text{mean}}$ be a trained mean-MVNN with a standard loss (e.g., MAE and L2-regularization). Using $\mathcal{M}_i^{\text{mean}}$ and $\mathcal{M}_i^{\text{100\%-uUB}}$, we then only train the parameters θ of $\mathcal{M}_i^{\text{uUB}}$ with loss L^π and L2-regularization parameter $\lambda > 0$, i.e., minimizing $L^\pi(\mathcal{M}_i^{\text{uUB}}) + \lambda \|\theta\|_2^2$ via gradient descent. In particular, the parameters of $\mathcal{M}_i^{\text{100\%-uUB}}$ and $\mathcal{M}_i^{\text{mean}}$ are not influenced by the training of $\mathcal{M}_i^{\text{uUB}}$ (see Appendix D.3 for details on the loss and training procedure).

Definition 3 (NOMU Loss Tailored to Combinatorial Assignment)**.** *Let $\pi = (\pi_{sqr}, \pi_{exp}, c_{exp}, \overline{\pi}, \underline{\pi}) \in \mathbb{R}_+^5$ be a tuple of hyperparameters and let $s(\mathcal{M}_i^{uUB}, x) := \min\{\mathcal{M}_i^{uUB}(x), \mathcal{M}_i^{100\%\text{-}uUB}(x)\} - \mathcal{M}_i^{mean}(x)$ for all $x \in \mathcal{X}$. For a training set D^{train}, L^π is defined as*

$$L^\pi(\mathcal{M}_i^{uUB}) := \pi_{sqr} \sum_{l=1}^{n^{train}} L_1^\beta \left(\mathcal{M}_i^{uUB}(x^{(l)}), y^{(l)}\right) \tag{9a}$$

$$+ \pi_{exp} \int_{[0,1]^m} g\left(-c_{exp} s(\mathcal{M}_i^{uUB}, x)\right) dx \tag{9b}$$

$$+ \pi_{exp} c_{exp} \overline{\pi} \int_{[0,1]^m} L_1^\beta \left((\mathcal{M}_i^{uUB}(x) - \mathcal{M}_i^{100\%\text{-}uUB}(x))^+\right) dx \tag{9c}$$

$$+ \pi_{exp} c_{exp} \underline{\pi} \int_{[0,1]^m} L_1^\beta \left((\mathcal{M}_i^{mean}(x) - \mathcal{M}_i^{uUB}(x))^+\right) dx\,, \tag{9d}$$

where L_1^β is the smooth L1-loss with threshold β (see Appendix Definition D.1), $(\cdot)^+$ the positive part, and $g := 1 + ELU$[4] is convex monotonically increasing with ELU being the exponential linear unit *(see Appendix Definition D.2).*

The interpretations of the four terms are as follows:

(9a) enforces that $\mathcal{M}_i^{\text{uUB}}$ fits through the training data.

(9b) pushes $\mathcal{M}_i^{\text{uUB}}$ up as long as it is below the 100%-uUB $\mathcal{M}_i^{\text{100\%-uUB}}$. This force gets weaker the further $\mathcal{M}_i^{\text{uUB}}$ is above the mean $\mathcal{M}_i^{\text{mean}}$ (especially if c_{exp} is large). π_{exp} controls the overall strength of (9b) and c_{exp} controls how fast this force increases when $\mathcal{M}_i^{\text{uUB}} \to \mathcal{M}_i^{\text{mean}}$. Thus, increasing π_{exp} increases the uUB and increasing c_{exp} increases the uUBs in regions where it is close to $\mathcal{M}_i^{\text{mean}}$. Weakening (9b) (i.e., $\pi_{\text{exp}} c_{\text{exp}} \to 0$) leads to $\mathcal{M}_i^{\text{uUB}} \approx \mathcal{M}_i^{\text{mean}}$. Strengthening (9b) by increasing $\pi_{\text{exp}} c_{\text{exp}}$ in relation to regularization[5] leads to $\mathcal{M}_i^{\text{uUB}} \approx \mathcal{M}_i^{\text{100\%-uUB}}$.

(9c) enforces that $\mathcal{M}_i^{\text{uUB}} \leq \mathcal{M}_i^{\text{100\%-uUB}}$. The strength of this term is determined by $\overline{\pi} \cdot (\pi_{\text{exp}} c_{\text{exp}})$, where $\overline{\pi}$ is the (9c)-specific hyperparameter and $\pi_{\text{exp}} c_{\text{exp}}$ adjusts the strength of (9c) to (9b).

(9d) enforces $\mathcal{M}_i^{\text{uUB}} \geq \mathcal{M}_i^{\text{mean}}$. The interpretation of $\underline{\pi}$ and $\pi_{\text{exp}} c_{\text{exp}}$ is analogous to (9c).

As in (Heiss et al. 2022), in the implementation of L^π, we approximate Equations (9b) to (9d) via Monte Carlo integration using additional, *artificial input points* $D^{\text{art}} := \{x^{(l)}\}_{l=1}^{n^{\text{art}}} \overset{i.i.d.}{\sim} \text{Unif}([0,1]^m)$.

Visualization of the uUB. In Figure 2, we present a visualization of the output of $\mathcal{M}_i^{\text{NOMU}}$ (i.e., $\mathcal{M}_i^{\text{mean}}$ and $\mathcal{M}_i^{\text{uUB}}$) and $\mathcal{M}_i^{\text{100\%-uUB}}$ for the national bidder in the LSVM domain of the spectrum auction test suite (SATS) (Weiss, Lubin, and Seuken 2017). In noiseless regression, uncertainty should vanish at observed training points, but (model) uncertainty should remain about value predictions for bundles that are very different from the bundles observed in training. Figure 2 shows that our uUB $\mathcal{M}_i^{\text{uUB}}$ nicely fulfills this. Moreover, we have shown in Appendix D.2 that $\mathcal{M}_i^{\text{uUB}}$ is monotonically increasing, since we assume that value functions fulfill the monotonicity property. This implies that once we observe a value for the full bundle, we obtain a globally bounded 100%-uUB, i.e., see $\mathcal{M}_i^{\text{100\%-uUB}}$ in Figure 2. Furthermore, we see that $\mathcal{M}_i^{\text{100\%-uUB}}$ jumps to a high value when only a single item is added to an already queried bundle, but then often stays constant (e.g., $|x| = 12, \ldots, 18$ in Figure 2). Thus, using such a 100%-uUB in our acquisition function, BOCA would only add a single item to an already queried bundle to have more items left for the other bidders instead of properly exploring the bundle space. Our uUB $\mathcal{M}_i^{\text{uUB}}$ circumvents this via implicit and explicit regularization and yields a useful uUB.

[4] In our notation, $g(\cdot)$ is the analog of the function $e^{(\cdot)}$ used in the original NOMU loss in (Heiss et al. 2022).

[5] Regularization can be early stopping or a small number of neurons (implicit) or L2-regularization on the parameters (explicit).

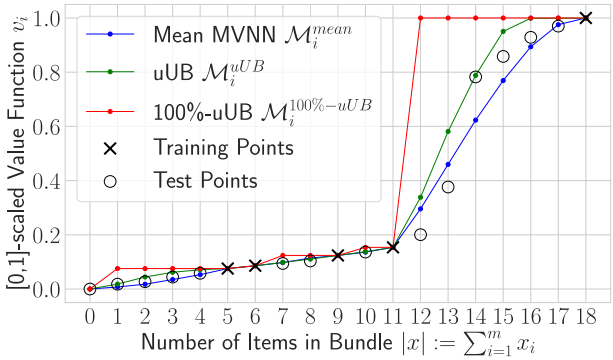


Figure 2: $\mathcal{M}_i^{\text{mean}}$, $\mathcal{M}_i^{\text{uUB}}$ and $\mathcal{M}_i^{\text{100\%-uUB}}$ along an increasing 1D subset-path (i.e., for all bundles $x^{(j)}, x^{(k)}$ on the x-axis it holds that for $j \leq k : x^{(j)} \subset x^{(k)}$).

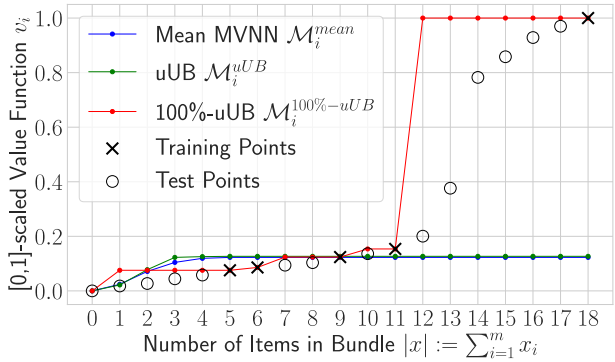


Figure 3: In contrast to our proposed initialization (see Figure 2), training fails with generic initialization already for relatively small [64,64]-architectures that were used here.

3.2 Parameter Initialization for MVNNs

We now discuss how to properly initialize parameters for MVNNs. Importantly, the *MVNN-based uUBs* $\mathcal{M}_i^{\text{uUB}}$ are MVNNs. As we will show next, to achieve the best performance of BOCA (in fact of any MVNN training), an adjusted, non-generic parameter initialization is important.

Generic Initialization. For standard NNs, it is most common to use a parameter initialization with zero mean $\mu_k := \mathbb{E}\left[W_{j,l}^{i,k}\right] = 0$ and non-zero variance $\sigma_k^2 := \mathbb{V}\left[W_{j,l}^{i,k}\right] \neq 0$. Then the mean of each pre-activated neuron of the first hidden layer is zero and the variance $\mathbb{V}\left[\left(W^{i,1}x\right)_j\right] = d^{i,0}\sigma_1^2\overline{x^2}$, if $\left(W_{j,l}^{i,1}\right)_{l=1}^{d^{i,0}}$ are i.i.d., where $\overline{x^2} = \frac{1}{d^{i,0}}\sum_{l=1}^{d^{i,0}} x_l^2$.[6] Analogously, one can compute the *conditional* mean and the *conditional* variance of a pre-activated neuron in any layer k by replacing x by the output $z^{i,k-1}$ of the previous layer, i.e., $\mathbb{E}\left[\left(W^{i,k}z^{i,k-1}\right)_j \middle| z^{i,k-1}\right] = 0$ and $\mathbb{V}\left[\left(W^{i,k}z^{i,k-1}\right)_j \middle| z^{i,k-1}\right] = d^{i,k-1}\sigma_k^2\overline{(z^{i,k-1})^2}$. For $\sigma_k \propto \frac{1}{\sqrt{d^{i,k-1}}}$, the conditional mean and variance do not depend on the layer dimensions $d^{i,k}$, which is why generic initialization methods scale the initial distribution by $s_k \propto \frac{1}{\sqrt{d^{i,k-1}}}$.

Problem. Unfortunately, this generic initialization approach can dramatically fail for MVNNs: For any non-zero initialization, the non-negativity constraint of the weights implies that the mean $\mu_k > 0$. This implies that the mean of a pre-activated neuron in the first hidden layer is $\mathbb{E}\left[\left(W^{i,1}x\right)_j\right] = d^{i,0}\mu_1\bar{x}$. For a generic scaling s_k one would obtain $\mu_k \propto \frac{1}{\sqrt{d^{i,k-1}}}$ and thus the mean $\mathbb{E}\left[\left(W^{i,1}x\right)_j\right] \propto d^{i,0}\frac{1}{\sqrt{d^{i,0}}}\bar{x} = \sqrt{d^{i,0}}\bar{x}$ of the pre-activated neurons diverges to infinity with a rate of $\sqrt{d^{i,0}}$ as $d^{i,0} \to \infty$. Analogously, the pre-activated neurons of every layer diverge to infinity as $d^{i,k-1} \to \infty$. This is particularly problematic for bReLUs (as used in MVNNs) as their gradient is zero on $[0, t^{i,k}]^c$. Figure 3 shows that both MVNNs $\mathcal{M}_i^{\text{mean}}$ and $\mathcal{M}_i^{\text{uUB}}$ get "stuck." This happens because already at initialization, every neuron in the first hidden layer has a pre-activation that is larger than $t^{i,1}$ for every training point.

This could be solved by scaling down the initial weights even more, e.g., $W_{j,l}^{i,k} \sim \text{Unif}[0, \frac{2}{d^{i,k-1}}]$ resulting in $\mu_k = \frac{1}{d^{i,k-1}}$. However, since for $W_{j,l}^{i,k} \sim \text{Unif}[0, \frac{2}{d^{i,k-1}}]$ it holds that $\sigma_k^2 \propto \frac{1}{(d^{i,k-1})^2}$, this induces a new problem of vanishing conditional variance $\mathbb{V}\left[\left(W^{i,k}z^{i,k-1}\right)_j \middle| z^{i,k-1}\right]$ with a rate of $\mathcal{O}(\frac{1}{d^{i,k-1}})$ for wide (i.e., $d^{i,k-1}$ large) MVNNs. Overall, it is impossible to simultaneously solve both problems by just scaling the distribution by a factor s_k, because the conditional mean $\mathbb{E}\left[\left(W^{i,k}z^{i,k-1}\right)_j \middle| z^{i,k-1}\right]$ scales with $s_k \cdot d^{i,k-1}$ and the conditional variance $\mathbb{V}\left[\left(W^{i,k}z^{i,k-1}\right)_j \middle| z^{i,k-1}\right]$ scales with $s_k^2 \cdot d^{i,k-1}$. Thus, for wide MVNNs, one of those two problems (i.e., either diverging expectation or vanishing variance) would persist.

Solution. We introduce a new initialization method that solves *both* problems at the same time. For this, we propose a mixture distribution of two different uniform distributions (see Appendix Definition E.1). For each layer k, we independently sample all weights $W_{jl}^{i,k}$ i.i.d. with probability $(1 - p_k)$ from $\text{Unif}[0, A_k]$, and with probability p_k from $\text{Unif}[0, B_k]$. If we choose p_k and A_k small enough, we can get arbitrarily small μ_k while not reducing σ_k too much. In Appendix E, we provide formulas for how to choose A_k, B_k and p_k depending on $d^{i,k-1}$. In Theorem 3 in Appendix E, we prove that, if the parameters are chosen in this way, then the conditional mean and conditional variance neither explode nor vanish with increasing $d^{i,k-1}$ but rather stay con-

[6] We assume that the biases $b^{i,k} = 0$ are all initialized to zero throughout Section 3.2 to keep the notation simpler, while we formulate everything for the general case including random biases in Appendix E and in our code.

stant for large $d^{i,k-1}$. Note that, in Figure 2, for $\mathcal{M}_i^{\text{mean}}$ and $\mathcal{M}_i^{\text{uUB}}$, we used our proposed initialization method for suitable A_k, B_k and p_k, such that the problem induced by a generic initialization from Figure 3 is resolved.

3.3 Mixed Integer Linear Program (MILP)

A key step in ML-powered iterative combinatorial assignment mechanisms is finding the (predicted) social welfare-maximizing allocation, i.e., solving the *ML-based WDP*. Thus, a key requirement posed on any acquisition function $\mathcal{A}$ in such a mechanism is to be able to efficiently solve $\max_{a \in \mathcal{F}} \mathcal{A}(a)$. Recall that, to define our acquisition function $\mathcal{A}$, we use $\mathcal{A}(a) = \sum_{i \in N} \mathcal{A}_i(a_i)$ where the $\mathcal{A}_i$'s are bidder-specific upper uncertainty bounds. Thus, the ML-based WDP becomes

$$\max_{a \in \mathcal{F}} \sum_{i \in N} \mathcal{A}_i(a_i). \tag{10}$$

Weissteiner et al. (2022a) proposed a MILP for MVNNs with $\mathcal{A}_i := \mathcal{M}_i^\theta$ to efficiently solve eq. (10). Their MILP was based on a reformulation of the $\min(\cdot,\cdot)$ and $\max(\cdot,\cdot)$ in the bReLU activation $\min(\max(\cdot,0),t)$. Thus, it required twice the number of binary variables *and* linear constraints as for a plain ReLU-NN. Since we use an MVNN-based uUB $\mathcal{A}_i := \mathcal{M}_i^{\text{uUB}}$ to define our acquisition function, we could directly use their MILP formulation. However, instead, we propose a new MILP, which is significantly more succinct. For this, let $o^{i,k} := W^{i,k} z^{i,k-1} + b^{i,k}$ be the *pre*-activated output and $z^{i,k} := \varphi_{0,t^{i,k}}(o^{i,k})$ be the output of the kth layer with $l^{i,k} \leq o^{i,k} \leq u^{i,k}$, where the tight lower (upper) bound $l^{i,k}$ ($u^{i,k}$) is derived by forward-propagating the empty (full) bundle (Weissteiner et al. 2022a, Fact 1). In Theorem 2, we state our new MILP (see Appendix F.1 for the proof).[7]

Theorem 2 (MVNN MILP Tailored to Combinatorial Assignment)**.** *Let $\mathcal{A}_i = \mathcal{M}_i^{uUB}$ be our MVNN-based uUBs. The ML-based WDP (10) can be formulated as the following MILP:*

$$\max_{a \in \mathcal{F}, z^{i,k}, \alpha^{i,k}, \beta^{i,k}} \left\{ \sum_{i \in N} W^{i,K_i} z^{i,K_i-1} \right\} \tag{11}$$

s.t. for $i \in N$ and $k \in \{1,\dots,K_i-1\}$

$$z^{i,0} = a_i \tag{12}$$

$$z^{i,k} \leq \alpha^{i,k} \cdot t^{i,k} \tag{13}$$

$$z^{i,k} \leq o^{i,k} - l^{i,k} \cdot (1-\alpha^{i,k}) \tag{14}$$

$$z^{i,k} \geq \beta^{i,k} \cdot t^{i,k} \tag{15}$$

$$z^{i,k} \geq o^{i,k} + (t^{i,k} - u)\beta^{i,k} \tag{16}$$

$$\alpha^{i,k} \in \{0,1\}^{d^{i,k}},\ \beta^{i,k} \in \{0,1\}^{d^{i,k}} \tag{17}$$

Note that for each neuron of $\mathcal{A}_i = \mathcal{M}_i^{\text{uUB}}$, our new MILP has only 4 linear constraints, i.e., respective components of eqs. (13) to (16), compared to 8 in (Weissteiner et al. 2022a). Moreover, in contrast to the MILP in (Weissteiner et al. 2022a), our MILP does not make use of any "big-M" constraints, which are known to be numerically unstable.

[7] All vector inequalities should be understood component-wise.

4 Experiments

In this section, we experimentally evaluate the performance of BOCA in CAs. To this end, we equip the MLCA mechanism (see Section 2.2) with our new acquisition function $\mathcal{A}(a) = \sum_{i \in N} \mathcal{M}_i^{\text{uUB}}(a_i)$. We compare the efficiency of BOCA against the previously proposed MVNN-based and NN-based MLCA from (Weissteiner et al. 2022a) which do not explicitly model the mechanism's uncertainty over values for not yet elicited bundles.[8] We use our new parameter initialization method (Section 3.2) for $\mathcal{M}_i^{\text{uUB}}$, and we use our new MILP (Theorem 2) for solving the WDPs.

Experiment Setup. To generate synthetic CA instances, we use the following three domains from the spectrum auction test suite (SATS) (Weiss, Lubin, and Seuken 2017): LSVM, SRVM, and MRVM (see Appendix G.1 for details).[9] SATS gives us access to the true optimal allocation a^*, which we use to measure the *efficiency loss*, i.e., $1 - V(a_R^*)/V(a^*)$ when eliciting reports R via MLCA. We report efficiency loss (and not revenue), as spectrum auctions are government-run, with a mandate to maximize welfare (Cramton 2013). See Appendix G.6 for a discussion of the corresponding results on revenue. To enable a fair comparison against prior work, for each domain, we use $Q^{\text{init}} = 40$ initial random queries (including the full bundle for the calculation of $\mathcal{M}_i^{100\%\text{-uUB}}$) and set the query budget to $Q^{\max} = 100$ (see Appendix G.8 for results for $Q^{\text{init}} = 20$). We terminate any mechanism in an intermediate iteration if it already found an allocation with 0% efficiency loss.

Hyperparameter Optimization (HPO). We use *random search (RS)* (Bergstra and Bengio 2012) to optimize the hyperparameters of the mean MVNN $\mathcal{M}_i^{\text{mean}}$ and of our MVNN-based uUB $\mathcal{M}_i^{\text{uUB}}$. The HPO includes the NN-architecture parameters, training parameters, NOMU parameters, and initialization parameters (see Section 3.2). RS was carried out independently for each bidder type and SATS domain with a budget of 500 configurations, where each configuration was evaluated on 100 SATS instances. For each instance, the MVNNs $\mathcal{M}_i^{\text{mean}}$ and $\mathcal{M}_i^{\text{uUB}}$ were trained on uniformly at random chosen bundle-value pairs D^{train} and evaluated on a disjoint test set of different bundle-value pairs D^{test}. To select the winner configuration, we consider as evaluation metric the quantile-loss on the test set *and* the MAE on the training set, i.e., for each configuration and instance we calculate

$$\frac{1}{|D^{\text{test}}|} \sum_{(x,y) \in D^{\text{test}}} \max\{(y - \mathcal{M}_i^{\text{uUB}}(x))q, (\mathcal{M}_i^{\text{uUB}}(x) - y)(1-q)\} + \text{MAE}(D^{\text{train}}), \tag{18}$$

which we then average over all 100 instances. We used four quantile parameters $q \in \{0.6, 0.75, 0.9, 0.95\}$ in eq. (18)

[8] In these methods, uncertainty over not yet elicited bundles is only modeled via the retraining of the (MV)NNs in each round, i.e., the random parameter initialization of the (MV)NNs. This can be seen as simple form of Thompson sampling (see last paragraph in Appendix C).

[9] We do not use GSVM, as Weissteiner et al. (2022a) already achieved 0% efficiency loss in GSVM via MVNN-based MLCA.

		Efficiency Loss in % ↓					T-Test for Efficiency:	
Domain	Q^{MAX}	BOCA	MVNN-MLCA	NN-MLCA	FT-MLCA	RS	$\mathcal{H}_0 : \mu_{\text{MVNN-MLCA}} \leq \mu_{\text{BOCA}}$	$\mathcal{H}_0 : \mu_{\text{NN-MLCA}} \leq \mu_{\text{BOCA}}$
LSVM	100	0.39±0.30	00.70±0.40	02.91±1.44	01.54 ±0.65	31.73±2.15	$p_{\text{VAL}} = 9\text{e}{-2}$	$p_{\text{VAL}} = 3\text{e}{-4}$
SRVM	100	0.06±0.02	00.23±0.06	01.13±0.22	00.72±0.16	28.56±1.74	$p_{\text{VAL}} = 5\text{e}{-6}$	$p_{\text{VAL}} = 2\text{e}{-13}$
MRVM	100	7.77±0.34	08.16±0.41	09.05±0.53	10.37±0.57	48.79±1.13	$p_{\text{VAL}} = 8\text{e}{-2}$	$p_{\text{VAL}} = 2\text{e}{-5}$

Table 1: BOCA vs MVNN-MLCA, NN-MLCA, Fourier transform (FT)-MLCA and random search (RS). Shown are averages and a 95% CI on a test set of 50 instances. Winners based on a t-test with significance level of 1% are marked in grey.

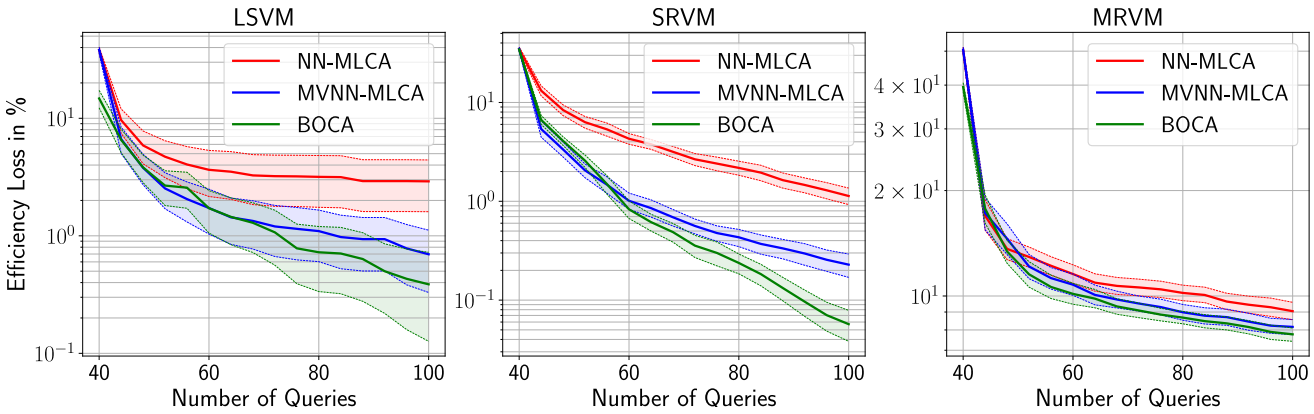


Figure 4: Efficiency loss paths (i.e., regret plots) of BOCA compared to the results from Weissteiner et al. (2022a) of MVNN-MLCA and NN-MLCA without any notion of uncertainty. Shown are averages with 95% CIs over 50 CA instances.

to achieve different levels of exploration (i.e., the resulting uUBs become larger the more we increase q in eq. (18)). This evaluation metric *simultaneously* measures the quality of the uUB on the test data (via the quantile-loss) as well as the quality of the uUB predictions on the training data (via the MAE). For each quantile q and SATS domain, we then proceed with the winner configuration of $\mathcal{M}_i^{\text{uUB}}$ and evaluate the efficiency of BOCA on a separate set of 50 instances. Details on hyperparameter ranges and the training procedure are provided in Appendices G.2 and G.3.

Results. In Table 1, we show the average efficiency loss of each approach after $Q^{\max} = 100$ queries (see Appendix G.5 for details). We see that BOCA significantly outperforms MVNN-MLCA (Weissteiner et al. 2022a) in SRVM, and it performs on-par in LSVM and MRVM, with a better average performance. Since MVNNs previously achieved SOTA performance, BOCA also outperforms the other benchmarks (i.e., NN (Weissteiner and Seuken 2020) and FT-MLCA (Weissteiner et al. 2022b)). RS's poor performance highlights the intrinsic difficulty of this task. The amount of exploration needed is domain dependent (e.g., multi-modality of the objective), which explains why the significance of BOCA's improvement varies across domains. However, our results also show that using an uUB (as in BOCA) instead of just a mean prediction (as in MVNN-MLCA) never hurts.

Figure 4 shows the efficiency loss path for all domains. We see that the superior (average) performance of $\mathcal{M}_i^{\text{uUB}}$ does not only hold at the end of the auction (at $Q^{\max} = 100$), but also for a large range of queries: in LSVM, BOCA is better for [70,100]; in SRVM, BOCA is significantly better for [70,100]; in MRVM, BOCA is better for [50,100]. See Appendix G.6 for results on revenue where BOCA significantly outperforms MVNN-MLCA also for MRVM. In Appendix G.7, we study to what degree BOCA's performance increase is due to (a) our uncertainty model (Section 3.1) versus (b) our new parameter initialization method (Section 3.2). Finally, in Appendix G.8, we provide further experiments for a reduced number of $Q^{\text{init}} = 20$ initial queries, which lead to similar results as shown in Table 1.

5 Conclusion

In this paper, we have proposed a Bayesian optimization-based combinatorial assignment (BOCA) mechanism. On a conceptual level, our main contribution was the integration of model uncertainty over agents' preferences into ML-based preference elicitation. On a technical level, we have designed a new method for estimating an upper uncertainty bound that exploits the monotonicity of agents' preferences in the combinatorial assignment domain and the finite nature of this setting. Our experiments have shown that BOCA performs as good or better than the SOTA in terms of efficiency. An interesting direction for future work is the evaluation of BOCA in other combinatorial assignment domains, such as combinatorial exchanges or course allocation (e.g., see (Soumalias et al. 2023)). Finally, it would also be interesting to apply BOCA's conceptual idea in the combinatorial BO settings outside of combinatorial assignment.

Acknowledgments

We thank the anonymous reviewers for helpful comments. This paper is part of a project that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (Grant agreement No. 805542).

References

Ausubel, L.; and Cramton, P. 2011. Auction design for wind rights. *Report to Bureau of Ocean Energy Management, Regulation and Enforcement*. 1

Ausubel, L. M.; and Baranov, O. 2017. A practical guide to the combinatorial clock auction. *Economic Journal*, 127(605): F334–F350. 1, 25

Ausubel, L. M.; Cramton, P.; and Milgrom, P. 2006. The clock-proxy auction: A practical combinatorial auction design. In Cramton, P.; Shoham, Y.; and Steinberg, R., eds., *Combinatorial Auctions*, 115–138. MIT Press. 1

Bergstra, J.; and Bengio, Y. 2012. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2). 7

Bichler, M.; Fux, V.; and Goeree, J. K. 2019. Designing combinatorial exchanges for the reallocation of resource rights. *Proceedings of the National Academy of Sciences*, 116(3): 786–791. 1

Blundell, C.; Cornebise, J.; Kavukcuoglu, K.; and Wierstra, D. 2015. Weight uncertainty in neural networks. In *32nd International Conference on Machine Learning (ICML)*. 2

Bonilla, E. V.; Guo, S.; and Sanner, S. 2010. Gaussian process preference elicitation. *Advances in neural information processing systems*, 23. 2, 13

Brero, G.; Lubin, B.; and Seuken, S. 2018. Combinatorial Auctions via Machine Learning-based Preference Elicitation. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. 1

Brero, G.; Lubin, B.; and Seuken, S. 2021. Machine Learning-powered Iterative Combinatorial Auctions. *arXiv preprint arXiv:1911.08042*. 1, 2, 3, 11, 12

Budish, E. 2011. The combinatorial assignment problem: Approximate competitive equilibrium from equal incomes. *Journal of Political Economy*, 119(6): 1061–1103. 1

Chapelle, O.; and Li, L. 2011. An Empirical Evaluation of Thompson Sampling. In *NIPS*. 3, 13

Cramton, P. 2013. Spectrum auction design. *Review of Industrial Organization*, 42(2): 161–190. 1, 7

De Ath, G.; Everson, R. M.; Rahat, A. A.; and Fieldsend, J. E. 2021. Greed is good: Exploration and exploitation trade-offs in Bayesian optimisation. *ACM Transactions on Evolutionary Learning and Optimization*, 1(1): 1–22. 13, 25

Frazier, P. I. 2018. A tutorial on Bayesian optimization. *arXiv preprint arXiv:1807.02811*. 3, 13

Gal, Y.; and Ghahramani, Z. 2016. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *33rd International Conference on Machine Learning (ICML)*, 1050–1059. 2, 16

Goeree, J. K.; and Holt, C. A. 2010. Hierarchical package bidding: A paper & pencil combinatorial auction. *Games and Economic Behavior*, 70(1): 146–169. 22

Graves, A. 2011. Practical variational inference for neural networks. In *Advances in neural information processing systems*, 2348–2356. 2

Heiss, J. M.; Weissteiner, J.; Wutte, H. S.; Seuken, S.; and Teichmann, J. 2022. NOMU: Neural Optimization-based Model Uncertainty. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, 8708–8758. PMLR. 2, 4, 5, 13, 14, 15, 16, 17, 24

Hernández-Lobato, J. M.; and Adams, R. 2015. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning*, 1861–1869. 2

Kandasamy, K.; Dasarathy, G.; Schneider, J.; and Póczos, B. 2017. Multi-fidelity bayesian optimisation with continuous approximations. In *International Conference on Machine Learning*, 1799–1808. PMLR. 2

Kuleshov, V.; Fenner, N.; and Ermon, S. 2018. Accurate Uncertainties for Deep Learning Using Calibrated Regression. In *Proceedings of the 35th International Conference on Machine Learning*, 2796–2804. PMLR. 16

Lakshminarayanan, B.; Pritzel, A.; and Blundell, C. 2017. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in neural information processing systems*, 6402–6413. 2, 13, 16

Nisan, N.; and Segal, I. 2006. The communication requirements of efficient allocations and supporting prices. *Journal of Economic Theory*, 129(1): 192–224. 1

Ober, S. W.; and Rasmussen, C. E. 2019. Benchmarking the neural linear model for regression. *arXiv preprint arXiv:1912.08416*. 2

Papalexopoulos, P.; Tjandraatmadja, C.; Anderson, R.; Vielma, J. P.; and Belanger, D. 2022. Constrained Discrete Black-Box Optimization using Mixed-Integer Programming. In *Proceedings of the 39th International Conference on Machine Learning*, 17295–17322. PMLR. 13, 14

Sandholm, T.; and Boutilier, C. 2006. Preference elicitation in combinatorial auctions. *Combinatorial auctions*, 10. 1

Scheffel, T.; Ziegler, G.; and Bichler, M. 2012. On the impact of package selection in combinatorial auctions: an experimental study in the context of spectrum auction design. *Experimental Economics*, 15(4): 667–692. 22

Soumalias, E.; Zamanlooy, B.; Weissteiner, J.; and Seuken, S. 2023. Machine Learning-powered Course Allocation. *arXiv preprint arXiv:2210.00954*. 8

Srinivas, N.; Krause, A.; Kakade, S. M.; and Seeger, M. W. 2012. Information-Theoretic Regret Bounds for Gaussian Process Optimization in the Bandit Setting. *IEEE Transactions on Information Theory*, 58(5): 3250–3265. 3, 13

Weiss, M.; Lubin, B.; and Seuken, S. 2017. Sats: A universal spectrum auction test suite. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, 51–59. 5, 7, 22

Weissteiner, J.; Heiss, J.; Siems, J.; and Seuken, S. 2022a. Monotone-Value Neural Networks: Exploiting Preference Monotonicity in Combinatorial Assignment. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, 541–548. International Joint Conferences on Artificial Intelligence Organization. Main Track. 2, 3, 4, 7, 8, 13, 14, 21, 22, 24, 25, 26

Weissteiner, J.; Heiss, J.; Siems, J.; and Seuken, S. 2023. Bayesian Optimization-based Combinatorial Assignment. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37. 1

Weissteiner, J.; and Seuken, S. 2020. Deep Learning—Powered Iterative Combinatorial Auctions. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(02): 2284–2293. 2, 8, 13, 14

Weissteiner, J.; Wendler, C.; Seuken, S.; Lubin, B.; and Püschel, M. 2022b. Fourier Analysis-based Iterative Combinatorial Auctions. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, 549–556. International Joint Conferences on Artificial Intelligence Organization. Main Track. 2, 8

Wenzel, F.; Snoek, J.; Tran, D.; and Jenatton, R. 2020. Hyperparameter ensembles for robustness and uncertainty quantification. *arXiv preprint arXiv:2006.13570*. 2

Appendix

A A Machine Learning-Powered ICA

In this section, we present in detail the *machine learning-powered combinatorial auction (MLCA)* by Brero, Lubin, and Seuken (2021).

At the core of MLCA is a *query module* (Algorithm 1), which, for each bidder $i \in I \subseteq N$, determines a new value query q_i. First, in the *estimation step* (Line 1), an ML algorithm $\mathcal{A}_i$ is used to learn bidder i's valuation from reports R_i. Next, in the *optimization step* (Line 2), an *ML-based WDP* is solved to find a candidate q of value queries. In principle, any ML algorithm $\mathcal{A}_i$ that allows for solving the corresponding ML-based WDP in a fast way could be used. Finally, if q_i has already been queried before (Line 4), another, more restricted ML-based WDP (Line 6) is solved and q_i is updated correspondingly. This ensures that all final queries q are new.

Algorithm 1: NEXTQUERIES(I, R) (Brero et al. 2021)

Inputs: Index set of bidders I and reported values R
1 **foreach** $i \in I$ **do** Fit $\mathcal{A}_i$ on R_i: $\mathcal{A}_i[R_i]$ ▷ `Estimation step`
2 Solve $q \in \operatorname{argmax}_{a \in \mathcal{F}} \sum_{i \in I} \mathcal{A}_i[R_i](a_i)$ ▷ `Optimization step`
3 **foreach** $i \in I$ **do**
4 | **if** $(q_i, \hat{v}_i(q_i)) \in R_i$ **then** ▷ `Bundle already queried`
5 | | Define $\mathcal{F}' = \{a \in \mathcal{F} : a_i \neq x, \forall (x, \hat{v}_i(x)) \in R_i\}$
6 | | Re-solve $q' \in \operatorname{argmax}_{a \in \mathcal{F}'} \sum_{l \in I} \mathcal{A}_l[R_l](a_l)$
7 | | Update $q_i = q'_i$
8 | **end**
9 **end**
10 **return** *Profile of new queries* $q = (q_1, \ldots, q_n)$

In Algorithm 2, we present MLCA. In the following, let $R_{-i} = (R_1, \ldots, R_{i-1}, R_{i+1}, \ldots, R_n)$. MLCA proceeds in rounds until a maximum number of queries per bidder Q^{max} is reached. In each round, it calls Algorithm 1 $(Q^{\text{round}}-1)n+1$ times: for each bidder $i \in N$, $Q^{\text{round}} - 1$ times excluding a different bidder $j \neq i$ (Lines 5–10, sampled *marginal economies*) and once including all bidders (Line 11, *main economy*). In total each bidder is queried Q^{round} bundles per round in MLCA. At the end of each round, the mechanism receives reports R^{new} from all bidders for the newly generated queries q^{new} and updates the overall elicited reports R (Lines 12–14). In Lines 16–17, MLCA computes an allocation a^*_R that maximizes the *reported* social welfare (see Equation (1)) and determines VCG payments $p(R)$ based on the reported values R (see Appendix Definition B.1).

In this paper, we consider the following two minor adaptations of the generic MLCA mechanism described above:

1. **Balanced and global marginal economies:** In Lines 5–6 of Algorithm 2, MLCA draws *for each bidder* $i \in N$ uniformly at random a set of marginal economies $\tilde{N}$ to generate queries in this marginal economy. However, this implies that at the end of the auction it only holds on average that the same number of queries is asked from each bidder in each marginal economy. Moreover, since

Algorithm 2: MLCA$(Q^{\text{init}}, Q^{\text{max}}, Q^{\text{round}})$ (Brero et al. 2021)

Params: $Q^{\text{init}}, Q^{\text{max}}, Q^{\text{round}}$ initial, max and #queries/round
1 **foreach** $i \in N$ **do**
2 | Receive reports R_i for Q^{init} randomly drawn bundles
3 **end**
4 **for** $k = 1, \ldots, \lfloor (Q^{max} - Q^{init})/Q^{round} \rfloor$ **do** ▷ `Round iterator`
5 | **foreach** $i \in N$ **do** ▷ `Marginal economy queries`
6 | | Draw uniformly without replacement (Q^{round}–1) bidders from $N \setminus \{i\}$ and store them in $\tilde{N}$
7 | | **foreach** $j \in \tilde{N}$ **do**
8 | | | $q^{\text{new}} = q^{\text{new}} \cup \mathit{NextQueries}(N \setminus \{j\}, R_{-j})$
9 | | **end**
10 | **end**
11 | $q^{\text{new}} = \mathit{NextQueries}(N, R)$ ▷ `Main economy queries`
12 | **foreach** $i \in N$ **do**
13 | | Receive reports R_i^{new} for q_i^{new}, set $R_i = R_i \cup R_i^{\text{new}}$
14 | **end**
15 **end**
16 Given elicited reports R compute a^*_R as in Equation (1)
17 Given elicited reports R compute VCG-payments $p(R)$
18 **return** *Final allocation* a^*_R *and payments* $p(R)$

$\tilde{N}$ is re-drawn for each bidder this creates an computational overhead, since typically the WDPs in Line 8 in *NextQueries* needs to be solved more often. For example consider the case $N = \{1, 2, 3, 4, 5, 6\}, Q^{\text{round}} = 3$ and that for bidder 1 the $Q^{\text{round}} - 1 = 2$ sampled marginal economies were given as $\tilde{N} = \{3, 4\}$, whilst for bidder 2, $\tilde{N} = \{5, 6\}$, and for bidder 3, $\tilde{N} = \{1, 2\}$. In this case, the WDPs in *NextQueries* would need to be solved 6 times, which is the maximum possible. In our implementation, we change the following two things: First, we reduce the computational overhead by once globally selecting Q^{round} marginal economies in each iteration, i.e., we select a set $\tilde{N}_{global}$ consisting of Q^{round} marginal economies before Line 5, and then select $\tilde{N}$ for each bidder $i \in N$ in the loop in Line 5 as admissible subset of size $Q^{\text{round}} - 1$ of $\tilde{N}_{global}$. In the above example, if $\tilde{N}_{global} = \{3, 4, 1\}$, then this ensures that only $Q^{\text{round}} = 3$ WDPs in the marginal economies are solved in one iteration. Second, we do not determine $\tilde{N}_{global}$ uniformly at random, but ensure that at the end of the auction each marginal economy was selected equally often up to a difference in counts of at most one.

2. **Single training per iteration:** To further reduce computational overhead, we train each bidder's ML algorithm $\mathcal{A}_i$ once at the beginning of each iteration, and then only select in *NextQueries* the trained $\mathcal{A}_i$ corresponding to the active set of bidders I. This reduces the amount of total training procedures per iteration from (worst case) n^2 to n.

B Incentives of MLCA

In this section, we review the key arguments by Brero, Lubin, and Seuken (2021) why MLCA has good incentives in

practice. First, we define VCG-payments[10] given bidder's reports.

Definition B.1. (VCG PAYMENTS FROM REPORTS) *Let $R = (R_1, \ldots, R_n)$ denote an elicited set of reported bundle-value pairs from each bidder obtained from* MLCA *(Algorithm 2) and let $R_{-i} := (R_1, \ldots, R_{i-1}, R_{i+1}, \ldots, R_n)$. We then calculate the VCG payments $p(R) = (p(R)_1 \ldots, p(R)_n) \in \mathbb{R}^n_+$ as follows:*

$$p(R)_i := \sum_{j \in N \setminus \{i\}} \hat{v}_j \left(\left(a^*_{R_{-i}} \right)_j \right) - \sum_{j \in N \setminus \{i\}} \hat{v}_j \left((a^*_R)_j \right). \tag{19}$$

where $a^*_{R_{-i}}$ maximizes the reported social welfare (SW) when excluding bidder i, i.e.,

$$a^*_{R_{-i}} \in \operatorname*{argmax}_{a \in \mathcal{F}} \widehat{V}(a|R_{-i}) = \operatorname*{argmax}_{a \in \mathcal{F}} \sum_{\substack{j \in N \setminus \{i\}: \\ (a_j, \hat{v}_j(a_j)) \in R_j}} \hat{v}_j(a_j), \tag{20}$$

and a^*_R is a reported-social-welfare-maximizing allocation (including all bidders), i.e,

$$a^*_R \in \operatorname*{argmax}_{a \in \mathcal{F}} \widehat{V}(a|R) = \operatorname*{argmax}_{a \in \mathcal{F}} \sum_{i \in N: (a_i, \hat{v}_i(a_i)) \in R_i} \hat{v}_i(a_i). \tag{21}$$

Therefore, when using VCG, bidder i's utility is:

$$\begin{aligned} u_i =& v_i((a^*_R)_i) - p(R)_i \\ =& \underbrace{v_i((a^*_R)_i) + \sum_{j \in N \setminus \{i\}} \hat{v}_j((a^*_R)_j)}_{\text{(a) Reported SW of main economy}} - \underbrace{\sum_{j \in N \setminus \{i\}} \hat{v}_j((a^*_{R_{-i}})_j)}_{\text{(b) Reported SW of marginal economy}}. \end{aligned}$$

Any beneficial misreport must increase the difference (a) − (b).

MLCA has two features that mitigate manipulations. First, MLCA explicitly queries each bidder's marginal economy (Algorithm 2, Line 5), which implies that (b) is practically independent of bidder i's bid (Section 7.3 in (Brero, Lubin, and Seuken 2021) provides experimental support for this). Second, MLCA enables bidders to "push" information to the auction which they deem useful. This mitigates certain manipulations that target (a), as it allows bidders to increase (a) with truthful information. Brero, Lubin, and Seuken (2021) argue that any remaining manipulation would be implausible as it would require almost complete information.[11]

If we are willing to make further assumptions, we also obtain two theoretical incentive guarantees:

- Assumption 1 requires that, for all bidders $i \in N$, if all other bidders report truthfully, then the reported social welfare of bidder i's marginal economy (i.e., term (b)) is *independent* of her value reports.
- Assumption 2 requires that, if all bidders $i \in N$ bid truthfully, then MLCA *finds an efficient allocation*.

Result 1: Social Welfare Alignment If Assumption 1 holds, and if all other bidders are truthful, then MLCA is social welfare aligned, i.e., increasing the reported social welfare of a^*_R in the main economy (i.e., term (a)), which in this case equals the *true* social welfare of a^*_R, is the only way for a bidder to increase her true utility (Brero, Lubin, and Seuken 2021, Proposition 3).

Result 2: Ex-Post Nash Equilibrium Moreover, if Assumption 1 *and* Assumption 2 hold, then bidding truthfully is an ex-post Nash equilibrium in MLCA (Brero, Lubin, and Seuken 2021, Proposition 4).

C BO Perspective of ICAs

In this section, we discuss the Bayesian optimization (BO) perspective of iterative combinatorial assignment (see Section 2.3). Specifically, we analyze the MLCA mechanism (see Section 2.2 for an overview or Appendix A for a detailed description) in the light of BO.

Iterative combinatorial assignment can be seen as a combinatorial BO task with an expensive-to-evaluate function:

- The objective (e.g., social welfare) in general lacks known structure and when evaluating it (e.g., value queries) one only observes the objective at a single input point and no derivatives such that gradient-based optimization cannot be used.
- Typically one can only query a very limited amount of information to find an approximately optimal allocation, For example, in a real-world spectrum auction, the auctioneer can only ask each bidder to answer on the order of 100 value queries for different bundles of items, even though the space of possible bundles is exponential in the number of items m, i.e., there are 2^m possible bundles and $(n+1)^m$ possible allocations.

C.1 BO Perspective of MLCA

In this paper, we extend prior work on MLCA with a notion of uncertainty that makes MLCA more similar to classic BO. Specifically, we now use an MVNN-based upper uncertainty bound (uUB) to define our acquisition function. This allows MLCA to trade-off exploration and exploitation making it more likely to find optimal allocations.

However, in addition to the challenges that arise in BO, combinatorial assignment adds their own set of challenges. For example, Gaussian process-based BO often does not extend beyond 10-20 input dimensions, which is problematic as in combinatorial assignment the input space can be much larger, e.g., for $m = 98$ items and $n = 10$ bidder the input space would be 980 dimensional (MRVM). In addition, integrality constraints to obtain only whole items (i.e., combinatorial assignment deals with assigning m *indivisible* items to agents) and feasibility constraints that ensure each item is

[10] VCG is an abbreviation for "Vickrey–Clarke–Groves". For the VCG-mechanism (which can be seen as a generalization of the second prize auction) it is always optimal for the bidders to report the truth.

[11] In this paper, we propose a new method that uses a notion of epistemic uncertainty to actively explore regions of the bundle space with high uncertainty. Intuitively, this makes manipulation even harder, since additional exploration makes it more difficult for a bidder to prevent other bidders from getting queries in certain regions.

only allocated once to a single agent are often only incorporated via rounding or randomization.

Our work addresses both problems by combining (MV)NN-based MLCA by (Weissteiner and Seuken 2020; Weissteiner et al. 2022a) with the recently introduced NOMU (Heiss et al. 2022), an optimization-based method to obtain uncertainty estimates for the predictions of NNs. Importantly, NOMU enables to represent an uUB as a *single* NN in contrast to other more expensive methods such as ensembles (Lakshminarayanan, Pritzel, and Blundell 2017). Thus, NOMU is particularly suited for iterative combinatorial assignment, where uUB-based WDPs are solved hundreds of times to generate informative queries.

C.2 How does BOCA Differ from Classical BO?

There are two main differences that set BOCA from classical BO apart.

1. **More information per query:** In classical BO, one would obtain only one number $f(x)$ per query, which would correspond to $f(x) = \sum_{i \in N} \hat{v}_i(a_i)$ in the case of ICA. However, we obtain all the individual values $\hat{v}_i(a_i)$ instead of only obtaining their sum. This additional information is very valuable for MLCAs such as BOCA. We benefit from this additional information by representing our acquisition function $\mathcal{A}(a) = \sum_{i \in N} \mathcal{A}_i(a_i)$ as sum of functions $\mathcal{A}_i$, which are trained based on their individual values. The benefits of this additional information are even more valuable because of our very strong prior knowledge on the individual value functions $\hat{v}_i$ (e.g., monotonicity) compared to rather vague prior on their sum. Also our notion of uncertainty incorporates this additional information.
   In classical BO $\operatorname{argmax}_{x \in \{x^{(1)}, \dots, x^{(T)}\}} f(x)$ is outputted as an approximation for $\operatorname{argmax}_{x \in X} f(x)$. However, the fact that we revive the individual values $\hat{v}_i(a_i)$, allows us to output the solution a^*_R of (1) instead of just outputting the best allocation that we have queried. In other words (1) allows us to recombine queried bundles potentially differently than in any queried *allocation*.
2. **Multiple objectives:** In classical BO there is only one objective function f. In the case of MLCA, this corresponds to the primary objective $f(x) = \sum_{i \in N} \hat{v}_i(a_i)$. However, besides social welfare, we also have the secondary objective of achieving a high revenue in the case of MLCA. The revenue depends on the payments $p(R)$ from Definition B.1. In order to compute the payments $p(R)$, we need to approximate n further maximization problems (i.e., the n marginal economies $\sum_{j \in N \setminus \{i\}} \hat{v}_j(a_j)$, see Appendices A and B). Since all this $n+1$ objectives are different sub-sums of the same n unknown functions $\hat{v}_i$ each bundle-value-pair $(a_i, \hat{v}_i(a_i))$ can provide useful information for a better approximation of n objectives. Intuitively speaking, we think that in a multi-objective setting, exploration is particularly useful (compared to exploitation), since the uncertainty of the individual value functions are independent of the different objectives. If a bundle of high uncertainty is queried one gains plenty of new information which can be very helpful for the better approximation of n objectives. Whereas, very specialized queries that seem to be optimal with respect to one specific objective might not be as helpful for the other objectives in general.

C.3 Choice of Acquisition Function

Recall that in BOCA we use as acquisition function $\mathcal{A}(a) = \sum_{i \in N} \mathcal{A}_i(a_i)$ where the $\mathcal{A}_i$'s are bidder-specific MVNN-based upper uncertainty bounds (uUBs). Thus, the WDP $\max_{a \in \mathcal{F}} \mathcal{A}(a)$, which we solve to generate new informative queries, decomposes to

$$\max_{a \in \mathcal{F}} \mathcal{A}(a) = \max_{a \in \mathcal{F}} \sum_{i \in N} \mathcal{A}_i(a_i). \tag{22}$$

The key design choice to make solving this WDP practically feasible, is that we were able to encode each uUB $\mathcal{A}_i$ as a succinct MILP such that the acquisition function $\mathcal{A}(a) = \sum_{i \in N} \mathcal{A}_i(a_i)$ as a whole is also MILP-encodable and thus the WDP can be reformulated itself as a MILP. Importantly, our acquisition function $\mathcal{A}$ was carefully chosen to make solving this WDP practically feasible for moderate-sized MVNNs. In the following, we briefly discuss the usage (i.e., advantages and disadvantages) of other popular acquisition functions in the combinatorial assignment setting.

Popular examples of acquisition functions in classical BO from the literature include:

- *Upper uncertainty bound* (uUB) (aka *upper confidence bound or upper credible bound (UCB)*) (e.g., see (Srinivas et al. 2012)),
- *Expected improvement* (e.g., see (Frazier 2018, Section 4.1)),
- *Thompson sampling* (e.g., see (Chapelle and Li 2011)),
- *Probability of improvement* (e.g., see (De Ath et al. 2021, Section 2.2.4)),
- *Expected value of information* (e.g., see (Bonilla, Guo, and Sanner 2010)).

Upper Uncertainty Bound (uUB) We decided to use an UCB-type acquisition function since this enables a succinct MILP formulation of the acquisition function optimization (i.e., the WDP). Moreover, this also offers good performance in practice and good intuitive and theoretical motivation (Srinivas et al. 2012).

Expected improvement (EI) While it would be theoretically optimal to use EI for the very last query, it is neither theoretically nor empirically clear if EI is better or worse than the uUB for all other queries. We are not aware of any practically feasible solver in the combinatorial assignment domain of the corresponding WDP based on EI and on a monotonic regression technique. Moreover, we do not see any straightforward practically feasible implementation and we do not expect any significant improvement from EI over uUB.

Thompson Sampling Recently, Papalexopoulos et al. (2022) also proposed a MILP-based BO method using a ReLU NN as surrogate model and Thompson sampling as acquisition function. Concretely, Papalexopoulos et al.

(2022) approximate Thompson sampling via retraining of the NN from scratch with a new random initialization. Subsequently, they determine the next query by solving an NN-based MILP. However, unlike our proposal, Papalexopoulos et al. (2022) do only implicitly integrate a very limited notion of uncertainty, i.e., via a random parameter initialization of the NN, making their approach conceptually equivalent to the (MV)NN-based MLCA by Weissteiner and Seuken (2020); Weissteiner et al. (2022a). In particular, this approximation of Thompson sampling only achieves sufficient exploration if the diversity induced by different random initialization seeds is large enough. However, Heiss et al. (2022, Remark B.5) showed that the diversity of an NN ensemble (in noiseless settings) is rather small and in (Heiss et al. 2022, Remark B.5) the ensemble's uncertainty needed to be scaled up by a factor of ~ 10 to make its uncertainty competitive. However, in Thompson sampling the desired amount of uncertainty/exploration cannot be easily calibrated/scaled. Moreover, our experimental results in spectrum auctions suggests that indeed the method by Weissteiner et al. (2022a), which uses conceptually the same notion of uncertainty as in (Papalexopoulos et al. 2022), is outperformed by our proposal.

Probability of Improvement (PI) Intuitively we believe that PI is inferior to EI and uUB. We are not aware of any theoretical or empirical results that suggest significant advantages of PI. We are not aware of any practically feasible solver of the corresponding WDP based on PI and on a monotonic regression technique.

Expected Value of Information (EVOI) All previously mentioned acquisition functions are still to some extend myopic/greedy, as they do not optimize explicitly for multiple queries ahead (while still being significantly less myopic than just using the mean prediction as acquisition function). E.g., as mentioned above, EI is only optimal for the last query. The other acquisition functions are not mathematically optimal in any provable sense. In theory, EVOI is optimal for a horizon of two BO-steps. In principle, one could adapt it recursively to be optimal also for longer horizons, i.e. multiple BO-steps. While this approach would be the most appealing from a purely theoretical point of view, it would also be by far the hardest from a computational point of view (because of the nested EI-like optimization problems). We are not aware of any practically feasible solver of the corresponding WDP based on EVOI and on a monotonic regression technique. We believe this would be a very exciting direction for long-term future research. For example, a very weak approximation of an EVOI-WDP could outperform a good approximation of our uUB-WDP, especially if compute power significantly increases in the future.

D NOMU for Monotone NNs

In this section, we give more details regarding our uncertainty estimates for monotonically increasing functions based on NOMU (see Section 3.1).

D.1 Proof of Theorem 1

The 100%- uUB $f^{100\%\text{-uUB}}(x) := \sup_{f \in \mathcal{H}_{D^{\text{train}}}} f(x)$ is defined via a 2^m-dimensional optimization problem with n^{train} constraints. In the following proof, we analytically derive the explicit closed-form joint solution of 2^m such optimization problems (one for each x), which we can represent as an MVNN that does not require *any* optimization algorithm.

Proof of Theorem 1. In the 1st part of the proof (which is based on the proof of Weissteiner et al. (2022a, Theorem 1)), we explicitly construct $\mathcal{M}_i^{100\%\text{-uUB}}$ and show that $\mathcal{M}_i^{100\%\text{-uUB}} \in \mathcal{V}_{D^{\text{train}}}$. Equation (27) gives our explicit (fast to evaluate) closed-form formula for $\mathcal{M}_i^{100\%\text{-uUB}}$, which can be directly written down without any training algorithm.

The 2nd part of the proof (which is new) shows that $\mathcal{M}_i^{100\%\text{-uUB}}$ is indeed the 100%-uUB by showing that it is maximal (and thus the supremum is actually a maximum).

1. Let $\hat{v}_i \in \mathcal{V}$ and let $D^{\text{train}} = \{(x^{(l)}, \hat{v}_i(x^{(l)}))\}_{l=1}^{n^{\text{train}}}$ be a set of n^{train} observed training points corresponding to $\hat{v}_i$. First, given D^{train}, we construct an MVNN $\mathcal{M}_i^{100\%\text{-uUB}}$ with weights $\theta = (W^i_{D^{\text{train}}}, b^i_{D^{\text{train}}})$ such that $\mathcal{M}_i^{100\%\text{-uUB}}(x) = f^{100\%\text{-uUB}}(x)$ for all $x \in \mathcal{X}$.
   Recall, that by definition $\hat{v}_i((0,\dots,0)) = 0$ and that we assume that $((1\dots,1), \hat{v}_i((1\dots,1))) \in D^{\text{train}}$. Now let $(w_l)_{l=0}^{n^{\text{train}}}$ denote the observed/known values corresponding to $\hat{v}_i$ sorted in increasing order, i.e, let $x^{(0)} = (0,\dots,0)$ with
   $$w_0 := \hat{v}_i(x^{(0)}) = 0, \tag{23}$$
   let $x^{(n^{\text{train}})} = (1,\dots,1)$ with
   $$w_{n^{\text{train}}} := \hat{v}_i(x^{(n^{\text{train}})}), \tag{24}$$
   and $x^{(j)}, x^{(k)} \in \mathcal{X} \setminus \{x^{(0)}, x^{(n^{\text{train}})}\}$ for $0 < j < k \leq n^{\text{train}} - 1$ with
   $$w_j := \hat{v}_i(x^{(j)}) \leq w_k := \hat{v}_i(x^{(k)}). \tag{25}$$
   In the following, we slightly abuse the notation and write for $x^{(j)}, x^{(k)} \in \mathcal{X}$, $x^{(j)} \subseteq x^{(k)}$ iff for the corresponding sets $A^j, A^k \in 2^M$ it holds that $A^j \subseteq A^k$. Furthermore, we denote by $\langle \cdot, \cdot \rangle$ the Euclidean scalar product on $\mathbb{R}^m$. Before we show that our construction fulfills $\mathcal{M}_i^{100\%\text{-uUB}} \in \mathcal{V}_{D^{\text{train}}}$, we define it as
   $$\mathcal{M}_i^{100\%\text{-uUB}}(x) := \sum_{k=0}^{n^{\text{train}}-1} (w_{k+1} - w_k)\, \mathbb{1}_{\{\forall j \in \{0,\dots,k\} \,:\, x \not\subseteq x^{(j)}\}} \tag{26}$$
   $$= \sum_{k=0}^{n^{\text{train}}-1} (w_{k+1} - w_k)\, \varphi_{0,1}\left(\sum_{j=0}^{k} \varphi_{0,1}\left(\left\langle 1 - x^{(j)}, x \right\rangle \right) - k \right), \tag{27}$$
   where the second equality follows since
   $$x \not\subseteq x^{(j)} \iff \left\langle 1 - x^{(j)}, x \right\rangle \geq 1 \tag{28}$$
   $$\iff \varphi_{0,1}\left(\left\langle 1 - x^{(j)}, x \right\rangle \right) = 1, \tag{29}$$

which implies that

$$\forall j \in \{0, \dots, k\} : x \not\subseteq x^{(j)} \tag{30}$$

$$\iff \sum_{j=0}^{k} \varphi_{0,1}\left(\left\langle 1 - x^{(j)}, x \right\rangle\right) = k + 1, \tag{31}$$

and

$$\mathbb{1}_{\left\{\forall j \in \{0,\dots,k\} : x \not\subseteq x^{(j)}\right\}} = \varphi_{0,1}\left(\sum_{j=0}^{k} \varphi_{0,1}\left(\left\langle 1 - x^{(j)}, x \right\rangle\right) - k\right). \tag{32}$$

We now show that $\mathcal{M}_i^{100\%\text{-uUB}}$ is actually an MVNN. Equation (27) can be equivalently written in matrix notation as

$$\underbrace{\begin{bmatrix} w_1 - w_0 \\ w_2 - w_1 \\ \vdots \\ w_{n^{\text{train}}} - w_{n^{\text{train}}-1} \end{bmatrix}^\top}_{(W^{i,3}_{D^{\text{train}}})^\top \in \mathbb{R}^{n^{\text{train}}}_{\geq 0}} \varphi_{0,1}\left(W^{i,2}_{D^{\text{train}}} \varphi_{0,1}\left(\underbrace{\begin{bmatrix} 1 - x^{(0)} \\ 1 - x^{(1)} \\ \vdots \\ 1 - x^{(n^{\text{train}}-1)} \end{bmatrix}}_{W^{i,1}_{D^{\text{train}}} \in \mathbb{R}^{(n^{\text{train}}) \times m}_{\geq 0}} x\right) + \underbrace{\begin{bmatrix} 0 \\ -1 \\ \vdots \\ -(n^{\text{train}} - 1) \end{bmatrix}}_{b^{i,2}_{D^{\text{train}}} \in \mathbb{R}^{n^{\text{train}}}_{\leq 0}}\right)$$

with $W^{i,2}_{D^{\text{train}}} \in \mathbb{R}^{(n^{\text{train}}) \times (n^{\text{train}})}_{\geq 0}$ a lower triangular matrix of ones, i.e.,

$$W^{i,2}_{D^{\text{train}}} := \begin{bmatrix} 1 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & 0 \\ 1 & \dots & \dots & 1 \end{bmatrix}.$$

From that, we can see that $\mathcal{M}_i^{100\%\text{-uUB}}$ is indeed an MVNN with four layers in total (i.e., two hidden layers) and respective dimensions $[m, n^{\text{train}}, n^{\text{train}}, 1]$. From Equation (26) we can see that for all $l = 0, \dots, n^{\text{train}}$: $\mathcal{M}_i^{100\%\text{-uUB}}(x^{(l)}) = \hat{v}_i(x^{(l)})$ and therefore $\mathcal{M}_i^{100\%\text{-uUB}} \in \mathcal{V}_{D^{\text{train}}}$.

2. Now we have to check what happens for $x \in \mathcal{X}$: $(x, \hat{v}_i(x)) \notin D^{\text{train}}$. Therefore, let $x \in \mathcal{X} : (x, \hat{v}_i(x)) \notin D^{\text{train}}$. Then, by definition we get that $\mathcal{M}_i^{100\%\text{-uUB}}(x) = w_k$, where $k \in \{0, \dots, n^{\text{train}}\}$ is the smallest integer such that $x \subseteq x^{(k)}$. Assume there exists an $h \in \mathcal{V}_{D^{\text{train}}}$ with $h(x) > \mathcal{M}_i^{100\%\text{-uUB}}(x) = w_k = h(x^{(k)})$. However, since $x \subseteq x^{(k)}$ this is a contradiction to h fulfilling the monotonicity property. Thus, we get that $\mathcal{M}_i^{100\%\text{-uUB}}(x) = \max_{f \in \mathcal{V}_{D^{\text{train}}}} f(x)$. Since x was chosen arbitrarily, we finally get that $\mathcal{M}_i^{100\%\text{-uUB}}(x) = \max_{f \in \mathcal{V}_{D^{\text{train}}}} f(x)$ for all $x \in \mathcal{X}$ which concludes the proof.

□

D.2 ICA-Based New NOMU Architecture

In this section, we provide more details on our carefully chosen ICA-based new NOMU architecture. The original NOMU architecture from (Heiss et al. 2022) outputs a mean prediction $\hat{f}$, which is in our case the mean-MVNN $\mathcal{M}_i^{\text{mean}}$ and a model uncertainty prediction $\hat{\sigma}_f$. Then uUBs in the original NOMU algorithm at an input point x are defined as $\hat{f}(x) + c \cdot \hat{\sigma}_f(x)$ for $c > 0$.

Monotone uUBs and Non-Monotone $\hat{\sigma}_f$

Monotone Value Functions Imply Monotone uUBs. Knowing that the unknown ground truth function is monotonically increasing[12] means that the support of the prior in function space only contains monotonically increasing functions in Bayesian language. In frequentist language this means that the hypothesis class $\mathcal{H}$ only contains monotonically increasing functions. In both cases the resulting uUBs are monotonically increasing too, which we next prove in Propositions 1 and 2.

Proposition 1. *Let $\mathbb{P}[f \text{ is monotonically increasing}] = 1$ according to the prior and let $uUB_\alpha(x) := \inf\{y \in \mathbb{R} : \mathbb{P}\left[f(x) \leq y | D^{train}\right] \geq \alpha\}$ $\forall x \in X$ be the α-credible upper bound (i.e., the α-quantile of the posterior distribution of $f(x)$).[13] Then it holds that $uUB_\alpha(x)$ is monotonically increasing (i.e., $x \leq \tilde{x} \implies uUB_\alpha(x) \leq uUB_\alpha(\tilde{x})$).*

Proof. Let $x \leq \tilde{x}$. For a shorter notation we write "f is (M)" instead of "f is monotonically increasing" and we define $\tilde{\mathcal{V}} := \{f \in \mathbb{R}^X : f \text{ is (M)}\}$. Then, from the definition of monotonicity, it follows for every $y \in \mathbb{R}$ that[14]

$$\begin{aligned} &\{f : f(x) > y, f \text{ is (M)}\} \subseteq \{f : f(\tilde{x}) > y, f \text{ is (M)}\} \\ \iff &\{f : f(x) > y\} \setminus \tilde{\mathcal{V}}^c \subseteq \{f : f(\tilde{x}) > y\} \setminus \tilde{\mathcal{V}}^c \\ \iff &\mathbb{P}\left[\{f : f(x) > y\} | D^{\text{train}}\right] \leq \mathbb{P}\left[\{f : f(\tilde{x}) > y | D^{\text{train}}\}\right] \\ \iff &\mathbb{P}\left[\{f : f(x) \leq y\} | D^{\text{train}}\right] \geq \mathbb{P}\left[\{f : f(\tilde{x}) \leq y | D^{\text{train}}\}\right] \end{aligned}$$

Since $\text{uUB}_\alpha(x)$ is the infimum, we know that for every $y < \text{uUB}_\alpha(x)$:

$$\mathbb{P}\left[\{f : f(\tilde{x}) \leq y | D^{\text{train}}\}\right] \leq \mathbb{P}\left[\{f : f(x) \leq y\} | D^{\text{train}}\right] < \alpha.$$

This means that every $y < \text{uUB}_\alpha(x)$ is too small to be equal to $\text{uUB}_\alpha(\tilde{x})$, thus $\text{uUB}_\alpha(x) \leq \text{uUB}_\alpha(\tilde{x})$. □

Proposition 2. *Let $\mathcal{H}$ be a hypothesis class that only contains monotonically increasing functions and let $uUB_{\mathcal{H}}(x) := \sup\{f(x) : f \in \mathcal{H}_{D^{train}}\}$ be an uUB, then $uUB_{\mathcal{H}}$ is monotonically increasing (i.e., $x \leq \tilde{x} \implies uUB_{\mathcal{H}}(x) \leq uUB_{\mathcal{H}}(\tilde{x})$).*

Proof. Let $x \leq \tilde{x}$, then $\sup\{f(x) : f \in \mathcal{H}_{D^{\text{train}}}\} \leq \sup\{f(\tilde{x}) : f \in \mathcal{H}_{D^{\text{train}}}\}$, since $\forall f \in \mathcal{H}_{D^{\text{train}}} : f(x) \leq f(\tilde{x})$. □

Monotone Value Functions Do Not Imply Monotone Uncertainty. However, the model uncertainty $\hat{\sigma}_f$ (defining the width of the UBs) is *not* monotonic at all. For already observed training input points there is zero model uncertainty while smaller unobserved input points can have much bigger model uncertainty.

[12] Within this paper "monotonically increasing" should always be interpreted as "monotonically non-decreasing".

[13] In the literature, $\mathbb{P}\left[f(x) \leq y | D^{\text{train}}, x\right]$ is often used instead of $\mathbb{P}\left[f(x) \leq y | D^{\text{train}}\right]$ which is equal for every given $x \in X$. In both notations, f is seen as a random variable in function space.

[14] For the second equivalence we use that for every measurable set A, we obtain $\mathbb{P}[A] - 0 = \mathbb{P}[A] - \mathbb{P}[\mathcal{V}^c] \leq \mathbb{P}[A \setminus \mathcal{V}^c] \leq \mathbb{P}[A]$, thus $P[A \setminus \mathcal{V}^c] = \mathbb{P}[A]$.

If one would simply use the prior knowledge about the monotonicity to improve the mean prediction (by assuring its monotonicity) but then estimate the uUB by simply adding a standard (e.g., original NOMU or Gaussian processes) estimator for the (scaled) model uncertainty $c \cdot \hat{\sigma}_f$ to the mean prediction to obtain an uUB, one would obtain non-monotonic uUBs violating Propositions 1 and 2.

If one would simply use ensemble methods (Lakshminarayanan, Pritzel, and Blundell 2017; Gal and Ghahramani 2016) with MVNNs as ensemble members the uUBs obtained from the formula given in (Lakshminarayanan, Pritzel, and Blundell 2017; Gal and Ghahramani 2016) (restated in (Heiss et al. 2022) for the noiseless case) would also lead to non-monotonic uUBs violating Propositions 1 and 2. In theory, this problem could be circumvented by directly using a certain quantile of the ensembles as uUB instead of calculating uUBs based on empirical mean and variance of the ensemble. However, even the 100%-quantile of the ensemble (the point-wise maximum of the ensemble predictions) would often not sufficiently capture enough uncertainty, since Heiss et al. (2022, Remark B.5) empirically showed that the uncertainty of ensemble methods often needs to be scaled up by a very big factor to capture enough uncertainty. Moreover, Kuleshov, Fenner, and Ermon (2018) empirically showed that the uncertainty needs to be calibrated to achieve good results. To summarize, ensemble methods of monotonic functions have the problem that their calibration is limited or calibrating them results in non-monotonic uUBs. Note that the calibration method from Kuleshov, Fenner, and Ermon (2018) cannot solve this problem. Furthermore, in the ICA setting of this paper, optimizing the acquisition function (i.e., the ML-based WDP) based on the uUBs obtained from deep ensembles (Lakshminarayanan, Pritzel, and Blundell 2017) would be computationally too expensive.

However, for our proposed uUB $\mathcal{M}_i^{\text{uUB}}$ the uncertainty can be calibrated by varying π_{exp} without sacrificing monotonicity of the uUB. In the limit $\pi_{\text{exp}} \to 0$ one would just obtain the mean prediction and in the limit $\pi_{\text{exp}} \to \infty$ in relation to explicit and implicit regularization one would obtain the 100%-uUB as solution to our optimization problem.

Linear Skip Connections We have a hyper-parameter (that is part of our HPO) that decides whether we add a trainable linear connection directly from the input to the output for $\mathcal{M}_i^{\text{uUB}}$ and $\mathcal{M}_i^{\text{mean}}$ as formally defined in Definition F.1.

No Connections Between the Two Sub-Architectures The architecture suggested in Heiss et al. (2022) contains connections from the mean-sub-architecture to the uncertainty-sub-architecture (the dashed lines in (Heiss et al. 2022, Figure 2)). In our architecture (see Figure 1) we do not use such connections for two reasons:

1. Solving the WDP via the MILP given in Theorem 2 is computationally much faster because we can completely ignore $\mathcal{M}_i^{\text{mean}}$ for the MILP formulation.
2. In the case of Heiss et al. (2022), without these connections, the uncertainty would be completely independent from the mean prediction (or from the labels y^{train}), prohibiting (Heiss et al. 2022, Desiderata D4). However, our architecture directly outputs the uUB instead of the uncertainty $\hat{\sigma}_f$, thus $\mathcal{M}_i^{\text{uUB}}$ is automatically not independent from the labels $y^{(l)} = \hat{v}_i(x^{(l)})$.

D.3 ICA-Based New NOMU Loss

Definition D.1 (Smooth L1 Loss)**.** *The smooth L1 loss $L_1^\beta : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ with threshold parameter $\beta \geq 0$ is defined as follows:*

$$L_1^\beta(x, y) = \begin{cases} \frac{0.5}{\beta} \cdot (x-y)^2, & |x-y| \leq \beta \\ |x-y| - 0.5 \cdot \beta, & \textit{otherwise}. \end{cases} \tag{33}$$

Definition D.2 (Exponential Linear Unit (ELU))**.** *The ELU function $ELU : \mathbb{R} \to \mathbb{R}$ (with default parameter $\alpha = 1$) is defined as follows:*

$$ELU(x) = \begin{cases} x, & x \geq 0 \\ 1 \cdot (\exp(x) - 1), & x < 0. \end{cases} \tag{34}$$

Definition D.3 (Detailed ICA-based New NOMU Loss)**.** *Let $\pi = (\pi_{sqr}, \pi_{exp}, c_{exp}, \overline{\pi}, \underline{\pi}) \in \mathbb{R}_+^5$ be a tuple of hyperparameters. For a training set D^{train}, L^π is defined as*

$$L^\pi(\mathcal{M}_i^{uUB}) := \pi_{sqr} \sum_{l=1}^{n^{train}} L_1^\beta \left(\mathcal{M}_i^{uUB}(x^{(l)}), y^{(l)} \right) \tag{35a}$$

$$+ \pi_{exp} \int_{[0,1]^m} g \left(0.01 - c_{exp} \left(\min\{\mathcal{M}_i^{uUB}(x), \mathcal{M}_i^{100\%\text{-}uUB}(x)\} - \mathcal{M}_i^{mean}(x) \right) \right) dx \tag{35b}$$

$$+ \pi_{exp} c_{exp} \overline{\pi} \int_{[0,1]^m} L_1^\beta \left((\mathcal{M}_i^{uUB}(x) - \mathcal{M}_i^{100\%\text{-}uUB}(x))^+ \right) dx \tag{35c}$$

$$+ \pi_{exp} c_{exp} \underline{\pi} \int_{[0,1]^m} L_1^\beta \left((\mathcal{M}_i^{mean}(x) - \mathcal{M}_i^{uUB}(x))^+ \right) dx \tag{35d}$$

$$+ \pi_{sqr} \sum_{l=1}^{n^{train}} 0.001 \left(\mathcal{M}_i^{uUB}(x^{(l)}) - y^{(l)} \right)^+ + 0.5 L_1^\beta \left((\mathcal{M}_i^{uUB}(x^{(l)}) - y^{(l)})^+, 0 \right) \tag{35e}$$

where L_1^β is the smooth L1-loss with threshold β (see Definition D.1), $(\cdot)^+$ the positive part, and $g := 1 + ELU$ is convex monotonically increasing with ELU being the exponential linear unit (ELU) *(see Definition D.2).*

Detailed interpretations of all five terms (including (35e) which was added to slightly improve the numerical stability) are as follows:

(35a) enforces that $\mathcal{M}_i^{\text{uUB}}$ fits through the training data.

(35b) pushes $\mathcal{M}_i^{\text{uUB}}$ up as long as it is below the 100%-uUB $\mathcal{M}_i^{\text{100\%-uUB}}$. This force gets weaker the further $\mathcal{M}_i^{\text{uUB}}$ is above the mean $\mathcal{M}_i^{\text{mean}}$ (especially if c_{exp} is large). π_{exp} controls the overall strength of (9b) and c_{exp} controls how fast this force increases when $\mathcal{M}_i^{\text{uUB}} \to \mathcal{M}_i^{\text{mean}}$. Thus, increasing π_{exp} increases the uUB and increasing c_{exp} increases the uUB in regions where it is close to the mean. Weakening (9b) (i.e., $\pi_{\text{exp}} c_{\text{exp}} \to 0$) leads $\mathcal{M}_i^{\text{uUB}} \approx \mathcal{M}_i^{\text{mean}}$. Strengthening (9b) by increasing $\pi_{\text{exp}} c_{\text{exp}}$ in re-

lation to regularization[15] leads $\mathcal{M}_i^{\text{uUB}} \approx \mathcal{M}_i^{100\%\text{-uUB}}$. In practice, we obtain reasonable approximations to $\alpha\%$-uUB with $\alpha \in [50, 100]$ depending on the value of $\pi_{\text{exp}} c_{\text{exp}}$ in relation to regularization.

(35c) enforces that $\mathcal{M}_i^{\text{uUB}} \leq \mathcal{M}_i^{100\%\text{-uUB}}$. In theory, (9c) would be redundant in the limit $\pi_{\text{sqr}} \to \infty$, because $\mathcal{M}_i^{\text{uUB}} \in \mathcal{V}_{D^{\text{train}}}$. The strength of this term is determined by $\overline{\pi} \cdot (\pi_{\text{exp}} c_{\text{exp}})$, where $\overline{\pi}$ is the (9c)-specific hyperparameter and $\pi_{\text{exp}} c_{\text{exp}}$ adjusts the strength of (9c) to (9b).

(35d) enforces $\mathcal{M}_i^{\text{uUB}} \geq \mathcal{M}_i^{\text{mean}}$. In theory, one should take the limit $\underline{\pi} \to \infty$. However, in practice, a moderate value of $\underline{\pi}$ is numerically more stable and typically enforces that $\mathcal{M}_i^{\text{uUB}} \geq \mathcal{M}_i^{\text{mean}}$. The interpretation of $\underline{\pi}$ and $\pi_{\text{exp}} c_{\text{exp}}$ is analogous to (9c).

(35e) is an asymmetric version of (35a) for numerical stability. Since (35b) pushes $\mathcal{M}_i^{\text{uUB}}$ for all $x \in \mathcal{X}$ upwards, $\mathcal{M}_i^{\text{uUB}}$ would have a tendency to give slightly too high predictions for training data points, but (35e) compensates this effect for slightly improved numerical stability. In theory (35e) would be redundant in the limit $\pi_{\text{sqr}} \to \infty$.

As in (Heiss et al. 2022), in the implementation of L^{π}, we approximate Equations (35b) to (35d) (in the main paper Equations (9b) to (9d)) via MC-integration using additional, *artificial input points* $D^{\text{art}} := \left\{x^{(l)}\right\}_{l=1}^{n^{\text{art}}} \overset{i.i.d.}{\sim} \text{Unif}([0,1]^m)$, where we sample new artificial input points for each batch of our mini-batch gradient descent based training algorithm.

Furthermore, note that in practice we train $\mathcal{M}_i^{\text{mean}}$ and $\mathcal{M}_i^{\text{uUB}}$ simultaneously but where $\mathcal{M}_i^{\text{mean}}$ is *detached* (using `torch.tensor.detach()`) in the loss L^{π} such that L^{π} does not influence the mean MVNN $\mathcal{M}_i^{\text{mean}}$.

E Parameter Initialization for MVNNs

In this section, we provide more details on our new parameter initialization method for MVNNs (see Section 3.2). Specifically, we give recommendations on how to set the hyperparameters of our proposed mixture distribution depending on the architecture size of the considered MVNN.

E.1 Theoretical Results

Definition E.1 (Uniform Mixture Distribution)**.** *We define the probability density g_{A_k,B_k,p_k} of an initial weight $W_{j,l}^{i,k}$ corresponding to an MVNN as*[16]

$$g_{A_k,B_k,p_k}(w) = \frac{1-p_k}{A_k} \mathbb{1}_{[0,A_k]}(w) + \frac{p_k}{B_k} \mathbb{1}_{[0,B_k]}(w), \tag{36}$$

which corresponds to a mixture distribution of Unif$[0, A_k]$ *and* Unif$[0, B_k]$.

We assume that for each layer k, we independently sample all weights $W_{jl}^{i,k}$ i.i.d. according to the distribution in Definition E.1.

We then obtain that

$$\mu_k = \mathbb{E}\left[W_{j,l}^{i,k}\right] = (1-p_k)\frac{A_k}{2} + p_k \frac{B_k}{2}. \tag{37}$$

Moreover, by using that the variance of an Unif$[0, c]$-distributed random variable is $\frac{c^2}{12}$ we get that

$$\sigma_k^2 = \mathbb{V}\left[W_{j,l}^{i,k}\right] = (1-p_k)\frac{A_k^2}{3} + p_k \frac{B_k^2}{3} - \mu_k^2. \tag{38}$$

In the following, we provide a rule how to set for each layer k, A_k, B_k and p_k depending on the dimension $d^{i,k-1}$ of the previous $(k-1)^{\text{st}}$ layer.

This rule ensures that the conditional expectation and the conditional variance are equal to two constants $\mathbb{E}^{\text{init}}$, and $\mathbb{V}^{\text{init}}$, which are independent of the dimension of the previous layer $d^{i,k-1}$. More formally, let $z^{i,k-1} \in \mathbb{R}^{d^{i,k-1}}$ be the output of the neurons in the $(k-1)^{\text{st}}$ layer, then we set A_k, B_k and p_k such that[17]

$$\mathbb{E}\left[\left(W^{i,k} z^{i,k-1}\right)_j + b_j^{i,k} \middle| z^{i,k-1} = (1,\dots,1)^\top\right] = \mathbb{E}^{\text{init}} \tag{39a}$$

$$\mathbb{V}\left[\left(W^{i,k} z^{i,k-1}\right)_j \middle| z^{i,k-1} = (1,\dots,1)^\top\right] = \mathbb{V}^{\text{init}}. \tag{39b}$$

Lemma 1. *Equivalent formulations of* (39a) *are*

$$\begin{aligned}(39a) &\Leftrightarrow \mathbb{E}\left[\left(W^{i,k}(1,\dots,1)^\top\right)_j + b_j^{i,k}\right] = \mathbb{E}^{init} \\ &\Leftrightarrow \mathbb{E}\left[\left(W^{i,k} z^{i,k-1}\right)_j + b_j^{i,k} \middle| \overline{z^{i,k-1}} = 1\right] = \mathbb{E}^{init},\end{aligned}$$

where $\overline{z^{i,k-1}} = \frac{1}{d^{i,k-1}} \sum_{l=1}^{d^{i,k-1}} z_l^{i,k-1}$. *And equivalent formulations of* (39b) *are*

$$\begin{aligned}(39b) &\Leftrightarrow \mathbb{V}\left[\left(W^{i,k}(1,\dots,1)^\top\right)_j\right] = \mathbb{V}^{init} \\ &\Leftrightarrow \mathbb{V}\left[\left(W^{i,k} z^{i,k-1}\right)_j \middle| z^{i,k-1}\right] = \mathbb{V}^{init} \overline{(z^{i,k-1})^2},\end{aligned}$$

where $\overline{(z^{i,k-1})^2} = \frac{1}{d^{i,k-1}} \sum_{l=1}^{d^{i,k-1}} \left(z_l^{i,k-1}\right)^2$.

Proof. The first and the third equivalence are trivial. The second equivalence can be seen as:

$$\begin{aligned}&\mathbb{E}\left[\left(W^{i,k} z^{i,k-1}\right)_j + b_j^{i,k} \middle| \overline{z^{i,k-1}} = 1\right] = \\ &= \sum_{l=1}^{d^{i,k-1}} \mathbb{E}\left[W_{jl}^{i,k} z_l^{i,k-1} \middle| \overline{z^{i,k-1}} = 1\right] + \mathbb{E}\left[b_j^{i,k}\right]\end{aligned}$$

[15] Regularization can be early stopping or a small number of neurons (implicit) or L2-regularization on the parameters (explicit). The same principle would also hold true if one uses other forms of regularization such as L1-regularization or dropout.

[16] The bidder index i is not relevant in this section and we assume that each bidder i has the same architecture. If bidders would have different layer widths $d^{i,k-1}$, then all the values $A_k, B_k, P_k, \mu_k, \sigma_k$ would depend on the bidder index i too, i.e., $A_{i,k}, B_{i,k}, P_{i,k}, \mu_{i,k}, \sigma_{i,k}$.

[17] Note that $\mathbb{E}^{\text{init}} =$ (39a) (`init_E` in our code) includes already the bias, while $\mathbb{V}^{\text{init}} =$ (39b) (`init_V` in our code) does not include the bias. But this is a simple additive term that could easily be adjusted. One could easily include the bias in both or in none of them to make it more consistent.

Using that $W_{jl}^{i,k}$ and $z^{i,k}$ are independent, we can further simplify this term as

$$
\begin{aligned}
&= \sum_{l=1}^{d^{i,k-1}} \mathbb{E}\left[z_l^{i,k-1} \middle| \overline{z^{i,k-1}} = 1\right] \mathbb{E}\left[W_{jl}^{i,k}\right] + \mathbb{E}\left[b_j^{i,k}\right] \\
&= \mathbb{E}\left[W_{j1}^{i,k}\right] \sum_{l=1}^{d^{i,k-1}} \mathbb{E}\left[z_l^{i,k-1} \middle| \overline{z^{i,k-1}} = 1\right] + \mathbb{E}\left[b_j^{i,k}\right] \\
&= \mathbb{E}\left[W_{j1}^{i,k}\right] \mathbb{E}\left[\sum_{l=1}^{d^{i,k-1}} z_l^{i,k-1} \middle| \overline{z^{i,k-1}} = 1\right] + \mathbb{E}\left[b_j^{i,k}\right] \\
&= \mathbb{E}\left[W_{j1}^{i,k}\right] d^{i,k-1} + \mathbb{E}\left[b_j^{i,k}\right] \\
&= \mathbb{E}\left[\left(W^{i,k}(1,\dots,1)^\top\right)_j + b_j^{i,k}\right].
\end{aligned}
$$

Finally, the last equivalence can be seen as

$$
\begin{aligned}
&\mathbb{V}\left[\left(W^{i,k} z^{i,k-1}\right)_j \middle| z^{i,k-1}\right] = \\
&= \sum_{l=1}^{d^{i,k-1}} \mathbb{V}\left[W_{jl}^{i,k} z_l^{i,k-1} \middle| z^{i,k-1}\right] \\
&= \sum_{l=1}^{d^{i,k-1}} \mathbb{V}\left[W_{jl}^{i,k}\right] \left(z_l^{i,k-1}\right)^2 \\
&= d^{i,k-1} \mathbb{V}\left[W_{j1}^{i,k}\right] \frac{1}{d^{i,k-1}} \sum_{l=1}^{d^{i,k-1}} \left(z_l^{i,k-1}\right)^2 \\
&= \mathbb{V}\left[\left(W^{i,k}(1,\dots,1)^\top\right)_j\right] \overline{\left(z^{i,k-1}\right)^2}.
\end{aligned}
$$

□

The solution to problem (39) is not unique, but we propose three hyperparameters $B^{\text{init}}, b^{\text{init}}, \epsilon \in \mathbb{R}_+$ to characterize one specific solution that has multiple nice properties (see Theorem 3).

Definition E.2 (Scaling Rule)**.** *For any choice of hyperparameters $\mathbb{E}^{init}, \mathbb{V}^{init}, B^{init}, b^{init}, \epsilon \in \mathbb{R}_+$ we propose the following values for B_k, A_k and p_k (for a shorter notation we write d instead of $d^{i,k-1} \in \mathbb{N}_+$):*

$$
B_k = \begin{cases} \max\left(\frac{3\mathcal{M}_k^2 + 3d\mathbb{V}^{init}}{2\mathcal{M}_k d} + \frac{\epsilon}{d}, B^{init}\right) & , d > \frac{\mathcal{M}_k^2}{3\mathbb{V}^{init}} \\ \frac{2}{d}\mathcal{M}_k & , \textit{else} \end{cases} \tag{40}
$$

$$
p_k = \begin{cases} 1 - \frac{B_k^2 d^2 - 4B_k\mathcal{M}_k d + 4\mathcal{M}_k^2}{B_k^2 d^2 - 4B_k\mathcal{M}_k d + 3\mathcal{M}_k^2 + 3d\mathbb{V}^{init}} & , d > \frac{\mathcal{M}_k^2}{3\mathbb{V}^{init}} \\ 1 & , \textit{else} \end{cases} \tag{41}
$$

$$
A_k = \begin{cases} \frac{2\mathcal{M}_k - B_k d p_k}{d(1-p_k)} & , d > \frac{\mathcal{M}_k^2}{3\mathbb{V}^{init}} \\ 0 & , \textit{else}, \end{cases} \tag{42}
$$

$$
W_{j,l}^{i,k} \overset{i.i.d.}{\sim} g_{A_k, B_k, p_k} \tag{43}
$$

$$
b_j^{i,k} \overset{i.i.d.}{\sim} \text{Unif}[-b^{init}, 0] \tag{44}
$$

where $\mathcal{M}_k = \mathbb{E}^{init} + \frac{b^{init}}{2}$ and where g_{A_k,B_k,p_k} is the mixture distribution described in Definition E.1 and where all weights and biases are sampled independently.

Finally, when using the scaling rule from Definition E.2, we can prove the following theorem.

Theorem 3. *For any choice of hyperparameters $\mathbb{E}^{init}, \mathbb{V}^{init}, B^{init}, b^{init}, \epsilon \in \mathbb{R}_+$, for every $k \in \{1, \dots, K_i\}$, for every dimension $d^{i,k-1} \in \mathbb{N}_+$ it holds that if we sample according to the distribution described in Definition E.2, we obtain that:*[18]

1. (39a) *holds,*
2. $\mathbb{V}\left[\left(W^{i,k} z^{i,k-1}\right)_j \middle| z^{i,k-1} = (1,\dots,1)^\top\right] \geq \mathbb{V}^{init}$ *holds,*
3. (39b) *holds if* $d^{i,k-1} > \frac{\mathcal{M}_k^2}{3\mathbb{V}^{init}}$,
4. $\mathbb{P}\left[W_{j,l}^{i,k} = 0\right] = 0$ *if* $\epsilon > 0$,
5. $\mathbb{P}\left[W_{j,l}^{i,k} = 0\right] = 1 - p_k$ *if* $\epsilon = 0$ *and* $B^{init} < B_k$,
6. $\mathbb{P}\left[W_{j,l}^{i,k} < 0\right] = 0$,
7. $\mathbb{P}\left[W_{j,l}^{i,k} > B_k\right] = 0$,
8. $B_k \geq \frac{3\mathbb{V}^{init}}{2\mathcal{M}_k}$,
9. $\lim_{d^{i,k-1}\to\infty} B_k = \max\left\{\frac{3\mathbb{V}^{init}}{2\mathcal{M}_k}, B^{init}\right\}$.

Proof. First, note that by Lemma 1

$$
\begin{aligned}
\text{(39a)} &\Leftrightarrow \mathbb{E}\left[\left(W^{i,k}(1,\dots,1)^\top\right)_j + b_j^{i,k}\right] = \mathbb{E}^{\text{init}} \\
&\Leftrightarrow \mathbb{E}\left[\left(W^{i,k}(1,\dots,1)^\top\right)_j\right] = \mathbb{E}^{\text{init}} - \mathbb{E}\left[b_j^{i,k}\right] \\
&\Leftrightarrow d^{i,k-1}\mathbb{E}\left[W_{j,l}^{i,k}\right] = \mathbb{E}^{\text{init}} + \frac{b^{\text{init}}}{2} \\
&\Leftrightarrow \mathbb{E}\left[W_{j,l}^{i,k}\right] = \frac{\mathcal{M}_k}{d^{i,k-1}}.
\end{aligned}
$$

We prove this in section "Proof of item 1" of the MATHEMATICA SCRIPT[19], where we use the notation `d` $:= d^{i,k-1}$, `M` $:= \mathcal{M}_k$, `V` $:= \mathbb{V}^{\text{init}}$, `epsilon` $:= \epsilon$, `Binit` $:= B^{\text{init}}$, `Bchoice` $:= B_k$ given in (40), `pchoice` $:= p_k$ given in (41), `Achoice` $:= A_k$ given in (42) and `EofW` $:= \mathbb{E}\left[W_{j,l}^{i,k}\right]$ computed in (37). This proves item 1.

[18] We assume that $\mathbb{E}^{\text{init}}$, $\mathbb{V}^{\text{init}}$ and d are strictly positive. However, $B^{\text{init}}, b^{\text{init}}$ and ϵ could in theory also be set to 0 and Theorem 3 is formulated such that it would still hold true for $B^{\text{init}}, b^{\text{init}}, \epsilon \in \mathbb{R}_+ \cup \{0\}$.

[19] You can open the MATHEMATICA SCRIPT including the results in your web-browser (https://www.wolframcloud.com/obj/jakob.heiss/Published/MVNN_initialization_V1_4.nb) without the need to install anything on your computer. (The interactive plot might only work if you open the script with an installed version of Mathematica, but it is not necessary for the proof.) We use a computer algebra system to make it much more convenient to check the correctness of our proof. Some of the terms that appear in this proof are already very long as one can see for example in section "Proof of item 2 and 3" in our MATHEMATICA SCRIPT. Simplifying these terms by hand would take multiple pages of very tedious calculations, which would be highly prone to typos and other mistakes. Within our MATHEMATICA SCRIPT we only use exact symbolic methods and no numerical approximations (except for the visualizations).

We manipulate using Lemma 1

$$
\begin{aligned}
(39b) &\Leftrightarrow \mathbb{V}\left[\left(W^{i,k}(1,\dots,1)^\top\right)_j\right] = \mathbb{V}^{\text{init}} \\
&\Leftrightarrow d^{i,k-1}\mathbb{V}\left[W^{i,k}_{j,l}\right] = \mathbb{V}^{\text{init}} \\
&\Leftrightarrow \mathbb{V}\left[W^{i,k}_{j,l}\right] = \frac{\mathbb{V}^{\text{init}}}{d^{i,k-1}}.
\end{aligned}
$$

We prove this in section "Proof of item 2 and 3" of the MATHEMATICA SCRIPT with the additional notation $\texttt{VofW} := \mathbb{V}\left[W^{i,k}_{j,l}\right]$ computed in (38). This section proves items 2 and 3.

To show item 4 it is sufficient to show $A_k > 0$ (because of $A_k \leq B_k$, as shown in section "Proof of item 7: ($\texttt{A}\leq\texttt{B}$)") or ($p_k = 1$ and $B_k > 0$). If $d^{i,k-1} > \frac{\mathcal{M}_k^2}{3\mathbb{V}^{\text{init}}}$, we show $A_k > 0$ in section "Proof of item 4" of the MATHEMATICA SCRIPT. In the other case $p_k = 1$ and $B_k > 0$ as one can directly see from (40) and (41).

To show item 5, it is sufficient to show that $A_k = 0$. Section "Proof of item 5" of the MATHEMATICA SCRIPT shows in the case $d > \frac{\mathcal{M}_k^2}{3\mathbb{V}^{\text{init}}}$ that $A_k = 0 \iff (\epsilon = 0$ and $B^{\text{init}} \leq \frac{3\mathcal{M}_k^2+3d\mathbb{V}^{\text{init}}}{2\mathcal{M}_k d})$. By definition this statement is equivalent to $A_k = 0 \iff B_k = \frac{3\mathcal{M}_k^2+3d\mathbb{V}^{\text{init}}}{2\mathcal{M}_k d}$. Finally, since $(\epsilon = 0$ and $B^{\text{init}} < B_k) \implies B_k = \frac{3\mathcal{M}_k^2+3d\mathbb{V}^{\text{init}}}{2\mathcal{M}_k d}$, item 5 holds true. In the case $d^{i,k-1} \leq \frac{\mathcal{M}_k^2}{3\mathbb{V}^{\text{init}}}$, $A_k = 0$ (i.e., item 5) follows directly from (42). Note that $B_k > 0$ always holds true, since we always assume $\mathcal{M}_k > 0$ and $d > 0$.

Item 6 follows directly from Definition E.1 and the fact that $A_k \geq 0$ and $B_k \geq 0$ (see section "Proof of item 6" in the MATHEMATICA SCRIPT).

Item 7 follows directly from Definition E.1 and $A_k \leq B_k$ (see section "Proof of item 7: ($\texttt{A}\leq\texttt{B}$)" in the MATHEMATICA SCRIPT).

Item 8 is shown for the two cases in section "Proof of item 8" of the MATHEMATICA SCRIPT.

Item 9 is shown in section "Proof of item 9" in our MATHEMATICA SCRIPT. □

Discussion of Theorem 3. Items 1 and 3 in Theorem 3 tell us that our initialization scheme actually solves problem (39) if $d^{i,k-1} > \frac{\mathcal{M}_k^2}{3\mathbb{V}^{\text{init}}}$. Note that problem (39) does not have any solution for for $d^{i,k-1} \leq \frac{\mathcal{M}_k^2}{3\mathbb{V}^{\text{init}}}$ (as can be seen in the first 5 lines of our MATHEMATICA SCRIPT with the notation explained in the proof of Theorem 3). However, items 1 and 2 in Theorem 3 tell us that we still have a reasonable initialization scheme for $d^{i,k-1} \leq \frac{\mathcal{M}_k^2}{3\mathbb{V}^{\text{init}}}$ that solves the relaxed problem of items 1 and 2. In practice solutions to this relaxed problem are still fine, since it also prevents us from exploding expectation or vanishing variance. Too much variance in the initialization is less of a problem.[20]

[20] Alternatively one could consider to relax problem (39) in the other direction by allowing smaller expectation instead of bigger variance, which would also be fine in practice. A solution to this alternative relaxed problem could be simply achieved by changing the definition of B_k in the "else"-case of (40).

Items 4 and 5 motivate to choose $\epsilon > 0$ to prevent weights being initialized to zero, which can lead to bad local minima.

Item 6 guarantees that our network is actually a valid MVNN at initialization fulfilling the non-negativity constraints of the weights at initialization.[21]

Items 7 to 9 give us guarantees on the upper bound of the weights.

E.2 Recommended Hyperparameter Choices

In this section, we provide intuition about each hyperparameter $\mathbb{E}^{\text{init}}, \mathbb{V}^{\text{init}}, B^{\text{init}}, b^{\text{init}}, \epsilon \in \mathbb{R}_+$ and recommendations on how to set them in practice.

1. **Parameter $\mathbb{E}^{\text{init}}$:** (`init_E` in our code) gives the conditional expectation (39a) of a pre-activated neuron (including bias) conditioned on $z^{i,k-1} = 1$ (see item 1 in Theorem 3 and Lemma 1). This corresponds to an upper bound of the expected value of the MVNN $\mathbb{E}\left[\mathcal{M}_i^\theta((1,...,1))\right]$ (i.e., the predicted value of the full bundle) at initialization of the network, if all cutoffs $t^{i,k}$ of the bReLU activation function are equal to 1. $\mathbb{E}^{\text{init}}$ is approximately equal to $\mathbb{E}\left[\mathcal{M}_i^\theta((1,...,1))\right]$ if $\mathbb{E}^{\text{init}} \geq 1$.
   If you normalize the data such that the full bundle has always value 1 (i.e., $\hat{v}_i((1,...,1)) = 1$), setting $\mathbb{E}^{\text{init}} = 1$ is our recommended choice. If you choose to initialize the bReLU cutoffs $t^{i,k}$ i.i.d. uniformly at random, i.e., $t^{i,k} \sim \text{Unif}(0,1)$, then $\mathbb{E}^{\text{init}} \in [1,2]$ is recommended, because in this case $\frac{\mathbb{E}^{\text{init}}}{2}$ is an upper bound for the expectation of the pre-activated values[22] $o_j^{i,k}$ of any neuron in the MVNN at initialization for $k > 1$. This can be seen as follows:

$$
\begin{aligned}
\mathbb{E}\left[o_j^{i,k}\right] &\overset{\text{Lemma 1}}{=} (\mathbb{E}^{\text{init}} - \mathbb{E}\left[b_j^{i,k}\right])\mathbb{E}\left[z_1^{i,k-1}\right] + \mathbb{E}\left[b_j^{i,k}\right] && (45)\\
&\leq (\mathbb{E}^{\text{init}} - \mathbb{E}\left[b_j^{i,k}\right])\mathbb{E}\left[t^{i,k-1}\right] + \mathbb{E}\left[b_j^{i,k}\right] && (46)\\
&= \frac{\mathbb{E}^{\text{init}} + \frac{b^{\text{init}}}{2}}{2} - \frac{b^{\text{init}}}{2} && (47)\\
&= \frac{\mathbb{E}^{\text{init}}}{2} - \frac{b^{\text{init}}}{4} < \frac{\mathbb{E}^{\text{init}}}{2}, && (48)
\end{aligned}
$$

   where the second inequality follows from the definition of the bRelu activation function. Specifically, this also shows for $j = 1$ and $k = K_i$ that $\frac{\mathbb{E}^{\text{init}}}{2}$ is an upper bound

[21] Note that if we would initialize our network with a weight-distribution that solves problem (39), but does not fulfill Item 6 (e.g., a generic initialization with $\mu_k = 0$ and $\sigma_k \propto \frac{1}{\sqrt{d^{i,k-1}}}$), we would get a valid MVNN after the first gradient step, which projects all the negative weights to zero. However, this almost initial network would have exploding conditional expectations, i.e., almost all neurons would have pre-activations $o_j^{i,k} > 1$ independent of the input training data point and thus one would end up with an almost constant function and zero gradients as shown in Figure 5.

[22] Recall, that $o_j^{i,k}$ is the pre-activated value of the j-th neuron in the k-th layer (of the i-th bidder) including biases.

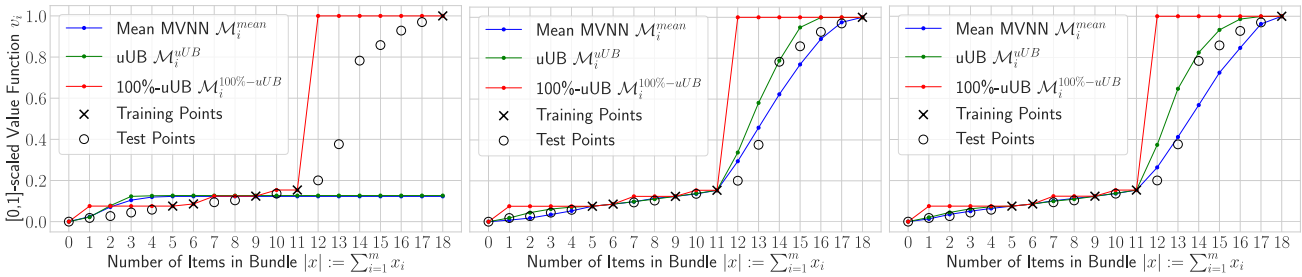


Figure 5: $\mathcal{M}_i^{\text{mean}}$, $\mathcal{M}_i^{\text{uUB}}$ and $\mathcal{M}_i^{\text{100\%-uUB}}$ along an increasing 1D subset-path in LSVM for the national bidder. **Left:** [64,64]-architecture with generic initialization fails. **Middle:** [64,64]-architecture with our initialization works. **Right:** even larger [256,256]-architecture with our initialization still works.

for $\mathbb{E}\left[\mathcal{M}_i^\theta((1,...,1))\right] = o_1^{i,K_i}((1,...,1))$ at initialization.[23] If the values of your MVNN are in a different order of magnitude, you should scale your data in a preprocessing step to $[0,1]$.

2. **Parameter $\mathbb{V}^{\text{init}}$:** (`init_V` in our code) gives the conditional variance (39b) of a pre-activated neuron (without bias) conditional on $z^{i,k-1} = (1,\dots,1)^\top$, if $d^{i,k-1} > \frac{\mathcal{M}_k^2}{3\mathbb{V}^{\text{init}}}$ (see item 3 in Theorem 3). In any case, $\mathbb{V}^{\text{init}}$ is a lower bound for this conditional variance (see item 2 in Theorem 3). Typically, we select $\mathbb{V}^{\text{init}} \in [1/50, 1]$. Choosing $\mathbb{V}^{\text{init}}$ too small yields an almost deterministic network initialization. Since we prefer initial weights that are smaller than one, $\mathbb{V}^{\text{init}}$ should not be chosen too large (i.e., preferably $\frac{3\mathbb{V}^{\text{init}}}{2\mathcal{M}_k} \leq 1$ because of items 7 to 9 in Theorem 3).
3. **Parameter b^{init}:** (`init_bias` in our code[24]) All the biases of the MVNN $b_j^{i,k}$ are sampled uniformly at random from $[-b^{\text{init}}, 0]$ as given in eq. (44). Setting $b^{\text{init}} = 0.05$ is our recommendation, although any other small values would be sufficient. We discourage zero to avoid numerical issues during training.
4. **Parameter B^{init}:** (`init_b` in our code[24]) B^{init} gives a *lower* bound for B_k in the case of large number of neurons (see item 9 in Theorem 3 and eq. (40)). The "big" weights are sampled from $\text{Unif}(0, B_k)$, i.e., B_k is an upper bound for the weights (see item 7 in Theorem 3). Usually, we prefer B_k that are not unnecessary big, but too small B_k leads to almost vanishing A_k (in the case of $d > \frac{\mathcal{M}_k^2}{3\mathbb{V}^{\text{init}}}$).[25] Thus, setting $B^{\text{init}} = 0.05$ is our recommendation (but one could also use any other small value including zero).
5. **Parameter ϵ:** (`init_little_const` in our code) prevents weights from being initialized to zero, i.e., $\epsilon > 0$ guarantees that no initial weight will be *exactly* zero, see item 4 in Theorem 3. Conversely setting $\epsilon = 0$ leads weights to be zero with probability $(1 - p_k)$ (see item 5 in Theorem 3). If one chooses ϵ too large B_k can become too large (see eq. (40)). Thus, setting $\epsilon = 0.1$ is our recommendation (similarly to b^{init} any other small value is also admissible).

In the last section of our MATHEMATICA SCRIPT we provide an interactive plot that shows how $B_k, A_k, p_k, \mathbb{V}\left[o_j^{i,k} - b_j^{i,k} \middle| z^{i,k-1} = (1,\dots,1)^\top\right]$ and $\mathbb{E}\left[o_j^{i,k} \middle| z^{i,k-1} = (1,\dots,1)^\top\right]$ depend on $d^{i,k-1}$ for different choices of our hyperparameters.

E.3 Visualization for Wider MVNNs

In this section, we provide an additional visualization for a wider MVNN that uses our proposed new initialization method.

In Figure 5, we present the results. Figure 5 confirms that our initialization method also properly works for an even larger MVNN-architecture with two hidden layers with 256 neurons per hidden layer. While the problems of the generic initialization methods described in Section 3.2 increase as the number of neurons increase, our initialization method can deal with an arbitrarily large number of neurons.

F MILP

In this section, we provide more details on Section 3.3.

F.1 Proof of Theorem 2

In this section, we provide the proof of Theorem 2.

[23] On the other hand one could argue that one wants the conditional expectation of pre-activated neurons to be smaller because of the smaller cut-offs. However, note especially for small values of $\mathbb{E}^{\text{init}}$, the actual expectation $\mathbb{E}\left[\mathcal{M}_i^\theta((1,...,1))\right]$ at initialization decreases with increasing depth of the network, since the upper bound can loose its tightness as depth increases especially for $\mathbb{E}^{\text{init}} < 1$.

[24] Be careful, when translating the notation of our paper into the notation of our code. B_k is `b` in our code and the biases b are denoted by `bias` in our code. b^{init} does *not* translate to `init_b`.

[25] Section "Proof of item 5" in the MATHEMATICA SCRIPT shows that $A_k = 0$ iff $\epsilon = 0$ and $B^{\text{init}} \leq \frac{3\mathcal{M}_k^2 + 3d\mathbb{V}^{\text{init}}}{2\mathcal{M}_k d}$. This statement is equivalent to $A_k = 0$ iff $B_k = \frac{3\mathcal{M}_k^2 + 3d\mathbb{V}^{\text{init}}}{2\mathcal{M}_k d}$, which is the minimal possible choice for $B_k \in [\frac{3\mathcal{M}_k^2 + 3d\mathbb{V}^{\text{init}}}{2\mathcal{M}_k d}, \infty)$. By continuity $B_k \approx \frac{3\mathcal{M}_k^2 + 3d\mathbb{V}^{\text{init}}}{2\mathcal{M}_k d} \implies A_k \approx 0$. Note that in the other case $d \leq \frac{\mathcal{M}_k^2}{3\mathbb{V}^{\text{init}}}$, $p_k = 1$ and thus A_k is irrelevant for the mixture distribution from Definition E.1.

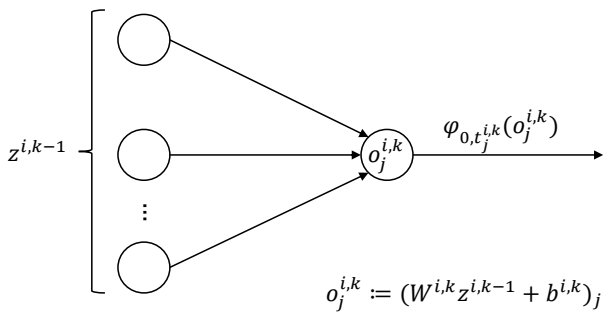


Figure 6: Schematic representation of the jth neuron in the kth layer. Shown are the output $z^{i,k-1}$ of the previous $(k-1)^{\text{st}}$ layer, the pre-activated output $o_j^{i,k}$ of the jth neuron in the kth layer, and the output of the jth neuron of the kth layer after applying bRELU $\varphi_{0,t_j^{i,k}}(o_j^{i,k}) = \min(t_j^{i,k}, \max(0, o_j^{i,k}))$.

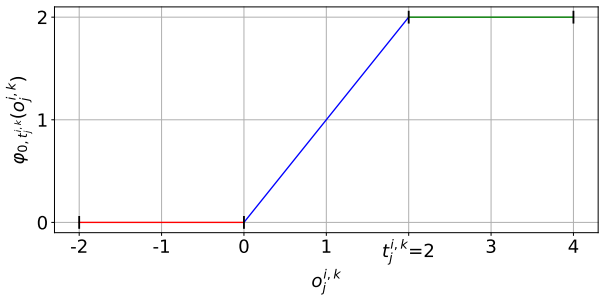


Figure 7: Example plot for agent i, layer k and neuron j of the bReLU activation function $\varphi_{0,t_j^{i,k}}(\cdot) := \min(t_j^{i,k}, \max(0, \cdot))$ (Weissteiner et al. 2022a) with cutoff $t_j^{i,k} = 2$ in the interval $[-2, 4]$. Shown are the pre-activated output $o_j^{i,k}$ of the jth neuron in the kth layer (x-axis) and the output of the jth neuron in the kth layer after applying bRELU (y-axis).

First, we show in Lemma 2 how to encode an arbitrary single hidden MVNN layer into multiple linear constraints. For this fix a bidder $i \in N$ and an arbitrary layer $k \in \{1, \dots, K_i - 1\}$. Recall, that $z^{i,k-1} \in \mathbb{R}^{d^{i,k-1}}$ denotes the output of the previous layer (with $z^{i,0}$ being equal to the input $x \in \mathbb{R}^{d^{i,0}} = \mathbb{R}^m$) and that $o^{i,k} := W^{i,k}z^{i,k-1} + b^{i,k}$ denotes the *pre*-activated output of the kth layer with $l^{i,k} \leq o^{i,k} \leq u^{i,k}$, where the tight lower/upper bound $l^{i,k}$/ $u^{i,k}$ can be computed by forward-propagating the empty/full bundle. Then the following Lemma holds:[26]

Lemma 2. *Consider the following set of linear constraints:*

$$z^{i,k} \leq \alpha^{i,k} \cdot t^{i,k} \tag{49}$$

$$z^{i,k} \leq o^{i,k} - l^{i,k} \cdot (1 - \alpha^{i,k}) \tag{50}$$

$$z^{i,k} \geq \beta^{i,k} \cdot t^{i,k} \tag{51}$$

$$z^{i,k} \geq o^{i,k} + (t^{i,k} - u^{i,k}) \cdot \beta^{i,k} \tag{52}$$

$$\alpha^{i,k} \in \{0,1\}^{d^{i,k}},\ \beta^{i,k} \in \{0,1\}^{d^{i,k}}. \tag{53}$$

Then it holds for the output of the kth layer $\varphi_{0,t^{i,k}}(o^{i,k}) = z^{i,k}$.

Proof. Recall, that $o^{i,k} := W^{i,k}z^{i,k-1} + b^{i,k}$ denotes the pre-activated output of the kth layer. Let $j \in \{1, \dots, d^{i,k}\}$ denote the jth neuron of that layer and let $l_j^{i,k} \leq o_j^{i,k} \leq u_j^{i,k}$, where $l_j^{i,k}$ and $u_j^{i,k}$ are the tight box bounds computed by forward propagating the empty, i.e., $x = (0, \dots, 0)$, and the full, i.e., $x = (1, \dots, 1)$ bundle (see (Weissteiner et al. 2022a, Appendix C.5 and Fact C.1)). Moreover, let $\varphi_{0,t_j^{i,k}}(o_j^{i,k}) = \min(t_j^{i,k}, \max(0, o_j^{i,k}))$ with $t_j^{i,k} \geq 0$ be the output of the jth neuron in the kth layer. In Figure 6, we present a schematic representation for a single neuron. In Figure 7, we present an example of the bReLU activation function.

[26] All vector inequalities should be understood component-wise.

We distinguish the following three exclusive cases:

- **Case 1:** $o_j^{i,k} < 0$, i.e., the red line segment in Figure 7. Per definition it follows that $\varphi_{0,t_j^{i,k}}(o_j^{i,k}) = 0$. Setting $\alpha_j^{i,k} = \beta_j^{i,k} = 0$ in Equations (49) to (53) implies that $z_j^{i,k} = 0$.
- **Case 2:** $o_j^{i,k} \in [0, t_j^{i,k}]$, i.e., the blue line segment in Figure 7. Per definition it follows that $\varphi_{0,t_j^{i,k}}(o_j^{i,k}) = o_j^{i,k}$. Setting $\alpha_j^{i,k} = 1$ and $\beta_j^{i,k} = 0$ in Equations (49) to (53) implies that $z_j^{i,k} = o_j^{i,k}$.
- **Case 3:** $o_j^{i,k} > t_j^{i,k}$, i.e., the green line segment in Figure 7. Per definition it follows that $\varphi_{0,t_j^{i,k}}(o_j^{i,k}) = t_j^{i,k}$ Setting $\alpha_j^{i,k} = \beta_j^{i,k} = 1$ in Equations (49) to (53) implies that $z_j^{i,k} = t_j^{i,k}$.

Thus, in total $z^{i,k} = \varphi_{0,t_j^{i,k}}(o_j^{i,k})$. □

Using Lemma 2, we can now proof Theorem 2 which provides our new and more succinct MILP formulation.

Proof. Consider the ML-based WDP from Equation (10). For each bidder $i \in N$, we first set $z^{i,0}$ equal to the input bundle a_i. Then we proceed by using Lemma 2 for $k = 1$, i.e., we reformulate the output of the 1st layer as the linear Equations (49) to (52). We iterate this procedure until we have reformulated the last hidden layer, i.e, layer $k = K_i - 1$. Doing so for each bidder $i \in N$ yields the desired MILP formulation from Theorem 2. □

F.2 Removing Constraints via Box Bounds

Let $l_j^{i,k} \leq o_j^{i,k} \leq u_j^{i,k}$, where $l_j^{i,k}$ and $u_j^{i,k}$ are the *tight* box bounds computed by forward propagating the empty, i.e., $x = (0, \dots, 0)$, and the full, i.e., $x = (1, \dots, 1)$ bundle (see (Weissteiner et al. 2022a, Appendix C.5 and Fact C.1)).

In the following cases one can remove the constraints and corresponding variables in Lemma 2 and thus also in Theorem 2.

- **Case 1:** $0 \leq t_j^{i,k} < l_j^{i,k} \leq u_j^{i,k}$. Then one can simply set

$$z_j^{i,k} := t_j^{i,k} \tag{54}$$

  and remove the jth components from Equations (49) to (53) of the corresponding layer k ($o_j^{i,k}$ lies for sure in the green line segment in Figure 7).
- **Case 2:** $l_j^{i,k} \leq u_j^{i,k} < 0 \leq t_j^{i,k}$. Then one can simply set

$$z_j^{i,k} := 0 \tag{55}$$

  and remove the jth components from Equations (49) to (53) of the corresponding layer k ($o_j^{i,k}$ lies for sure in the red line segment in Figure 7).
- **Case 3:** $0 \leq l_j^{i,k} \leq u_j^{i,k} \leq t_j^{i,k}$. Then one can simply set

$$z_j^{i,k} := o_j^{i,k} \tag{56}$$

  and remove the jth components from Equations (49) to (53) of the corresponding layer k ($o_j^{i,k}$ lies for sure in the blue line segment in Figure 7).
- **Case 4:** $0 \leq l_j^{i,k} \leq t_j^{i,k} < u_j^{i,k}$. Then one can set

$$\alpha_j^{i,k} := 1 \tag{57}$$

  and only one binary decision variable for the jth neuron of the kth layer remains. ($o_j^{i,k}$ lies for sure in the union of the blue and green line segment in Figure 7).
- **Case 5:** $l_j^{i,k} \leq 0 < u_j^{i,k} \leq t_j^{i,k}$. Then one can set

$$\beta_j^{i,k} := 0 \tag{58}$$

  and only one binary decision variable for the jth neuron of the kth layer remains. ($o_j^{i,k}$ lies for sure in the union of the red and blue line segment in Figure 7).

F.3 MILP for MVNNs with Linear Skip Connection

In this section, we provide a simple extension of the MILP in Theorem 2 for MVNNs with a linear skip connection. First, we define MVNNs with a linear skip connection.

Definition F.1 (MVNN with Linear Skip Connection)**.** *An MVNN with linear skip connection $\mathcal{M}_i^{lskip,\theta} : \mathcal{X} \to \mathbb{R}_+$ for agent $i \in N$ is defined as*

$$\mathcal{M}_i^{lskip,\theta}(x) = W^{i,K_i}\varphi_{0,t^{i,K_i}}\left(\ldots\varphi_{0,t^{i,1}}(W^{i,1}x + b^{i,1})\ldots\right) + W^{i,0}x, \tag{59}$$

- *$K_i + 1 \in \mathbb{N}$ is the number of layers ($K_i - 1$ hidden layers),*
- *$\{\varphi_{0,t^{i,k}}\}_{k=1}^{K_i-1}$ are the MVNN-specific activation functions with cutoff $t^{i,k} > 0$, called* bounded ReLU (bReLU)*:*

$$\varphi_{0,t^{i,k}}(\cdot) := \min(t^{i,k}, \max(0, \cdot)) \tag{60}$$

- *$W^i := (W^{i,k})_{k=0}^{K_i}$ with $W^{i,k} \geq 0$ and $b^i := (b^{i,k})_{k=1}^{K_i-1}$ with $b^{i,k} \leq 0$ are the* non-negative *weights and* non-positive *biases of dimensions $d^{i,k} \times d^{i,k-1}$ (except $W^{i,0}$ which is of dimension $d^{i,K_i} \times d^{i,0}$) and $d^{i,k}$, whose parameters are stored in $\theta = (W^i, b^i)$, where $W^{i,0}$ represents the linear skip connection.*

For the MILP of an MVNN with linear skip connection the only thing that changes is the objective, i.e., one needs to replace Equation (11) in Theorem 2 with

$$\max_{a \in \mathcal{F}, z^{i,k}, \alpha^{i,k}, \beta^{i,k}} \left\{ \sum_{i \in N} W^{i,K_i} z^{i,K_i-1} + W^{i,0} z^{i,0} \right\} \tag{61}$$

G Experiment Details

In this section, we present all details of our experiments from Section 4.

G.1 SATS Domains

In this section, we provide a more detailed overview of the three SATS domains[27], which we use to experimentally evaluate BOCA:

- **Local Synergy Value Model (LSVM)** (Scheffel, Ziegler, and Bichler 2012) has 18 items, 5 *regional* and 1 *national bidder*. Complementarities arise from spatial proximity of items.
- **Single-Region Value Model (SRVM)** (Weiss, Lubin, and Seuken 2017) has 29 items and 7 bidders (categorized as *local*, *high frequency regional*, or *national*) and models large UK 4G spectrum auctions.
- **Multi-Region Value Model (MRVM)** (Weiss, Lubin, and Seuken 2017) has 98 items and 10 bidders (*local*, *regional*, or *national*) and models large Canadian 4G spectrum auctions.

In the efficiency experiments in this paper (i.e., Table 1, Table 3, Table 4, Table 5, and Table 6), we instantiated for each SATS domain the 50 synthetic CA instances with the seeds $\{10001, \ldots, 10050\}$. We used SATS version 0.8.0. All experiments were conducted on a compute cluster running Debian GNU/Linux 10 with Intel Xeon E5-2650 v4 2.20GHz processors with 24 cores and 128GB RAM and Intel E5 v2 2.80GHz processors with 20 cores and 128GB RAM and Python 3.7.10.

G.2 Hyperparameter Optimization

In this section, we provide details on the exact hyperparameter ranges that we used in our HPO. Table 2 shows all hyperparameter ranges that we used. In the following, we explain selected hyperparameters:

- DROPOUT PROBABILITY DECAY FACTOR: After each epoch t the dropout probability for the next epoch $t+1$ p_{drop}^{t+1} is updated as: $p_{\text{drop}}^{t+1} = p_{\text{drop}}^{t} \cdot \kappa$, where κ denotes this factor.

[27] We do not consider the simplest of all SATS domains, i.e., the *Global Synergy Value Model (GSVM)* (Goeree and Holt 2010), since prior work already achieves 0% efficiency loss without integrating any notion of uncertainty (Weissteiner et al. 2022a), and thus GSVM can be seen as already "solved".

Category	Hyperparameter	Range	Log-Uniform Sampling
Data	Number of Training Data Points: $\lvert D^{\text{TRAIN}}\rvert$	LSVM: 50 SRVM: 100 MRVM: 100	
	Number of Test Data Points: $\lvert D^{\text{TEST}}\rvert$	LSVM: 2^{10} SRVM: 2^{13} MRVM: 2^{13}	
	Number of SATS Instances	100	
Generic MVNN	Architecture: (#neurons per hidden layer, #hidden layers)	LSVM: {(96, 1), (32, 2), (16, 3)} SRVM: {(32, 2), (16, 3)} MRVM: {(96, 1), (64, 1), (20, 2)}	
	Linear Skip Connection (see Definition F.1)	{True, False}	
	Batch Size	$\{\lvert D^{\text{TRAIN}}\rvert/4, \lvert D^{\text{TRAIN}}\rvert/2, \lvert D^{\text{TRAIN}}\rvert\}$	
	Epochs	$\{4000\frac{\text{Batch Size}}{\lvert D^{\text{TRAIN}}\rvert}, 4500\frac{\text{Batch Size}}{\lvert D^{\text{TRAIN}}\rvert}, 5000\frac{\text{Batch Size}}{\lvert D^{\text{TRAIN}}\rvert}, \ldots, 8000\frac{\text{Batch Size}}{\lvert D^{\text{TRAIN}}\rvert}\}$	
	Dropout Probability	[0, 0.8]	
	Dropout Probability Decay Factor	[0.75, 1.0]	
Generic Loss	Optimizer	Adam	
	Learning Rate	GSVM: [0.0002, 0.002] LSVM: [0.001, 0.01] SRVM: [0.0008, 0.006] MRVM: [0.0007, 0.004]	Yes
	L2-Regularization: λ (see Equation (9))	[1e-10, 1e-3]	Yes
	Smooth L1-Loss β (see Definition D.1)	{1/32, 1/64, 1/128}	
	Clip Grad Norm	[1e-6, 1]	Yes
NEW NOMU Loss	Number of Artificial Input Data Points: $\lvert D^{\text{ART}}\rvert$ (see Section 3.1)	{64,80,96,...,512}	
	π_{SQR} (see Equation (9))	1	
	π_{EXP} (see Equation (9))	[1e-6, 5e-1]	Yes
	$\underline{\pi}$ (see Equation (9))	[64, 256]	Yes
	$\overline{\pi}$ (see Equation (9))	0.25	
	c_{EXP} (see Equation (9))	[64, 256]	Yes
MVNN Initialization	Random Initialized ts	Unif([0,1])	
	Trainable ts	{True, False}	
	Initial Expectation: $\mathbb{E}^{\text{INIT}}$ (see Appendix E.2)	[1, 2]	
	Initial Variance: $\mathbb{V}^{\text{INIT}}$ (see Appendix E.2)	[0.02, 0.16]	Yes
	Initial Bias: b^{INIT} (see Appendix E.2)	0.05	
	Initial "B" Constant: B^{INIT} (see Appendix E.2)	0.05	
	Initial "Little" Constant: ϵ (see Appendix E.2)	0.1	

Table 2: Hyperparameter ranges used in our HPO for random search. If not explicitly stated otherwise, the ranges apply to all considered SATS domains.

- CLIP GRAD NORM: Parameter for gradient clipping in PYTORCH via *torch.nn.utils.clip_grad_norm_()*.
- RANDOM INITIALIZED TS: Uniform distribution, which is used to initialize the bReLU cutoffs $t^{i,k}$ i.i.d. uniformly at random, i.e. $t^{i,k} \sim \text{Unif}(A, B)$ (setting $A = B$ makes those cutoffs deterministic).
- TRAINABLE TS: If set to TRUE, the cutoffs of the bReLU activation function $\{t^{i,k}\}_{k=1}^{K_i-1}$ are learned (i.e., trained) during the training procedure of the corresponding MVNN.

Evaluation Metric HPO We motivate our choice of the two terms in our evaluation metric Equation (18) in the following way:

1. The first term
$$|D^{\text{test}}|^{-1}\sum_{(x,y)\in D^{\text{test}}}\max\{(y-\mathcal{M}_i^{\text{uUB}}(x))q, (\mathcal{M}_i^{\text{uUB}}(x)-y)(1-q)\}$$
of Equation (18) is the standard quantile-loss applied on the test data set. Achieving a low value in this evaluation metric is intuitively desirable since for values $q > 0.5$ we penalize predictions that are too low more severely than predictions that are too high. It is also theoretically well motivated, since the true[28] posterior q-credible bound $\text{uUB}_\alpha(x) := \inf\{y \in \mathbb{R} : \mathbb{P}\left[\hat{v}_i(x) \leq y | D^{\text{train}}\right] \geq \alpha\}\ \forall x \in X$ would minimize this evaluation metric in expectation. As we average this term over 100 different value functions and as we use a large test data set for each of them, this is a good approximation for the expectation.
2. The second term $\text{MAE}(D^{\text{train}})$ of Equation (18) might appear to be counter-intuitive, because we are using the training data set. However, in BO it is particularly important to fit well through the noiseless training data points. First, the training data points in BO have already been chosen to lie in a region of potential maximizers. Second, in BO, relative uncertainty (Heiss et al. 2022, Appendix A.2.1.) and particularly *Desiderata D2* from (Heiss et al. 2022) are more important than calibration as discussed in (Heiss et al. 2022, Appendix D.2.3). Adding a constant value to $\mathcal{M}_i^{\text{uUB}}$ would calibrate them but would not change the argmax Equation (2) (i.e., would not change the selected queries). However, the 1st term of our evaluation metric Equation (18) alone would assign quite low values to $\mathcal{M}_i^{\text{uUB}}$ of the form $\mathcal{M}_i^{\text{mean}} + c$. Fortunately, the second term $\text{MAE}(D^{\text{train}})$ prevents Equation (18) from assigning low values to $\mathcal{M}_i^{\text{uUB}}$ of the form $\mathcal{M}_i^{\text{mean}} + c$.

G.3 Details MVNN Training

Both for the HPO as well as when running our efficiency experiments, we use the following two techniques to achieve numerically more robust results.

1. At the end of the training procedure we use the best weights from all epochs, and not the ones from the last epoch.
2. If at the end of the training procedure the R^2 (coefficient of determination) was below 0.9 on the training set, we retrain once and finally select the model with the best performance across these two attempts.

G.4 Details MILP Parameters

In our experiments we use for all MILPs CPLEX version 20.1.0. Furthermore, we set the following MILP parameters: `time_limit` $= 600$ sec, `relative_gap` $= 0.005$, `integrality_tol` $= 1e-06$, `feasibility_tol` $= 1e-09$. Thus, the MILP solver stops as soon as it has found a feasible integer solution proved to be within 0.5% of optimality and otherwise stops after 600 seconds. CPLEX is automatically proving upper bounds for the relative gap (using duality) while approximating the solution for the MILP. Thus, when CPLEX finds a solution with a relative gap of at most 0.5%, we have a guarantee that the solution found by CPLEX is *at most* 0.5% worse than the unknown true solution. However, we have no a priori theoretical guarantee that CPLEX is always able to find such a solution and such a bound within 600 sec. Note that we set the time limit to 600 sec to allow researchers with a limited budget to reproduce our results hundreds of times with different seeds, while achieving already SOTA. For an actual spectrum auction the additional costs for increasing the computational budget by a factor of 10 would be negligible compared to the statistical significant increase of 200 million USD of revenue gained from using BOCA instead of MVNN in the case of MRVM for example (see Appendix G.6). In our experiments, the median relative gaps[29] across all runs per domain in Table 1 were: LSVM=0.004987, SRVM=0.005000, MRVM=0.004991, indicating that in most of the runs the MILP solver found within the time limit an optimal solution within tolerance.

G.5 Details BOCA Results

In this section, we provide detailed efficiency loss results of BOCA. Specifically, we present in Table 3 efficiency loss results for all four HPO winner configurations (with respect to the evaluation metric based on the four different quantile parameters $q \in \{0.6, 0.75, 0.9, 0.95\}$). Furthermore, we present the *relative revenue* $\sum_{i\in N} p(R)_i / V(a^*)$ of an allocation $a_R^* \in \mathcal{F}$ and payments $p(R) \in \mathbb{R}_+^n$ determined by BOCA when eliciting reports R as well as the average runtime in hours for a single instance (i.e, how long it would take to run a single auction). Note that since we stop BOCA when we have already found an allocation with 0% efficiency loss, the relative revenue numbers (in LSVM and SRVM) are pessimistic and typically increase if we let BOCA run until $Q^{\max}$ is reached. For the runtime results the opposite holds.

G.6 Revenue

Comparing the revenue of BOCA (see Table 3) to the revenue of MVNN-MLCA and NN-MLCA in (Weissteiner et al. 2022a, Table 6), we see that overall the mean relative

[28] By "true posterior" we denote the posterior coming from the "true prior" that we sample our value functions $\hat{v}_i$ from.

[29] In CPLEX, the "relative gap" always refers to the proven upper bound of the relative gap.

DOMAIN	QUANTILE PARAMETER Q	Q^{INIT}	Q^{ROUND}	Q^{MAX}	EFFICIENCY LOSS IN % ↓	REVENUE IN % ↑	RUNTIME IN HOURS
LSVM	0.60	40	4	100	0.69±0.41	74.73±3.68	4.90
	0.75	40	4	100	0.69±0.44	75.07±3.71	5.53
	0.90	40	4	100	0.39±0.30	73.53±3.72	15.64
	0.95	40	4	100	0.40±0.35	73.88±3.93	15.58
SRVM	0.60	40	4	100	0.16±0.04	54.34±1.48	24.67
	0.75	40	4	100	0.06±0.02	54.22±1.46	18.80
	0.90	40	4	100	0.54±0.08	53.89±1.44	33.90
	0.95	40	4	100	0.62±0.11	54.25±1.54	33.26
MRVM	0.60	40	4	100	7.88±0.43	41.81±1.06	61.48
	0.75	40	4	100	8.44±0.43	41.89±0.93	34.91
	0.90	40	4	100	7.77±0.34	42.04±0.89	28.15
	0.95	40	4	100	7.98±0.34	42.28±1.00	27.92

Table 3: Detailed BOCA results. We present efficiency loss, relative revenue and runtime with our MVNN-based uUB $\mathcal{M}_i^{\text{uUB}}$ as $\mathcal{A}_i$. Shown are averages including a 95%-normal-CI on a test set of 50 instances in all three considered SATS domains. The best MVNN-based uUBs per domain (w.r.t. the quantile parameter q based on the lowest efficiency loss are marked in grey.

revenue is as good or better than SOTA. For LSVM (and SRVM) this comparison could be flawed because we stop computing further queries when reaching 0% efficiency loss as described above.

However, for MRVM, we always compute all 100 queries as can be seen in (Weissteiner et al. 2022a, Table 6). Thus MRVM allows for a fair comparison of the relative revenue. In MRVM, BOCA's average relative revenue is 7.58% points higher than the one of MVNN-MLCA , which corresponds to $\sim$400 million USD in this domain.[30] A pairwise t-test with null hypothesis of equal means resulted in $p_{\text{VAL}} = 5\text{e}{-10}$. Moreover, with high statistical significance ($p_{\text{VAL}} = 2\text{e}{-4}$), BOCA achieves on average more than 200 million USD *more* than MVNN-MLCA. The comparison of BOCA to NN-MLCA is also in favour of BOCA, but much closer (and BOCA outperforms NN-MLCA in terms of social welfare with a $p_{\text{VAL}} = 2\text{e}{-5}$, see Table 1).

The strength of BOCA with respect to revenue is quite intuitive since each query that explores regions of high uncertainty in the bundle space is beneficial for all economies, while "exploiting" an allocation which leads to high efficiency in one certain economy is mainly beneficial for this certain economy (see Appendix A for the definition of main and marginal economies). As discussed in (Weissteiner et al. 2022a, Appendix E.3), MLCA queries the main economy more often than any other economy and revenue is high if the social welfare in the marginal economies is high relative to the social welfare in the main economy. Thus, exploration favours high revenue in settings (such as ours) where the main economy is queried more often than the marginal economies.

[30]The revenue of the 2014 Canadian 4G auction was 5.27 billion USD (Ausubel and Baranov 2017). If one accumulates the revenue of all spectrum auctions, one would obtain significantly larger values.

G.7 Understanding BOCA's Performance Increase

In this section, we present further efficiency loss results when using only the mean MVNN $\mathcal{M}_i^{\text{mean}}$ of $\mathcal{M}_i^{\text{NOMU}}$ in MLCA as $\mathcal{A}_i$. We call this method OUR-MVNN-MLCA. Note that OUR-MVNN-MLCA does not integrate any explicit notion of uncertainty and is thus the same method as MVNN-MLCA from Weissteiner et al. (2022a), but now with our new proposed parameter initialization method (see Section 3.2) and optimized with our HPO. Thus, this experiment investigates how much of the performance gain is attributed to the integration of our notion of uncertainty compared to the other changes we made (i.e., our parameter initialization method and our HPO).

In Figure 8, we present the efficiency loss path plots for OUR-MVNN-MLCA (yellow) compared to the results presented in Figure 4 in the main paper. For each domain we use for OUR-MVNN-MLCA (yellow) the best mean models $\mathcal{M}_i^{\text{mean}}$ from $\mathcal{M}_i^{\text{NOMU}}$ based on the smallest efficiency loss (i.e., the grey-marked winners from Table 4).

As expected, we see that BOCA, i.e., integrating a notion of uncertainty, is as good or better than OUR-MVNN-MLCA, i.e., only using the mean model. This effect is statistically significant in SRVM, while in LSVM and MRVM both lead to results that are statistically on par. The degree to which exploration via a notion of uncertainty is beneficial depends on intrinsic characteristics of the domain (e.g., the dimensionality or multi-modality of the objective function). Specifically, in MRVM where the query budget of $Q^{\max} = 100$ is extremely small compared to the dimensionality of the domain (i.e., MRVM has $m = 98$ items and $n = 10$ bidders thus the dimensionality is $980 = 98 \cdot 10$), it appears that exploitation might be beneficial compared to adding exploration (De Ath et al. 2021) and the power of adding exploration may reveal itself only when increasing the query budget. However, in LSVM and SRVM ($m = 18$ and $m = 29$), we see that adding exploration with an uUB $\mathcal{M}_i^{\text{uUB}}$ decreases the efficiency loss. Finally, these results

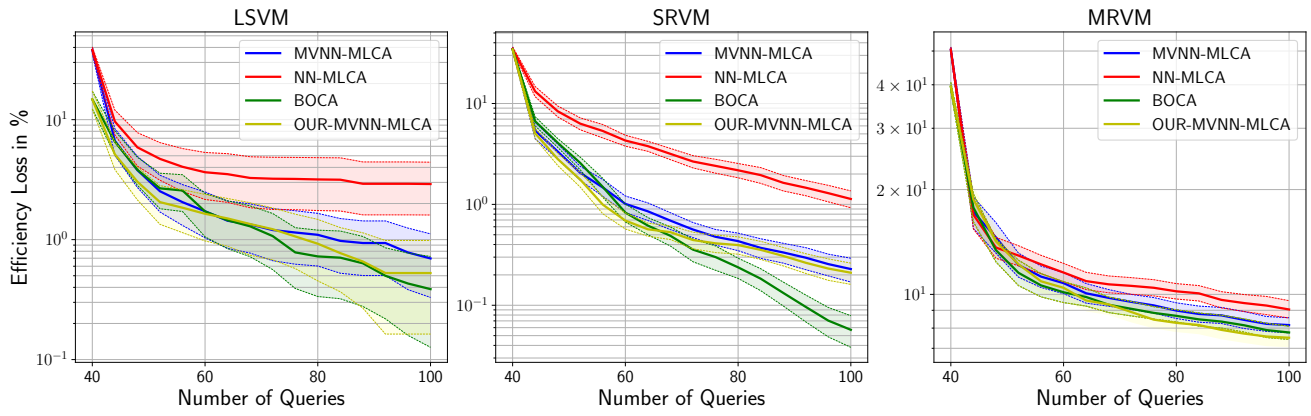


Figure 8: Efficiency loss paths (i.e., regret plots) for $Q^{\text{init}} = 40$ of OUR-MVNN-MLCA winners (yellow) from Table 4 compared to BOCA winners (green) from Table 3 and to the results presented in (Weissteiner et al. 2022a) of MVNNs (blue) and plain NNs (red). Shown are averages with 95% CIs over 50 instances.

also suggest that our proposed new parameter initialization method for MVNNs discussed in Section 3.2 tends to be better than a simple generic one, i.e, OUR-MVNN-MLCA is in every domain on average better than MVNN-MLCA.

Finally, in Table 4 we present the complete results of this experiment for all quantile parameters q per domain.

We refer to Appendix G.8 for an additional comparison of BOCA and OUR-MVNN-MLCA for a reduced number of initial queries.

G.8 Reduced Number of Initial Queries

In this section, we present results of BOCA and OUR-MVNN-MLCA for a reduced number of initial random queries. Specifically, we chose $Q^{\text{init}} = 20$ initial random queries (instead of $Q^{\text{init}} = 40$ that were selected in the main set of experiments following prior work; see Table 3 and Table 4). All other parameters are left untouched.

Note that the previous literature used only $Q^{\text{init}} = 40$, since batching reduces the cost per queries in practice, i.e., not only the queries are costly but also the rounds are costly. Reducing Q^{init} from 40 to 20 increases the number of rounds from 15 to 20 (given that $Q^{\text{round}} = 4$ queries, i.e., 3 marginal economy queries and 1 main economy query, are asked per round). Furthermore, the experiments get computationally more costly as we reduce Q^{init}, because we need to perform more (MV)NN trainings and solve more MILPs.

Because of this, we only compare BOCA vs. OUR-MVNN-MLCA in this section (as in Appendix G.7). Thus, in this section, we do not study the benefits of our new initialization method (Section 3.2), and only focus on studying the benefits of incorporating our proposed uncertainty model (Section 3.1).

In Table 5, we present the BOCA results for $Q^{\text{init}} = 20$. To isolate the effect of the integrated uncertainty in BOCA, we present the corresponding results for OUR-MVNN-MLCA for $Q^{\text{init}} = 20$ in Table 6, (see Appendix G.7 for a description of this experiment setting).

Effect of Q^{init} Parameter Comparing Table 5 to Table 3, we see that by reducing the randomly sampled initial queries from $Q^{\text{init}} = 40$ to $Q^{\text{init}} = 20$, BOCA' efficiency loss tends to be on average smaller for LSVM and MRVM and, perhaps surprisingly, larger for SRVM. However, by comparing the grey-marked winner models in each domain, we see that it does not make a statistically significant difference whether Q^{init} is chosen to be 20 or 40.

BOCA vs. OUR-MVNN-MLCA for $Q^{\text{init}} = 20$ Also when comparing BOCA to OUR-MVNN-MLCA for a reduced number of $Q^{\text{init}} = 20$ initial queries, i.e., comparing Table 5 to Table 6, we find that in each domain the BOCA winner model and the OUR-MVNN-MLCA winner model perform statistically on par (even though the average efficiency loss of the BOCA winner model is always better than that of the OUR-MVNN-MLCA winner model). This can also be seen in the efficiency loss paths (i.e., regret plots) shown in Figure 9. Furthermore, Figure 9 suggests that for a small query budget of $Q^{\max} < 60$, exploitation might be more important, while for query budgets larger than 80, i.e., $Q^{\max} > 80$, exploration might pay off more.

DOMAIN	QUANTILE PARAMETER Q	Q^{INIT}	Q^{ROUND}	Q^{MAX}	EFFICIENCY LOSS IN % ↓	REVENUE IN % ↑	RUNTIME IN HOURS
LSVM	0.60	40	4	100	0.66±0.44	74.93±3.48	4.62
	0.75	40	4	100	0.61±0.42	75.14±3.53	4.16
	0.90	40	4	100	0.53±0.41	73.80±3.88	10.44
	0.95	40	4	100	0.56±0.40	74.83±3.67	10.31
SRVM	0.60	40	4	100	0.22±0.04	54.24±1.51	21.18
	0.75	40	4	100	0.21±0.05	54.47±1.51	17.81
	0.90	40	4	100	0.36±0.06	54.51±1.49	20.03
	0.95	40	4	100	0.35±0.05	54.51±1.48	19.14
MRVM	0.60	40	4	100	7.86±0.42	41.91±0.91	41.51
	0.75	40	4	100	8.41±0.40	40.28±0.92	6.89
	0.90	40	4	100	7.51±0.47	40.22±0.96	4.91
	0.95	40	4	100	8.01±0.48	40.62±1.08	9.03

Table 4: Detailed OUR-MVNN-MLCA results. We present efficiency loss, relative revenue and runtime of OUR-MVNN-MLCA, i.e., MLCA with our MVNN-based mean $\mathcal{M}_i^{\text{mean}}$ as $\mathcal{A}_i$. Shown are averages including a 95%-normal-CI on a test set of 50 instances in all three considered SATS domains. The best $\mathcal{M}_i^{\text{mean}}$ per domain (w.r.t. the quantile parameter q) based on the lowest efficiency loss is marked in grey.

DOMAIN	QUANTILE PARAMETER Q	Q^{INIT}	Q^{ROUND}	Q^{MAX}	EFFICIENCY LOSS IN % ↓	REVENUE IN % ↑	RUNTIME IN HOURS
LSVM	0.60	20	4	100	0.79±0.47	73.74±3.65	5.93
	0.75	20	4	100	0.61±0.42	73.99±3.56	7.31
	0.90	20	4	100	0.37±0.24	73.18±3.60	23.99
	0.95	20	4	100	0.16±0.20	72.84±3.45	22.21
SRVM	0.60	20	4	100	0.14±0.04	53.86±1.44	32.07
	0.75	20	4	100	0.12±0.09	53.87±1.55	25.89
	0.90	20	4	100	0.72±0.11	53.94±1.59	45.20
	0.95	20	4	100	0.75±0.11	54.09±1.55	45.09
MRVM	0.60	20	4	100	7.94±0.36	42.14±0.98	68.84
	0.75	20	4	100	8.31±0.31	40.92±0.73	27.09
	0.90	20	4	100	7.92±0.33	42.61±0.89	21.70
	0.95	20	4	100	7.45±0.37	41.19±0.86	21.51

Table 5: Detailed BOCA results for a reduced number of $Q^{\text{init}} = 20$ initial random queries. We present efficiency loss, relative revenue and runtime of MLCA with our MVNN-based uUB $\mathcal{M}_i^{\text{uUB}}$ as $\mathcal{A}_i$. Shown are averages including a 95%-normal-CI on a test set of 50 instances in all three considered SATS domains. The best MVNN-based uUBs per domain (w.r.t. the quantile parameter q) based on the lowest efficiency loss are marked in grey.

DOMAIN	QUANTILE PARAMETER Q	Q^{INIT}	Q^{ROUND}	Q^{MAX}	EFFICIENCY LOSS IN % ↓	REVENUE IN % ↑	RUNTIME IN HOURS
LSVM	0.60	20	4	100	0.71±0.43	73.73±3.57	4.87
	0.75	20	4	100	0.71±0.45	74.71±3.32	5.11
	0.90	20	4	100	0.58±0.38	74.81±3.51	12.76
	0.95	20	4	100	0.59±0.34	73.88±3.64	14.11
SRVM	0.60	20	4	100	0.18±0.04	54.13±1.45	30.47
	0.75	20	4	100	0.16±0.04	54.32±1.49	28.62
	0.90	20	4	100	0.29±0.06	54.55±1.51	31.74
	0.95	20	4	100	0.30±0.05	54.35±1.58	30.39
MRVM	0.60	20	4	100	7.73±0.43	42.51±0.75	25.39
	0.75	20	4	100	8.14±0.40	41.46±0.85	7.70
	0.90	20	4	100	7.73±0.41	41.42±0.85	10.06
	0.95	20	4	100	7.52±0.36	41.33±1.11	9.44

Table 6: Detailed OUR-MVNN-MLCA results for a reduced number of $Q^{\text{init}} = 20$ initial random queries. We present efficiency loss, relative revenue and runtime of OUR-MVNN-MLCA, i.e., MLCA with our MVNN-based mean $\mathcal{M}_i^{\text{mean}}$ as $\mathcal{A}_i$. Shown are averages including a 95%-normal-CI on a test set of 50 instances in all three considered SATS domains.

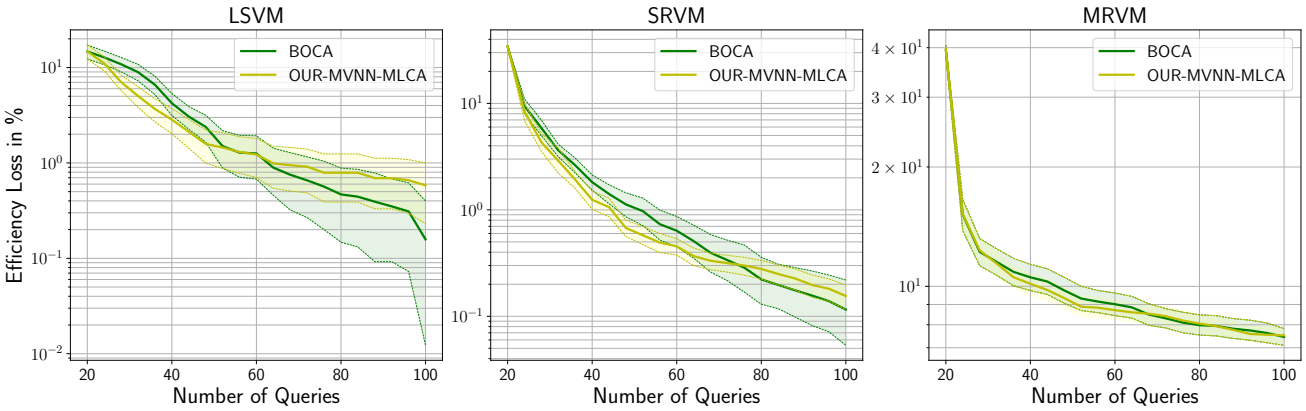


Figure 9: Efficiency loss paths (i.e., regret plots) for a reduced number of initial queries $Q^{\text{init}} = 20$ of BOCA winners (green) from Table 5 compared to OUR-MVNN-MLCA winners (yellow) from Table 6. Shown are averages with 95% CIs over 50 instances.

7 Machine Learning-powered Course Allocation

The content of this chapter has previously appeared in

Machine Learning-powered Course Allocation.
Ermis Soumalias*, Behnoosh Zamanlooy*, Jakob Weissteiner and Sven Seuken.
Working paper, March 2023.
URL: arxiv.org/pdf/2210.00954.pdf

*These authors contributed equally.

Machine Learning-powered Course Allocation

ERMIS SOUMALIAS*, University of Zurich & ETH AI Center, ermis@ifi.uzh.ch
BEHNOOSH ZAMANLOOY*, McMaster University, zamanlob@mcmaster.ca
JAKOB WEISSTEINER, University of Zurich & ETH AI Center, weissteiner@ifi.uzh.ch
SVEN SEUKEN, University of Zurich & ETH AI Center, seuken@ifi.uzh.ch

We introduce a machine learning-powered course allocation mechanism. Concretely, we extend the state-of-the-art *Course Match* mechanism with a machine learning-based preference elicitation module. In an iterative, asynchronous manner, this module generates pairwise comparison queries that are tailored to each individual student. Regarding incentives, our *machine learning-powered course match (MLCM)* mechanism retains the attractive *strategyproofness in the large* property of Course Match. Regarding welfare, we perform computational experiments using a simulator that was fitted to real-world data. Our results show that, compared to Course Match, MLCM increases average student utility by 4%-9% and minimum student utility by 10%-21%, even with only ten comparison queries. Finally, we highlight the practicability of MLCM and the ease of piloting it for universities currently using Course Match.

*These authors contributed equally.

1 INTRODUCTION

The *course allocation problem* arises when educational institutions assign bundles of courses to students [Budish and Cantillon, 2012]. Each course has a limited number of indivisible seats and monetary transfers are prohibited for fairness reasons. What makes this problem particularly challenging is the combinatorial structure of the students' preferences, as students may view certain courses as complements or substitutes [Budish and Kessler, 2022].

1.1 Course Match

The state-of-the-art practical solution to the course allocation problem is the *Course Match (CM)* mechanism by Budish et al. [2017], which provides a good trade-off between efficiency, fairness, and incentives. CM has now been adopted in many universities such as the Wharton School at the University of Pennsylvania and Columbia Business School.

CM uses a simple reporting language to elicit students' preferences over *schedules* (i.e., course bundles). Concretely, CM offers students a graphical user interface (GUI) to enter a *base value* between 0 and 100 for each course, and an *adjustment value* between −200 and 200 for each *pair* of courses. Adjustments allow students to report complementarities and substitutabilities between courses, up to *pairwise* interactions. The total value of a schedule is then the sum of the base values reported for each course in that schedule plus any adjustments (if both courses are in the schedule).

Prior to the adoption of CM in practice, Budish and Kessler [2022] performed a lab experiment to evaluate CM. Regarding efficiency, they found that, on average, students were happier with CM compared to the *Bidding Points Auction* [Sönmez and Ünver, 2010], the previously used mechanism. Regarding fairness, students also found CM fairer. Regarding the reporting language, they found that students were able to report their preferences "accurately enough to realize CM's theoretical benefits." Given these positive findings, Wharton was then the first school to switch to CM.

1.2 Preference Elicitation Shortcomings of Course Match

However, Budish et al. [2017] were already concerned that the CM language may not be able to fully capture all students' preferences. Furthermore, they mentioned that some students might find it non-trivial to use the CM language and might therefore make *mistakes* when reporting their preferences. Indeed, the lab experiment by Budish and Kessler [2022] revealed several shortcomings of CM in this regard.

First, students made very limited use of the CM language: on average, students only reported a base value for half of the 25 courses in the experiment. Furthermore, the average number of pairwise adjustments was only 1.08 (out of 300), and the median was 0. This suggests that cognitive limitations negatively affect how well students can report their preferences using the CM language. Second, in addition to not reporting part of their preferences, Budish and Kessler [2022] provided evidence that students are also *inaccurate* when they do report their preferences.

Budish and Kessler [2022] found that both of these reporting mistakes negatively affected the welfare of CM. In their experiment, about 16% of students would have preferred another schedule of courses, with a median utility difference for these schedules of 13%. Thus, preference elicitation in course allocation still remains an important challenge.

1.3 Machine Learning-powered Preference Elicitation

To address this challenge, we turn to *machine learning (ML)*. The high-level idea is to train a separate ML model for each student based on that student's reports after using the CM language (i.e., the GUI). These ML models can then be used in an *ML-powered preference elicitation algorithm* that asks each student a sequence of carefully selected queries (thus, enabling them to correct the reporting

mistakes they made in the GUI). Based on those queries, the student's ML model is updated in real-time and the next query is generated. At the end, the mechanism allocates schedules to students based on their trained ML models.

With our approach, we build on the ideas developed in a recent stream of papers on ML-powered preference elicitation. Lahaie and Parkes [2004] and Blum et al. [2004] were the first to combine ML and mechanism design, studying the relation between learning theory and preference elicitation. Brero et al. [2017, 2018] were the first to integrate an ML-powered preference elicitation component into a practical combinatorial auction mechanism. They used support vector regression (SVR) to learn bidders' value functions and to iteratively generate new queries in each auction round. In [Brero et al., 2021], the authors proposed the MLCA mechanism and showed that it achieves higher allocative efficiency than the widely-used combinatorial clock auction [Ausubel et al., 2006]. In recent years, there has been a stream of papers further improving the ML capability of the MLCA mechanism, which we discuss in Section 2.

While these works are important pre-cursors to the present paper, there are several noteworthy differences. First, these papers used *value queries* as the interaction paradigm (i.e., asking agents a query of the form "What is your value for bundle {XYZ}"), which would be unnatural in course allocation. Instead, we use *pairwise comparison queries* (i.e., asking students "Do you prefer course schedule A or B?"). Importantly, a pairwise comparison query is a simpler type of query, known to have low cognitive load [Chajewska et al., 2000, Conitzer, 2009]. Second, our goal is to build on top of the CM language; thus, we must be able to handle the *cardinal* input that students provide via the CM reporting language as well as the *ordinal* feedback from answering comparison queries. Third, while an auctioneer can require bidders in an auction to participate in a synchronous way (i.e., submitting a bid in every round), we must allow students to interact with the mechanism in an asynchronous manner (i.e., allowing students to answer a sequence of comparison queries without having to wait on other students). Finally, MLCA could only use the ML models to elicit information, but it could *not* use them to determine the final outcome, as that would often lead to an individual rationality violation. In contrast, in our setting (without monetary transfers), we can *also* use the ML models to determine the final allocation, which leads to additional efficiency gains.[1]

1.4 Overview of Contributions

In this paper, we introduce the *machine-learning based course match (MLCM)* mechanism (Section 4). MLCM builds on top of CM by improving its preference elicitation component. Importantly, our design makes the process of upgrading from CM to MLCM particularly simple for the universities and seamless for their students.

First, students use the CM reporting language (i.e., via the same GUI). As in CM, this input is required from all students. Second, MLCM uses these initial reports to train a separate ML model for each student so that it can predict each student's utility for any possible course schedule. Third, MLCM uses an ML-powered preference elicitation algorithm to generate *pairwise comparison queries* that are tailored to each student, and students simply answer which schedule they prefer. Based on this feedback, the ML model is retrained and the next query is generated. Importantly, this phase is *optional* – each student can answer as many such queries as she wants (including none). However, the more queries she answers, the better the ML model will typically approximate her true preferences, which will benefit her in the last phase, where MLCM computes the final allocation based on all ML models (in case a student has answered no queries, then only her GUI reports will be used for the final allocation calculation).

[1] An alternative approach would be to only use the trained ML models to update the student's GUI reports, and to then run the original CM mechanism on the updated GUI reports. See Section 8 for a discussion on this approach.

To understand the theoretical properties of MLCM, we extend the theoretical guarantees of CM to MLCM (Section 4.3). Importantly, we explain why MLCM is also *strategyproof in the large*.[2]

To evaluate MLCM, we introduce a new course allocation simulation framework (Section 5). The first component is a realistic student preference generator, which is designed such that each student's complete preferences can be encoded in a succinct *Mixed Integer Program* (MIP). This allows computing a benchmark allocation given the students' true preferences. The second component models the students' *reporting mistakes* when interacting with the CM language. We calibrate the framework's parameters based on real-world data from Budish and Kessler [2022].

In Section 6, we instantiate the ML model used by MLCM. We show experimentally that the recently introduced *monotone-value neural networks (MVNNs)* [Weissteiner et al., 2022a] exhibit the best generalization performance in our domain, while also being MIP-formalizable, such that the corresponding utility maximization problem can be solved fast enough in practice. Furthermore, we show how the cardinal input from the CM language and the ordinal feedback from the comparison queries can be combined when training neural networks.

In Section 7, we empirically evaluate the performance of MLCM. We find that MLCM significantly outperforms CM in terms of *average student utility* as well as *minimum student utility*, even with only five comparison queries (Section 7.2). Furthermore, we show that these results are robust to changes in students' reporting mistakes (Section 7.3), we show the expected benefit of an individual student unilaterally opting into MLCM (Section 7.4), we show that MLCM also outperforms CM when students have simple additive preferences (Section 7.5), and we show that the runtime of our mechanism scales gracefully in the number of courses (Section 7.6).

In Section 8, we put our results into perspective and discuss alternative approaches one could have taken. Finally, we conclude in Section 9 and discuss interesting avenues for future work.

2 RELATED WORK

Our work is related to the research on course allocation and ML-based preference elicitation.

2.1 Course Allocation

The course allocation problem is an instance of the *combinatorial assignment problem*, for which several impossibility results establish a tension between welfare, incentives, and fairness. For example, it is known that the only mechanisms for this problem that are ex-post Pareto efficient and strategyproof are dictatorships [Hatfield, 2009, Pápai, 2001].

Multiple empirical studies have pointed out design flaws of course allocation mechanisms used in practice. Budish and Cantillon [2012] showed that the *Harvard Business School (HBS) draft* mechanism [Brams and Straffin Jr, 1979] creates significant incentives for students to misreport, leading to large welfare losses. Similarly, the commonly used *Bidding Points Auction* [Sönmez and Ünver, 2010] implicitly assumes that students have positive value for left-over virtual currency, which harms incentives and ultimately leads to allocations that are neither efficient nor fair.

Motivated by these design flaws, Budish [2011] proposed a new mechanism for the combinatorial assignment problem called *approximate competitive equilibrium from equal incomes (A-CEEI)*. A-CEEI circumvents the impossibility results previously mentioned by making slight compromises in all three of those dimensions of interest. Specifically, A-CEEI is approximately efficient, satisfies desirable fairness criteria (envy bounded by a single good and $(n+1)$-maximin share guarantee), and is strategyproof in the large [Azevedo and Budish, 2019]. Later, Budish et al. [2017] introduced CM as the practical implementation of A-CEEI. We present A-CEEI and CM in Section 3.

[2] A mechanism is *strategyproof in the large* if, for a large enough number of students and any full-support i.i.d. distribution of opponent reports, reporting truthfully is approximately interim optimal [Azevedo and Budish, 2019].

Diebold et al. [2014] modeled course allocation as a two-sided matching problem where instructors also have preferences over students, which makes the problem quite different from combinatorial assignment. Bichler and Merting [2021] studied the assignment of tutorials for mandatory courses, which is more similar to a scheduling problem.

2.2 Machine Learning-based Preference Elicitation

Preference elicitation (PE) using comparison queries has received a lot of attention in the ML community. Bayesian approaches are a natural candidate for PE due to their ability to explicitly model uncertainty of users' utility functions. Chu and Ghahramani [2005] and Bonilla et al. [2010] used Gaussian processes (GPs) to learn and elicit preferences. In particular, Chu and Ghahramani [2005] used GPs for the problem of learning preferences given a (fixed) set of comparison queries. However, they did not answer the question of *which* comparison query to ask. Bonilla et al. [2010] addressed this question by iteratively selecting a pairwise comparison query that maximizes the *expected value of information* (EVOI). However, their approach is impractical in our setting, as the EVOI needs to be analytically calculated for each possible comparison query, thus scaling quadratically in the number of alternatives (course schedules in our case), which is already polynomial in the number of courses. Guo and Sanner [2010] introduced an approximate PE framework for performing efficient closed-form Bayesian belief updates and query selection for a multi-attribute belief state, speeding up the evaluation of EVOI heuristics. However, their approach is inherently limited as it can only model additive utility functions, and is thus not well-suited for course allocation, where students have more complex, non-additive preferences. In general, GPs are also not well suited to our setting due to the high dimensionality of the input space and the integrality constraints that make GP optimization intrinsically difficult.

Ailon [2012] took a different approach, proposing an active learning algorithm that, using binary comparison queries which may be non-transitive, can learn an almost optimal linear ordering of a set of alternatives with an almost optimal query complexity, which is further improved by Ailon et al. [2011]. However, these approaches are impractical for course allocation because they would require more than one hundred thousand queries per student. The reason for this large number of queries is that they do not exploit any notion of similarity between schedules.

Most related to our work is the research on ML-powered combinatorial auctions using SVRs mentioned above (see Section 1.3). Weissteiner and Seuken [2020] extended this work further by using neural networks (NNs), which further increased the efficiency. Weissteiner et al. [2022b] introduced Fourier-sparse approximations for the problem of learning combinatorial preferences. Weissteiner et al. [2022a] introduced *MVNNs*, which are specifically designed to learn *monotone* combinatorial preferences, resulting in a further efficiency increase. Finally, Weissteiner et al. [2023] proposed a *Bayesian optimization-based combinatorial assignment (BOCA)* mechanism which includes a notion of posterior *model uncertainty* [Heiss et al., 2022] to properly balance the *explore-exploit dilemma* during the preference elicitation phase in a principled manner.

3 PRELIMINARIES

In this section, we first present our formal model and then review A-CEEI and its practical implementation, CM.

3.1 Formal Model

Let $N = \{1, \dots, n\}$ denote the set of students indexed by i, and let $M = \{1, \dots, m\}$ denote the set of courses indexed by j. Each course j has a *capacity* $q_j \in \mathbb{N}_{>0}$. Each student i has a set $\Psi_i \subseteq 2^M$ of *permissible course schedules*. Ψ_i encapsulates both scheduling constraints as well as any student-specific constraints. An indicator vector $x \in \mathcal{X} = \{0, 1\}^m$ denotes a course schedule where $x_j = 1$

iff course $j \in M$ is part of schedule x. We let $a = (a_i)_{i \in N} \in \mathcal{X}^n$ denote an *allocation* of course schedules to students, where a_i is the course schedule of student i. A course j is *oversubscribed* in an allocation a iff $\sum_{i=1}^n a_{ij} > q_j$ and *undersubscribed* iff $\sum_{i=1}^n a_{ij} < q_j$. We denote the set of feasible allocations by $\mathcal{F} = \{a \in \mathcal{X}^n : \sum_{i=1}^n a_{ij} \leq q_j \ \forall j \in M, a_i \in \Psi_i \ \forall i \in N\}$. Students' preferences over course schedules are represented by their (private) *utility functions* $u_i : \mathcal{X} \to \mathbb{R}_+$, $i \in N$, i.e., $u_i(x)$ represents student i's utility for course schedule x.

3.2 Approximate Competitive Equilibrium from Equal Incomes (A-CEEI)

Budish [2011] proposed the A-CEEI mechanism as an approximation to the competitive equilibrium from equal incomes (CEEI). A-CEEI simulates a virtual economy where students are assigned budgets that are approximately (but not exactly) equal. To introduce A-CEEI formally, we represent each student i's complete ordinal preferences by $\preceq_i$. Each student i is allocated a budget $b_i \in [1, 1+\beta]$, $\beta > 0$. Next, *approximate market-clearing prices* $p^* \in \mathbb{R}^m_{\geq 0}$ are calculated (where p^*_j is the price for course j) such that, when each student i purchases her favorite permissible schedule a^*_i within her budget, the market approximately clears. Formally, given an allocation a^*, the *clearing error* z_j for course j and price p^*_j is

$$z_j := \begin{cases} \sum_i a^*_{ij} - q_j & p^*_j > 0, \\ \max\left\{\sum_i a^*_{ij} - q_j, 0\right\} & p^*_j = 0. \end{cases} \tag{1}$$

The *clearing error* of the allocation a^* is defined as $\alpha := \sqrt{\sum_j z_j^2}$. Following Budish [2011], we say that a price vector p^* *approximately clears* the market if $\alpha \leq \sqrt{\sigma m}/2$, with $\sigma = \min\{2k, m\}$, where k is the maximum number of courses in a permissible schedule.[3] Each student i is allocated her utility-maximizing schedule a^*_i that is permissible and within her budget. Formally,

$$a^*_i \in \arg\max_{\preceq_i} \left[a_i \in \Psi_i : \sum_j a_{ij} p^*_j \leq b_i \right]. \tag{2}$$

Then $[a^*, b, p^*]$ constitutes an (α, β)-A-CEEI.

3.3 Course Match (CM)

A-CEEI has many attractive properties (see Section 2), but it cannot be directly implemented in practice for multiple reasons. First, A-CEEI assumes access to the students' *full* ordinal preferences. Second, A-CEEI only *approximately* clears the market, which implies that some courses could be oversubscribed (which would violate a hard capacity constraint in many business schools, where seats cannot easily be added to a classroom). Third, the combinatorial allocation problem is PPAD-complete [Othman et al., 2016]; thus, we do not have a polynomial-time algorithm to solve it. To address these challenges, Budish et al. [2017] introduced *Course Match (CM)* as a practical implementation of A-CEEI. In CM, students first report their preferences using the GUI (see Section 1.1); the final allocation is then computed in three stages (see Figure 1).

Stage 1. In Stage 1, CM uses *tabu search* [Glover et al., 2018] to find a price vector that constitutes an A-CEEI. To do this, for every price vector examined, for every student, a MIP has to be solved to determine the student's utility-maximizing schedule within her budget.

Stage 2. In Stage 2, CM removes oversubscription by iteratively increasing the price of the most oversubscribed course until no oversubscribed courses are left.

[3] Budish [2011] proved that for any $\beta > 0$ such a price vector always exists.

Stage 3. In Stage 3, CM reduces undersubscription by first increasing all students' budgets by a fixed percentage and then allowing students, one after the other, to "purchase" courses that still have seats available.

4 MACHINE LEARNING-POWERED COURSE MATCH (MLCM)

In this section, we introduce our *ML-powered Course Match* (MLCM) mechanism and discuss its theoretical properties. Here, we describe MLCM for a generic ML model denoted $\mathcal{M}$. In Section 6, we instantiate the ML model using MVNNs.

4.1 Details of MLCM

MLCM proceeds in five phases (see Figure 1).

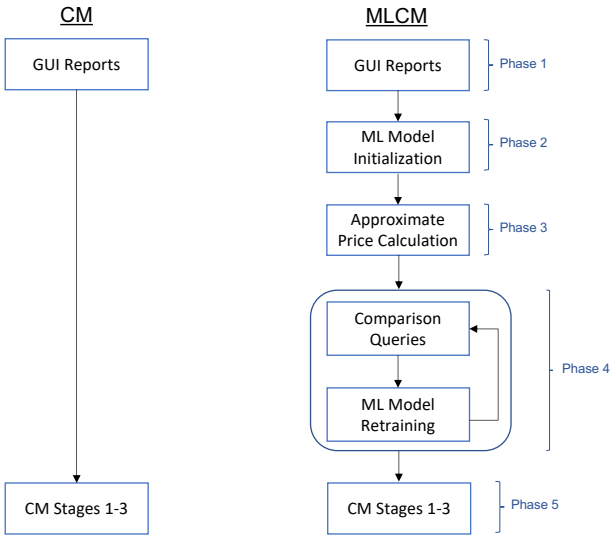


Fig. 1. Schematic overview of CM and MLCM

Phase 1: Preference Reporting via GUI. In Phase 1, students initially report their preferences using the same reporting language (and same GUI) as in CM (see Section 1.1). After this phase, each student can decide whether she also wants to use MLCM's ML-based preference elicitation feature (which we will simply call the "ML feature" going forward). If a student decides to "opt out", then MLCM treats that student's preference reports in the same way as CM would, without employing any ML.

Phase 2: ML Model Initialization. In Phase 2, for each student i that has not opted out of the ML feature, MLCM creates an initial ML model of her utility function based on her GUI reports from Phase 1. To do so, MLCM creates a *cardinal* training data set $D_{i,\text{card}}$ consisting of ℓ schedule-value pairs implied by student i's reports from Phase 1, i.e., $D_{i,\text{card}} = \{(x_{ik}, \hat{u}_i(x_{ik})\}_{k=1\ldots\ell}$ (see Appendix A for details) and then uses $D_{i,\text{card}}$ to train an initial ML model $\mathcal{M}_i^0 : \mathcal{X} \to \mathbb{R}_+$, where $\mathcal{M}_i^0(x)$ denotes the ML model's prediction of student i's utility for schedule x. See Section 6, for how to choose the ML model class.

Phase 3: Approximate Price Calculation. Next, MLCM runs Stage 1 of CM to calculate approximately market-clearing prices for all courses. As those prices are not final, but are only used to steer the preference elicitation in Phase 4, one can use less computation time for this step than in Phase 5, accepting a larger clearing error.[4] For those students who have opted out of the ML feature, MLCM uses the values implied by the CM GUI; for every other student i, it uses the values predicted by the ML model $\mathcal{M}_i^0$.

Phase 4: ML-based Preference Elicitation. In this phase, MLCM uses an ML-powered algorithm to generate a sequence of comparison queries (CQs) for every individual student. The algorithm has three main steps: (1) maintain an ordered list S_i of already queried schedules; (2) use the ML model to determine the next schedule x with the highest predicted value; (3) use "binary search" to generate a sequence of CQs, until the new schedule x can be sorted into S_i. Note that students can

[4]If it was desired that students could immediately start answering comparison queries after reporting their initial preferences via the GUI, one could also use last year's prices for Phase 4 (and setting the price of new courses to some average price). Of course, if students' preferences for courses have changed significantly, or if the new courses are particularly popular or unpopular, then using last year's prices would lower the effectiveness of the ML-powered elicitation phase.

ALGORITHM 1: ML-powered CQ Generation for Student i

Input: Initial ML model $\mathcal{M}_i^0$, cardinal data set $D_{i,\text{card}}$, price vector $\hat{p}$, permissible schedules Ψ_i, budget b_i
Output: Updated ML model $\mathcal{M}_i^t$

1: $S_i^0 = \{x \in \arg\max_{x' \in \Psi_i \cap D_{i,\text{card}}: x' \cdot \hat{p} \leq b_i} \mathcal{M}_i^0(x')\}$
2: **for** $t = 1$ to ∞ **do**
3: $\quad x \in \arg\max_{x' \in \Psi_i, x' \notin S_i^{t-1}: x' \cdot \hat{p} \leq b_i} \mathcal{M}_i^{t-1}(x')$
4: $\quad CQs^t$ = BinarySearchQueries(S_i^{t-1}, x)
5: $\quad$ **if** student i answered all CQs^t **then**
6: $\quad\quad S_i^t$ = Sort(S_i^{t-1}, x) based on answers to CQs^t
7: $\quad\quad D_{i,\text{ord}}^t$ = All pairwise orderings implied by S_i^t
8: $\quad\quad \mathcal{M}_i^t$ = Train($D_{i,\text{card}}, D_{i,\text{ord}}^t$)
9: $\quad$ **else**
10: $\quad\quad D_{i,\text{ord}}^t = D_{i,\text{ord}}^{t-1} \cup$ {answers to CQs^t}
11: $\quad\quad \mathcal{M}_i^t$ = Train($D_{i,\text{card}}, D_{i,\text{ord}}^t$)
12: $\quad\quad$ **break**
13: $\quad$ **end if**
14: **end for**
15: **return** $\mathcal{M}_i^t$

answer as many of these CQs as they like, and stop at any time. Algorithm 1 formally describes the process. In Line 1, we create the initial set of totally ordered schedules S_i^0, which only contains the one schedule x' from the cardinal training set with the highest predicted value $\mathcal{M}_i^0(x')$. Then we iteratively repeat the following procedure: In Line 3, we determine the highest-valued schedule x according to $\mathcal{M}_i^{t-1}$ that is not contained in S_i^{t-1}, subject to feasibility and budget constraints. In Line 4, we ask student i the CQs to determine x's position in the ordered set S_i^{t-1}. If i answered all CQs, then in Line 6, we completely order $S_i^t = S_i^{t-1} \cup \{x\}$ based on her answers, and then in Line 7, we create the ordinal training set $D_{i,\text{ord}}^t$ that contains all pairwise orderings implied by S_i^t. Finally, we train $\mathcal{M}_i^t$ on both $D_{i,\text{card}}$ and the ordinal set $D_{i,\text{ord}}^t$. If the student did not answer all CQs, then in Line 10, we append her (partial) answers to the previous ordinal data set $D_{i,\text{ord}}^t$, and then in Line 11, we train $\mathcal{M}_i^t$ using both data sets. This is the final ML model we return in Line 15.

Remark 4.1. *One could envision several alternative query heuristics. We have already conducted experiments using the following alternatives: (1) comparing the currently predicted utility-maximizing bundle only against the best one queried previously, and (2) comparing at each iteration the two schedules with the highest predicted utility. However, both approaches led to worse results compared to Algorithm 1. See Section 9 for possible future work on query generation procedures.*

Phase 5: Computing the Final Allocation. Finally, MLCM runs Stages 1–3 of CM to determine the final allocation. For those students who have opted out of the ML feature, MLCM uses the values implied by the CM GUI; for every other student i, it uses the utility function implied by $\mathcal{M}_i^t$.

Remark 4.2 (Runtime Considerations and Real-time Interaction). *In MLCM, Phases 3 (approximate price vector calculation) and 5 (final allocation calculation) are computationally expensive, because in those phases, MLCM computes an A-CEEI. However, this is not a concern in practice, because both of those phases do not happen in real-time (and furthermore, this computation is embarrassingly parallelizable). For a large school, one would use a compute cluster to calculate an A-CEEI for both of these phases. Furthermore, in work concurrent to the present paper, Budish et al. [2023] introduced a*

Preference Model	Courses 1	2	3	4	Utility Maximizing Schedule a_1^*	Utility $u_1(a_1^*)$	Answer to CQ
u_1	85	70	50	40	{1,3}	135	
u_1^{GUI}	75	77	42	45	{2,4}	110	
$\mathcal{M}_1^{t=0}$	75	77	42	45	{2,4}	110	$\{1,4\} \succ \{2,4\}$
$\mathcal{M}_1^{t=1}$	80	72	42	45	{1,4}	125	$\{1,3\} \succ \{1,4\}$
$\mathcal{M}_1^{t=2}$	80	72	47	40	{1,3}	135	$\{1,4\} \succ \{2,3\}$
$\mathcal{M}_1^{t=3}$	80	72	47	40	{1,3}	135	$\{2,3\} \succ \{2,4\}$
$\mathcal{M}_1^{t=4}$	80	72	47	40	{1,3}	135	

Table 1. Worked example illustrating the ML-based preference elicitation algorithm. Each row represents the linear coefficients (corresponding to the base values) that uniquely define the corresponding function, the current utility maximizing schedule a_1^*, its corresponding utility $u_1(a_1^*)$ and the answer to the CQ.

novel algorithm for computing an A-CEEI that is multiple orders of magnitudes faster than the tabu search currently in use, which will likely remove any runtime concerns in the near future.

The only phase with real-time interaction is Phase 4 (ML-based Preference Elicitation), which is computationally very cheap for our Algorithm 1. There are two distinct cases. If a student answered a CQ that did not complete an iteration of binary search (Line 4 in Algorithm 1), then generating the next query takes milliseconds. If a student answered a CQ that completed a round of binary search, then for the ML model class we selected, retraining the student's ML model (Line 8 in Algorithm 1) and solving the MIP to determine the bundle for the next iteration of binary search (Line 3 in Algorithm 1) takes less than two seconds, and can be further improved with the use of GPUs (please see Section 6 for details on the selected ML model class). Thus, in terms of real-time interaction, students do not perceive the computational cost, which highlights MLCM's practicability.

4.2 MLCM - A Worked Example

In this subsection, we present a worked example to illustrate MLCM's ML-based preference elicitation algorithm (i.e., *Phase 4*). Additionally, this example provides intuition for how the CQs help the ML model correct students' initial reporting mistakes. Note that we assume that students make no mistakes when answering CQs.

Example 4.3 (Correcting reporting mistakes). *We consider a setting with four courses $M := \{1,2,3,4\}$ with capacity 1 each. There is a single student $N := \{1\}$ who has a budget of $b_1 = 1$ and who wants a schedule consisting of at most two courses (and there are no other student-specific constraints). The prices of the courses are $\hat{p} := (0.6, 0.6, 0.3, 0.3)$, such that the student can afford all bundles of size two, except for $\{1,2\}$. For ease of exposition, we assume that the student has additive preferences and that the GUI only allows for additive preference reports. The whole example is presented in Table 1. The student's true utility function u_1 is presented in row 1. The student's GUI reports u_1^{GUI} (which include reporting mistakes) are shown in row 2. As the ML model $\mathcal{M}_1^t$ we use linear regression, such that each linear coefficient can be interpreted as an estimate of the student's base value for a course. Let $\mathcal{M}_1^t(x)$ denote the ML model trained on the cardinal data $D_{1,card}$ as well as $t \in \mathbb{N}_0$ answered CQs.*

First, given u_1^{GUI}, MLCM constructs a cardinal training data set $D_{1,card}$ (see Section A). Next, MLCM fits the linear regression model $\mathcal{M}_1^{t=0}$ on $D_{1,card}$.[5] We see that $\mathcal{M}_1^{t=0}$ is able to perfectly fit

[5]We train the linear regression models using gradient descent, because (i) we restrict them to be monotone, i.e., their coefficients are constrained to be non-negative, and (ii) this also allows us to train them on both cardinal and ordinal data.

u_1^{GUI}. Next, MLCM generates the first CQ by comparing the two admissible schedules within budget that have the highest and second highest predicted utility with respect to $\mathcal{M}_1^{t=0}$, i.e., $\{2,4\}$ with $\mathcal{M}_1^{t=0}(\{2,4\}) = 122$ and $\{1,4\}$ with $\mathcal{M}_1^{t=0}(\{1,4\}) = 120$. When presenting this CQ to the student, she answers that she actually prefers $\{1,4\}$ with a true utility of $u_1(\{1,4\}) = 125$ over $\{2,4\}$ with true utility $u_1(\{2,4\}) = 110$. Next, MLCM updates the ordinal training data set as $D^1_{1,ord} = \{\{1,4\} \succ \{2,4\}\}$ and retrains the model $\mathcal{M}_1^{t=1}$ on the combined ranking and regression loss (see Section 6.3). We see that the regression coefficient for course 1 of $\mathcal{M}_1^{t=1}$ increased from 75 to 80 and the coefficient for course 2 decreased from 77 to 72. Now the model $\mathcal{M}_1^{t=1}$ correctly predicts the ordinal ranking of the above elicited CQ, i.e., $\mathcal{M}_1^{t=1}(\{1,4\}) = 125 > 117 = \mathcal{M}_1^{t=1}(\{2,4\})$ and thus has corrected the student's initial reporting mistake on her two most preferred courses. We see that after a single CQ, the student's utility for her (predicted) utility maximizing schedule $u_1(a_1^*)$ has already increased from 110 to 125. At this point, the first iteration of "binary search" (Line 4 in Algorithm 1) has been completed (here, only leading to one CQ). For the next CQ, MLCM selects $\{1,3\}$ – the schedule with the highest predicted utility that has not been elicited so far. Then MLCM performs binary search again, trying to insert $\{1,3\}$ into the already elicited list of schedules (Line 4 in Algorithm 1) resulting in the CQ consisting of the schedules $\{1,3\}$ and $\{1,4\}$.[6] Since the student prefers the schedule $\{1,3\}$, now the complete list of bundles $\{1,3\}$, $\{1,4\}$ and $\{2,4\}$ can be sorted and binary search is completed. After retraining with the updated ordinal training data set $D^2_{1,ord} = \{\{1,4\} \succ \{2,4\}, \{1,3\} \succ \{1,4\}, \{1,3\} \succ \{2,4\}\}$, the new model $\mathcal{M}_1^{t=2}$ represents the correct ordinal ranking of all admissible schedules and thus already found the true utility maximizing schedule $\{1,3\}$. The next schedule selected by MLCM is $\{2,3\} \notin D^2_{1,ord}$ and binary search is again performed. Since the student's answers do not contradict the model's predicted ordinal ranking, the linear coefficients of $\mathcal{M}_1^{t=3}$ and $\mathcal{M}_1^{t=4}$ remain unchanged.

For an example where the student has additionally forgotten some of her base values and how MLCM infers those missing base values, please see Appendix J.

4.3 Theoretical Properties of MLCM

Budish [2011] showed that A-CEEI satisfies *envy-bounded by a single good*, $(n+1)$*-maximin share guarantee*, and *Pareto efficiency*. If CM had access to the full true ordinal preferences, then the Stage 1 allocation of CM would also satisfy the same properties. In Appendix B, we prove that the same properties also hold in an *approximate* sense for the Stage 1 allocation of MLCM if the students' preferences are captured *approximately* via the ML models $\mathcal{M}_i$.

Incentives. Regarding incentives, Budish [2011] showed that A-CEEI is *strategyproof in the large (SP-L)*. Budish et al. [2017] argued that CM is also SP-L. Their argument proceeds in two steps. First, as the number of students increases, p^* calculated at the end of Stage 2 will be exogenous to the students' reports, and hence the students become price-takers. Second, since the market-clearing error target $\alpha \leq \sqrt{\sigma m}/2$ is independent of the number of students, the probability that Stage 3 affects a student's allocation goes to zero as the market grows. In MLCM, the main difference is that we use the trained ML models instead of the GUI reports when calculating prices. However, students are still price-takers, and the probability that Stage 3 affects an individual student still goes to zero. Therefore, we argue that MLCM is also SP-L.

5 COURSE ALLOCATION SIMULATION FRAMEWORK

In this section, we describe our student preference generator (Section 5.1), how we model students' reporting mistakes (Section 5.2), and how we calibrate the parameters to real-world data (Section 5.3). For additional details, please see Appendices C to E.

[6] When performing binary search, we break ties in favor of the schedule with the higher predicted utility.

5.1 Student Preference Generator

We have two goals for our student preference generator. First, students' preferences (and their reports) should be realistic, i.e., they should closely match real-world data on the usage of CM. Second, we must be able to encode each student's complete preferences as a MIP so that we can compute an (optimal) benchmark allocation given the students' true preferences.

Correlation. One of the key features the simulator must capture is some notion of *correlation* between students' preferences. To this end, we divide courses into *popular* and *unpopular*. Popular courses are those that many (but not necessarily all) students have a high value for. Concretely, for every student i, we randomly select a set of *favorite* courses from the set of popular courses. Then, for each course, student i's base value is drawn from some distribution, where the mean of that distribution is high for her favorite courses and low for all others. Note that a smaller number of popular courses implies higher correlation, as more students will have the same favorite courses. Thus, we can use the number of popular courses to control the degree of correlation.

Complementarities and Substitutabilities. To model complementarities and substitutabilities between courses, we build on prior experimental work modeling bidders in combinatorial auctions [Goeree and Holt, 2010, Scheffel et al., 2012]. Concretely, following Scheffel et al. [2012], we assume that courses lie on a latent space and that the distance between courses in that space defines the students' view on the complementarity/substitutability of those courses. Figure 2 depicts an example latent space with $M = \{1, \ldots, 30\}$. The set of popular courses are $M_p = \{8, 9, 21, 29\}$ and are marked with a star. Assume that student i's favorite courses are 9 and 21. From those, we randomly draw a set of *centers* – which is course 9 in Figure 2. All courses with L_1 distance smaller or equal to 1 from that center form the set of substitutabilities $\{9, 3, 8, 10, 15\}$. All courses with L_∞ distance smaller or equal to 1 from that center (that are not in the set of substitutabilities) and the center 9 form the set of complementarities $\{9, 2, 4, 14, 16\}$. The number of centers and the distances control the *degree* of complementarity and substitutability.

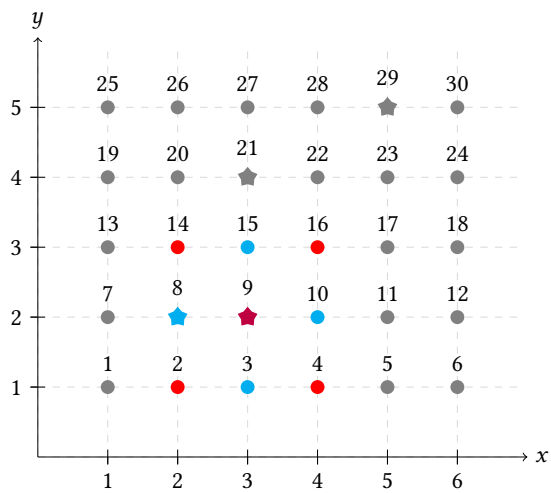


Fig. 2. A latent space with 30 courses.

Of course, one could use other topologies that result in the same or very similar preferences – and for the simulator, only the induced preferences matter. However, using our proposed common latent space enables a concise, mathematically tractable formalization of substitutabilities/complementarities. With this, our framework provides us with granular (and interpretable) control over the degree of complementarities/substitutabilities. As we will show in Section 5.3, this design allows us to produce instances very similar to those observed in practice by Budish and Kessler [2022].

Total utility calculation. We calculate a student's utility for a schedule based on her values for single courses and the number of courses from each substitutability/complementarity set in the schedule. In Appendix C, we provide mathematical details for the preference generator. In Appendix D, we provide the corresponding MIP formulation.

5.2 Students' Reporting Mistakes

Students can make mistakes when reporting their base value for single courses or when reporting pairwise adjustments. To capture these reporting mistakes, we use the following four parameters:

(1) $f_b \in [0, 1]$ denotes the probability that a student forgets one of her *base values*. We assume that a student forgets to report a base value for her lower-valued courses first. For example, if $M = \{1, 2, 3\}$, and a student values these courses as 2, 5 and 10, she will first forget course 1.
(2) $f_a \in [0, 1]$ denotes the probability a student forgets to report one of her adjustments (out of those adjustments for which she has not forgotten to report a base value for either of the courses in the pair). We assume that students forget adjustments uniformly at random.
(3) $\sigma_b \in \mathbb{R}_{\geq 0}$ denotes the standard deviation of the additive Gaussian noise $\mathcal{N}(0, \sigma_b^2)$ with which the students report their *base values*.
(4) $\sigma_a \in \mathbb{R}_{\geq 0}$ controls the support of the multiplicative uniform noise $\mathcal{U}[1 - \sigma_a, 1 + \sigma_a]$ with which the students report their *adjustment values*.

5.3 Calibration of the Simulation Framework

We calibrate the parameters of our framework (i.e., the preference generator and the reporting mistakes) to match the experimental results from Budish and Kessler [2022]. The key metrics from their experiment are:

(1) Students only report a base value for 49.9% of the courses.
(2) The number of courses with a reported value in $[50, 100]$ is approximately equal to the number of courses with a reported value in $[0, 50]$.
(3) Students report between 0 and 10 pairwise adjustments.
(4) Students report an average of 1.08 pairwise adjustments; the median is equal to 0.
(5) Students were asked to compare the schedule they received with several other course schedules. Their answers were consistent with their reported preferences in 84.41% of the cases, and in case of disagreements, the median utility difference, based on their reports, was 13.35%.

In Appendix E Table 5, we provide detailed results of our calibration procedure. In our experiments with 6 popular courses we set $(f_b, f_a, \sigma_b, \sigma_a) = (0.5, 0.4825, 17, 0.2)$, and with 9 popular courses we set $(f_b, f_a, \sigma_b, \sigma_a) = (0.5, 0.48, 23, 0.2)$. With these parameters, we are able to match metrics (1)-(3) exactly. For metric (4), our mean is 10% lower (in one of two settings), while our median is slightly larger (1 instead of 0). Regarding metric (5), both accuracy and scaled median utility difference are within 3 percentage points of the reported one. Thus, our framework produces instances very similar to those described in [Budish and Kessler, 2022]. See Appendix E for details.

6 MACHINE LEARNING INSTANTIATION

In this section, we instantiate the ML model used by MLCM.

6.1 Machine Learning Model Desiderata

There are three important desiderata for the choice of the ML model class. First, it needs to be sufficiently *expressive* to be able to learn students' complex preferences over course schedules (including complementarities and substitutabilities). Second, we must be able to train the ML model using both the *cardinal* input that students provide via the CM reporting language as well as the *ordinal* input from answering CQs. Third, in Phases 3–5 of MLCM, we must be able to compute each student's utility-maximizing schedule, given the ML model predictions, budget, and prices. More formally:

$$x \in \underset{x \in \Psi_i, x \cdot p \leq b_i}{\arg\max} \ \mathcal{M}_i(x) \tag{3}$$

Data Type	Ridge	nuSVR-gauss	xgboost	NN	MVNN
GUI	34.01±1.18	34.85±1.13	34.99±1.23	34.85±1.12	34.52±1.42
30 RAND VQs	19.36±1.04	21.76±0.81	23.01±0.88	23.25±0.88	18.64±0.99
50 RAND VQs	12.88±0.83	14.31±0.59	15.79±0.54	15.71±0.65	11.93±0.65
100 RAND VQs	9.87±0.68	9.40±0.50	15.75±0.55	9.32±0.56	5.84±0.53
150 RAND VQs	9.18±0.61	7.89±0.44	13.72±0.41	5.62±0.48	3.50±0.38

Table 2. Comparison of different ML models. Shown are MAEs on the test set and 95% CIs. Winners marked in grey.

Thus, a key requirement on the ML model $\mathcal{M}_i$ is that Equation (3) can be solved fast enough. For this, we adopt as a requirement that Equation (3) can be translated into a succinct MIP.

6.2 Generalization Performance

Next, we compare the generalization performance of different ML models to identify the best one for our domain.

Experiment Setup. As in the experiment by Budish and Kessler [2022], we consider a setting with 25 courses. We use our simulator to create 100 instances of student preferences, where we use 20 to tune the ML hyperparameters, and 80 for testing. We use two different types of training data sets. First, we use as the training set the cardinal data $\{D_{i,\text{card}}\}_{i \in N}$ generated using the same procedure as Phase 2 of MLCM (based on the students' GUI reports) (see Appendix A). Second, we use $\{30, 50, 100, 150\}$ *random* value queries as the training data set. To test the trained models, we use the complement of these training sets as the test sets and report the mean absolute error (MAE) and Kendall tau (KT) of each ML model. We use MAE and KT to determine the winners.

As ML models, we consider ridge linear regression (Ridge), nu-support vector regression with Gaussian kernel (nuSVR-gauss), XGBoost, neural networks (NN), and monotone-value neural networks (MVNNs). We provide a detailed description of our hyperparameter tuning procedure and the winner configurations in Appendix F.

Results. Table 2 presents MAE results for a setting with 9 popular courses (results for 6 popular courses, and for KT, are qualitatively similar; see Appendices G and H). We see that MVNNs have the best performance across all different training sets with regard to MAE and KT. For the GUI reports, we observe that all ML models perform on par. This can be explained by the fact that each ML model learns (approximately) the same quadratic preferences induced by the GUI on the training set and thus achieves out-of-sample the same test MAE. However, with random values queries as the training set, MVNNs significantly outperform the other models.

6.3 Integrating comparison queries into MVNNs

To simultaneously train MVNNs on GUI reports (regression data) and CQs (classification data), we use a method by Sculley [2010] called *combined ranking and regression (CRR).* CRR trains regression models using both regression and classification data. For this, CRR randomly alternates in each gradient step between a regression and a classification loss. Specifically, with probability $\alpha \in [0, 1]$, CRR selects a regression loss $l_{reg}(\cdot)$ and with probability $(1 - \alpha)$ a classification loss $l_{class}(\cdot)$, respectively. In case the classification loss is selected, the sigmoid of the difference between the predicted values for the two schedules included in the comparison query is interpreted as the probability with which the MVNN predicts that one schedule is better than the other, thus resulting

in a classification problem. To train MVNNs in our setting, we use CRR with MAE and binary cross entropy as the regression and classification loss, respectively. See Appendix I for details.

6.4 MIP-Formalizability of the Utility Maximization Problem

Weissteiner et al. [2022a] already provided a very succinct MIP formulation for MVNNs. It is straightforward to adopt their MIP formulation to the constraints of our setting, making the optimization of the student's utility maximization problem (i.e., eq. (3)) practically feasible.

Given that MVNNs satisfy all three desiderata for the choice of the ML model class we have laid out above (see Section 6.1), we adopt MVNNs as the ML model for MLCM.

7 EXPERIMENTAL EVALUATION

In this section, we experimentally evaluate the performance of MLCM using our simulation framework from Section 5. We provide the source code for all experiments in the supplementary material.

7.1 Experiment Setup

We consider a setting with 25 courses and 100 students.[7] We consider 6 and 9 popular courses (where 6 popular courses corresponds to extreme correlation). As in the lab experiment of [Budish and Kessler, 2022], every student wants a schedule of (at most) five courses. Thus, for 100 students the total demand is 500 seats. The *supply ratio* (SR) denotes the fraction of the total number of seats to the total demand. Thus, a SR of 1.5 means that there are a total of 750 course seats (split equally among all courses). We consider SRs of 1.25 and 1.5.

To compare CM and MLCM, we assume that both receive the same reports from the CM reporting language, with students' reporting mistakes calibrated as described in Section 5.3. For MLCM, each student additionally answers 1, 5, 10, 15, or 20 CQs. We denote these mechanisms as CM and MLCM (1/5/10/15/20 ML-BASED CQs), respectively. We assume that students make no mistakes when answering CQs.[8] We further consider four benchmarks. The first is CM (No Mistakes), which is CM, but assuming no reporting mistakes. The second is CM* (Full Preferences), which is a modified version of CM that takes as input the *full* correct cardinal preferences of the students. Importantly, CM* includes interactions between three or more courses, which cannot be expressed using CM's reporting language. Furthermore, we use a benchmark called MLCM (20 RANDOM CQs), where each student answers 20 *randomly* generated CQs. Finally, we also use random serial dictatorship (RSD) as another benchmark, with the GUI reports as input and the same amount of reporting mistakes.

We use the same 100 instances to test each mechanism. We use a maximum of 10 random restarts for the *tabu search* of Stage 1. Furthermore, we use three hidden layer MVNNs with 20 units per layer such that they provide a good trade-off between generalization performance and computational cost of the utility maximization MIP (Equation (3)). See Appendix K for details on the selected MVNN hyperparameters and the computing infrastructure. Unless otherwise noted, CQs denote *ML-based CQs*.

[7] We selected 25 courses to match the number of courses in the lab experiments of [Budish and Kessler, 2022], allowing us to properly calibrate the mistake profile of the students (see Section 5.3). However, since this calibration does not depend on the number of students, we increased the number of students from about 20 in [Budish and Kessler, 2022] to 100. Consequently, the course capacity increased from approximately 5 to 25, which means that every individual student's reports have a smaller impact on the (over-)demand for individual courses. This makes this setting more realistic and reduces variance during experimentation, while still keeping the runtime for our experiments manageable. Recall that the results from the lab experiments with 20 students per instance extrapolated well to Wharton with over 1700 students.

[8] Recall that pairwise comparison queries are known to have low cognitive load [Chajewska et al., 2000, Conitzer, 2009]. Furthermore, Budish and Kessler [2022] used the students' answers to pairwise comparison queries as *ground truth* for the same problem.

	Avgerage Student Utility			Minimum Student Utility			Overs.	Time
Mechanism	**Stage 1**	**Stage 2**	**Stage 3**	**Stage 1**	**Stage 2**	**Stage 3**	**Stage 1**	**in h**
CM* (Full Preferences)	100.0 ± 0.0	95.4 ± 0.9	98.1 ± 0.7	73.2 ± 0.9	70.9 ± 1.1	72.1 ± 1.0	7.2 ± 0.9	4.1
CM (No Mistakes)	98.6 ± 0.3	95.3 ± 0.7	97.4 ± 0.5	72.5 ± 0.9	70.5 ± 1.1	71.6 ± 1.0	5.7 ± 0.8	3.7
RSD	-	-	76.8 ± 0.5	-	-	29.8 ± 1.3	-	0.0
CM	79.6 ± 0.5	77.9 ± 0.6	79.0 ± 0.5	42.6 ± 1.2	41.5 ± 1.1	42.1 ± 1.1	2.9 ± 0.5	2.0
MLCM (1 ML-based CQ)	79.6 ± 0.5	77.4 ± 0.9	78.7 ± 0.8	41.3 ± 1.1	40.1 ± 1.2	41.2 ± 1.2	3.3 ± 0.5	23.2
MLCM (5 ML-based CQs)	83.9 ± 0.5	80.7 ± 1.2	82.1 ± 1.1	46.6 ± 1.3	44.0 ± 1.4	45.1 ± 1.5	3.0 ± 0.5	23.5
MLCM (10 ML-based CQs)	86.3 ± 0.5	82.8 ± 1.0	84.6 ± 0.9	50.5 ± 1.1	47.6 ± 1.3	48.3 ± 1.3	3.7 ± 0.6	26.0
MLCM (15 ML-based CQs)	88.0 ± 0.4	84.5 ± 0.9	86.4 ± 0.7	52.0 ± 1.3	49.2 ± 1.4	50.2 ± 1.4	4.3 ± 0.7	27.1
MLCM (20 ML-based CQs)	89.4 ± 0.5	86.4 ± 0.8	88.0 ± 0.7	53.3 ± 1.3	50.5 ± 1.4	51.4 ± 1.5	3.3 ± 0.5	27.4
MLCM (20 Random CQs)	78.9 ± 0.6	77.0 ± 0.8	78.2 ± 0.7	41.0 ± 1.2	40.3 ± 1.2	40.6 ± 1.2	3.2 ± 0.5	25.0

Table 3. Comparison of RSD, CM and MLCM (also using random comparison queries) in Stages 1–3 for a supply ratio of 1.25, 9 popular courses, and default parameterization for reporting mistakes. We normalize all results by the average utility of CM* after Stage 1. Shown are averages in % over 100 runs and 95% CIs. Additionally, we present the oversubscription (in number of seats) after Stage 1 (Overs.) and total runtime (in hours) per run.

7.2 Welfare Results

In Table 3, we present results for SR = 1.25 (which is very close to Wharton's SR; see [Budish and Kessler, 2022]) and 9 popular courses. The results are better for SR = 1.5 and worse for 6 popular courses (see Appendix L). We normalize all results by the average utility of CM* after Stage 1, so that all utility metrics can be reported in percent. The metrics of interest are the *average* and *minimum* student utility after Stage 3 (i.e., for the final allocation). We see that MLCM (10 ML-BASED CQs) significantly outperforms CM, both in average and minimum student utility.[9] In particular, MLCM (10 ML-BASED CQs) increases average utility from 79.0% to 84.6% (a 7.1% increase) and minimum utility from 42.1% to 48.3% (a 14.7% increase). As the number of CQs increases, the performance of MLCM improves further. Compared to CM, MLCM (20 ML-BASED CQs) increases average utility from 79.0% to 88.0% (an 11.4% increase) and minimum utility from 42.1% to 51.4% (a 22.1% increase).

Recall that, if a student answers zero CQs, then MLCM treats her preferences in the exact same way as CM (i.e., only using her GUI reports). Table 3 shows that, if the student answers a single CQ, then the performance of MLCM is statistically on par with CM's performance, and the performance only improves by increasing the number of CQs. Thus, MLCM never performs worse than CM. Additionally, we observe that asking 20 *random* CQs does *not* improve upon the performance of CM. This shows the importance of asking "smart" queries in MLCM. Furthermore, we see that switching from RSD to CM leads to a surprisingly small increase in average student utility, while it does substantially increase minimum student utility. This further highlights the performance of MLCM, since MLCM significantly improves both average and minimum student utility compared to CM. Finally, we observe that the performance difference between CM* (full preferences) and CM (No Mistakes) is not significant. This shows that our choice of hyperparameters for the simulation framework results in preferences that can be captured well by the CM language (which can capture at most pairwise interactions). Thus, the welfare improvements of MLCM over CM are primarily due to its ability to correct students' reporting mistakes, and not due to MLCM's ability to (in principle) capture more than pairwise interactions between courses.

[9]See Appendix M for statistical tests for all such statements.

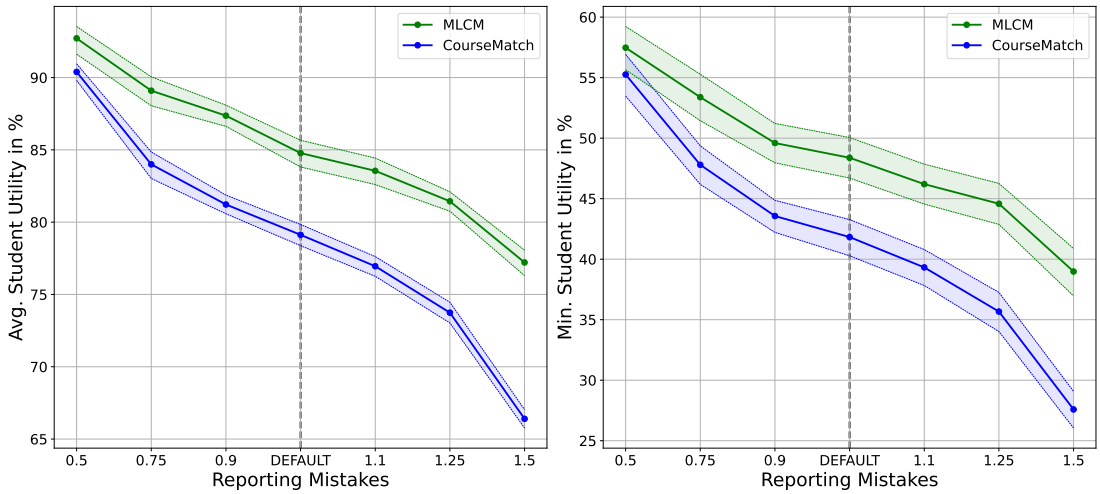


Fig. 3. Reporting mistakes ablation experiment for a supply ratio of 1.25 and 9 popular courses. Shown are average results in % for the final allocation over 50 runs including 95% CI.

7.3 Reporting Mistakes Ablation Study

We now vary how many reporting mistakes students make. For this, we keep the setting fixed (SR 1.25 and 9 popular courses) and multiply all parameters of the students' mistake profile (i.e., f_b, f_a, σ_b and σ_a) by a common constant γ. For $\gamma < 1$, students make fewer mistakes than in the default profile, while for $\gamma > 1$, the opposite is true. Importantly, γ does not linearly affect the students' mistakes. For example, for $\gamma = 0.5$, students make about 50% *fewer* mistakes compared to $\gamma = 1$, but the *severity* of these mistakes (utility difference) is only approximately 25% as large (see Appendix E). For each γ, we run the same 50 instances for CM and MLCM, with each student answering 10 CQs in MLCM.

Figure 3 shows the results of the ablation study (see Appendix N for SR = 1.5 and 6 popular courses). As γ increases, the performance of both, CM and MLCM, monotonically decreases. MLCM significantly outperforms CM for all $\gamma \in [0.5, 1.5]$. As γ increases, the relative performance gap between the two mechanisms gets significantly larger. For $\gamma = 1.5$, MLCM performs 16.2% and 41.3% better than CM in terms of average and minimum student utility, respectively. Those results could be further improved by retuning MLCM's hyperparameters for each value of γ.

7.4 Should Individual Students Opt Into the ML Feature?

Suppose that MLCM is implemented in practice, and an individual student must decide whether to opt into MLCM's ML feature or not. How much would the student benefit, if (a) no other student opted into the ML feature, or (b) if everyone else also opted in?

No Other Student to Opt into the ML Feature. We first consider the scenario where the student in question is considering opting in, and no one else does so. We study this by running both MLCM and CM twice – once where no students have opted in, and once where a single student has done so.[10] We use 20 instances and 100 students per instance. We report averages over those 2000 students. Table 4 shows the results for a SR of 1.25. We observe that the *expected relative gain* from opting in is at least 8.5% (across all settings). Furthermore, the student prefers the "MLCM schedule" in at

[10]Following [Budish and Kessler, 2022], we report results after Stage 1. Furthermore, to make the experiment computationally feasible, for each setting, we use the Stage 1 price vector that would result if no student would opt in (assuming that the student who is considering to opt in is a price taker).

Setting			Preferred Mechanism			Gain from Opting Into MLCM		
SR	#PoP	#CQs	MLCM	CM	Indiff.	Expected	if pref MLCM	if pref CM
1.25	9	10	72.70%	5.95%	21.35%	11.1%	16.1%	-10.8%
1.25	9	15	78.75%	4.25%	17.00%	13.5%	17.6%	-8.6%
1.25	9	20	83.15%	4.70%	12.15%	15.5%	19.0%	-10.1%
1.25	6	10	67.00%	9.30%	23.70%	8.5%	13.8%	-8.7%
1.25	6	15	73.20%	8.05%	18.75%	9.8%	14.4%	-8.3%
1.25	6	20	78.75%	6.70%	14.55%	11.4%	15.1%	-7.7%

Table 4. Expected gain of opting into MLCM's ML feature when no other student opts in. Shown are average results across 2000 students per setting (SR,#PoP,#CQs). CIs for all metrics are ≈ 0.

least 67% of the cases, while she prefers the "CM schedule" in at most 9.3% of the cases.[11] As the number of CQs the student answers increases, the benefit from opting into MLCM's ML feature becomes even larger. Finally, the improvement is larger for more popular courses and for a larger SR (see Appendix O for SR 1.5).

In Appendix O, we perform an analogous experiment for the case when all other students opt into MLCM's ML feature. These results (i.e., the expected utility gains) are almost identical.

7.5 Results for Additive Preferences

In this subsection, we test the robustness of our approach to changes in the *true* students' preferences. Specifically, we repeat all experiments described in the previous subsections for the simple case of students having *additive* true preferences.

We use our student preference generator (see Section 5.1 and Appendix C) to generate additive student utility functions and calibrate the mistake profile of the students so that both their accuracy and the reported utility difference in case of disagreements match those determined in the lab experiment of Budish and Kessler [2022] (see Table 36 in Appendix P). With these preferences and mistake profiles, we repeat the welfare experiment described in Section 7.2 for supply ratios of 1.25 and 1.5, the reporting mistakes ablation experiment described in Section 7.3, as well as the experiment investigating the expected gain of a student opting into MLCM's ML feature described in Section 7.4. The results of those experiments are qualitatively very similar to those for the original preferences and can be found in Appendix P. For example, for the most realistic supply ratio of 1.25, MLCM (10 ML-BASED CQs) increases average student utility by 7.0% and minimum student utility by 23.1%. Overall, these results show the robustness of our design and its applicability to a large range of settings.[12]

7.6 Scaling to Schools with more Courses

All experiments in Section 7 were performed in settings with 25 courses in order to match the experimental setup in [Budish and Kessler, 2022]. As the number of courses increases, the time required to solve the MIP that determines a student's most preferred schedule at a given price vector, which is required both for the calculation of the final allocation as well as the ML preference

[11] In the lab experiment of [Budish and Kessler, 2022], they compared CM to the previous mechanism used at Wharton, BPA. In those experiments, 42.4% of the students preferred the allocation under CM, 31.8% preferred BPA, while the remaining students were indifferent.

[12] Note that we did not perform hyperparameter optimization for this new setting, but instead we used the hyperparameters determined for the original setting described in Section 7.1, which further illustrates the robustness of our approach.

elicitation according to Algorithm 1, increases. This could raise some concerns about MLCM's applicability to a large school with hundreds of courses.

To test whether this is indeed an issue, we conducted the following experiment: Using our preference generator (Section 5.1), we generated problem instances varying the number of courses from 25 up to 350. For each problem instance, we trained an MVNN on the GUI reports of each student and measured the time required to determine that student's most preferred course schedule for a given price vector. For each number of courses, we tested 100 different students, each on the same 10 random price vectors. In Figure 4, we plot the average MIP solution times as well as their 95% confidence intervals.

From Figure 4, it becomes clear how gracefully the MIP solution time, for our choice of ML model, scales with the number of courses.[13] Concretely, a 14-fold increase in the number of courses, from 25 to 350, does not even lead to a doubling of the average MIP solution time. Thus, the runtime required for finding an A-CEEI would increase by at most a factor of 2, assuming that the number of price vectors evaluated by tabu search remains constant.[14] Regarding the real-time component of MLCM (i.e., generating pairwise comparison queries), note that even with 350 courses, we can still generate the query within at most two seconds (where most of the time is required for retraining the ML model, and not for solving the MIP).

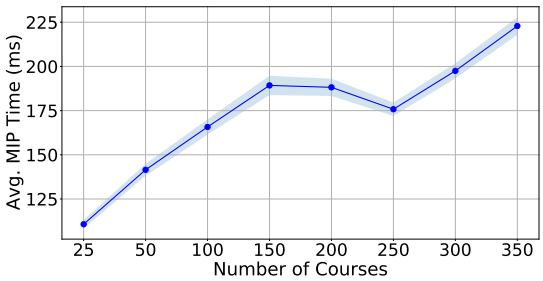


Fig. 4. MIP solution times for MVNNs as a function of the number of courses. Shown are average results over 100 students and 10 random price vectors including 95% CI.

8 DISCUSSION

For our most realistic setting, our results show that, compared to CM, MLCM with 10 comparison queries per student increases average student utility by over 7% and minimum student by almost 15%. Furthermore, the a priori expected utility gain of a student answering 10 comparison queries is over 10%. To put these numbers into perspective, consider that CM increased average student utility compared to RSD by less than 3% (while the increase in minimum student utility was large). Given this, we consider the improvements achieved by MLCM to be very large.

To realize these gains, the students incur additional costs, as they have to answer some CQs on top of reporting their preferences to the GUI. However, given that students are already encouraged to input their cardinal values to the GUI for at least 10 courses [Wharton, 2020], and given that pairwise comparison queries have a comparatively lower cognitive cost [Chajewska et al., 2000, Conitzer, 2009], we consider the relative increase in cost justified, given the large utility improvements.

Recall that MLCM uses a separate ML model for each student. An alternative approach would have been to instead use a *single* neural network for all students, leveraging possible similarities between students in the learning task. However, this approach might open new possibilities for

[13] For this test, we used the same MVNN architecture as for all experiments in Section 7, only changing the input layer to support the additional number of courses. Given that the structure of the students' preferences does not become significantly more complicated when increasing the number of courses (as suggested by Wharton's CM user manual, encouraging students willing to take up to 5 courses to report a base value for at least 10 courses, even though the school offers over 300 courses per semester [Wharton, 2020]), our original network architecture is still able to capture the students' preferences.

[14] Additionally, recall that the new algorithm for computing an A-CEEI that was introduced by Budish et al. [2023] (in work concurrent to the present paper) is multiple orders of magnitudes faster than the tabu search currently in use, which will likely remove any runtime concerns regarding finding an A-CEEI in the near future.

strategic manipulations since a student could then influence other students' utility representations via her reports. Additionally, this design would not allow for the asynchronous interaction paradigm between students and the mechanism (where every student can decide when to answer their CQs), which was a major consideration in our design.

An important element of MLCM is that we use the trained ML models, not only to generate CQs in the iterative preference elicitation phase, but also to determine the final allocation. An alternative approach would be to only use the trained ML models to *suggest corrections* to each student's GUI reports, which the student could then either accept or reject. With this approach, one would then run the original CM mechanism with the "corrected" GUI reports to determine the final allocation. While this approach might seem intuitive at first, it has multiple drawbacks. The main one is that this approach would require significantly more interaction by the students with the mechanism: the students would have to first report their preferences to the GUI, then answer some queries, and finally go back to correct their original GUI reports. This last step might be a task that is cognitively too demanding for most students, given that it involves reviewing "suggested corrections" for possibly tens of courses, which are all in conflict with what the student originally reported via the GUI. Additionally, since it requires that students enter their cardinal base values and adjustments a second time, this might introduce new reporting mistakes. Finally, this approach does not address the fact that the CM language may not be able to fully capture every student's preferences.

Some universities might prefer piloting our approach while making minimal changes to an already existing implementation of CM at their institution. To enable this, one could modify MLCM in Phases 3 and 5, *projecting* the learned ML models back into the original GUI language. Consequently, one could then simply use the original CM mechanism to find the A-CEEIs at the end of Phases 3 and 5, without the need for any changes to the core of the CM implementation. However, defining such a projection without a significant performance loss is non trivial but an interesting direction for future work. This approach would also not be able to capture preferences with more than pairwise interactions between courses.

9 CONCLUSION

In this paper, we have introduced the *machine learning-powered course match (MLCM)* mechanism. MLCM extends the well-known CM mechanism with an iterative, ML-based preference elicitation component. We have shown experimentally that MLCM significantly improves average and minimum student utility compared to CM across a wide range of settings.

The main challenge was to design a mechanism that can handle both cardinal and ordinal input from students, that does not *require* students to participate in the ML-powered preference elicitation phase, and that can handle asynchronous interactions of students while at the same time keeping computational costs in check.

In contrast to prior work on course allocation, we have focused on alleviating the students' reporting mistakes when declaring their preferences to the mechanism. We have found that, in realistic scenarios, the impact of the students' reporting mistakes on welfare can be even larger than the mechanism choice itself, highlighting the importance of correcting these mistakes. The main driving force behind our results is the careful selection of CQs, which alleviate the students' reporting mistakes in the most important area of the bundle space: bundles within the students' budgets, for which they have a high value.

There are several interesting avenues for future work. First, it seems promising to investigate the design of the query generation procedure in more depth. The most promising approach seems to be to deploy ML models for capturing model uncertainty (e.g., [Heiss et al., 2022]), to capture uncertainty regarding the students' utility for not yet queried course schedules. With such a model

in place, more principled query generation procedures based on *expected improvement* and *expected value of information* could be explored.

Second, it would be worthwhile to run a field experiment to empirically evaluate the performance of MLCM. Our design makes it particularly easy for universities to pilot MLCM, given that each student can decide whether to use the ML feature or not. It would be interesting to see how many students would opt into the ML feature and how many comparison queries they would answer.

Third, CM has already been perceived by some students as "black-box", i.e., non-transparent [Budish and Kessler, 2022]. MLCM's ML components could potentially amplify that sentiment. From a user experience standpoint, it would be worthwhile to investigate how to best explain to the users what the ML algorithm has learned about them and why (e.g., "because of your reply to query X, we believe that your utility for course A is lower than what you reported in the GUI"). Adding such explainability features to the interface would likely make students more comfortable with the ML feature of the mechanism, increasing its adoption, and thus ultimately increasing welfare.

Finally, we would like to highlight that our ML-powered preference elicitation approach using comparison queries could in principle be applied to a large variety of combinatorial assignment and matching problems. For example, one future application we envision is refugee resettlement. A recent stream of papers has treated refugee resettlement as a matching problem, taking into account the local communities' preferences as well as the refugees' preferences [Bansak et al., 2018, Delacrétaz et al., 2019]. Thus, one interesting direction for future work would be to investigate how an ML-powered preference elicitation approach could be used to help refugees better report their preferences to the matching mechanism (e.g., regarding employment opportunities, localities, and proximity to previously resettled family members), ultimately leading to a more efficient and fairer matching between refugees and local communities.

ACKNOWLEDGEMENTS

We would like to thank Craig Boutilier, Eric Budish, Yash Kanoria, Ben Lubin, and Hongyao Ma for very helpful discussions on this research project. We are also thankful for the feedback we received from participants at the 2022 INFORMS Workshop on Market Design. We thank the anonymous reviewers for helpful comments. Any errors remain our own. This paper is part of a project that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (Grant agreement No. 805542).

REFERENCES

Nir Ailon. 2012. An Active Learning Algorithm for Ranking from Pairwise Preferences with an Almost Optimal Query Complexity. *J. Mach. Learn. Res.* 13, 1 (jan 2012), 137–164. 4

Nir Ailon, Ron Begleiter, and Esther Ezra. 2011. A New Active Learning Scheme with Applications to Learning to Rank from Pairwise Preferences. 4

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. Association for Computing Machinery, New York, NY, USA, 2623–2631. 31

Lawrence M Ausubel, Peter Cramton, and Paul Milgrom. 2006. *The clock-proxy auction: A practical combinatorial auction design*. MIT, Cambridge, US. 120–140 pages. 2

Eduardo M Azevedo and Eric Budish. 2019. Strategy-proofness in the large. *The Review of Economic Studies* 86, 1 (2019), 81–116. 3

Kirk Bansak, Jeremy Ferwerda, Jens Hainmueller, Andrea Dillon, Dominik Hangartner, Duncan Lawrence, and Jeremy Weinstein. 2018. Improving refugee integration through data-driven algorithmic assignment. *Science* 359 (01 2018), 325–329. https://doi.org/10.1126/science.aao4408 19

Martin Bichler and Soeren Merting. 2021. Randomized scheduling mechanisms: Assigning course seats in a fair and efficient way. *Production and Operations Management* 30, 10 (2021), 3540–3559. 4

Avrim Blum, Jeffrey Jackson, Tuomas Sandholm, and Martin Zinkevich. 2004. Preference elicitation and query learning. *Journal of Machine Learning Research* 5, Jun (2004), 649–667. 2

Edwin V. Bonilla, Shengbo Guo, and Scott Sanner. 2010. Gaussian Process Preference Elicitation. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems - Volume 1* (Vancouver, British Columbia, Canada) *(NIPS'10)*. Curran Associates Inc., Red Hook, NY, USA, 262–270. 4

Steven J Brams and Philip D Straffin Jr. 1979. Prisoners' dilemma and professional sports drafts. *The American Mathematical Monthly* 86, 2 (1979), 80–88. 3

Gianluca Brero, Benjamin Lubin, and Sven Seuken. 2017. Probably Approximately Efficient Combinatorial Auctions via Machine Learning. *Proceedings of the AAAI Conference on Artificial Intelligence* 31, 1 (Feb. 2017), 397–405. https://doi.org/10.1609/aaai.v31i1.10624 2

Gianluca Brero, Benjamin Lubin, and Sven Seuken. 2018. Combinatorial Auctions via Machine Learning-Based Preference Elicitation. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence* (Stockholm, Sweden) *(IJCAI'18)*. AAAI Press, Stockholm, Sweden, 128–136. 2

Gianluca Brero, Benjamin Lubin, and Sven Seuken. 2021. Machine learning-powered iterative combinatorial auctions. 2

Eric Budish. 2011. The combinatorial assignment problem: Approximate competitive equilibrium from equal incomes. *Journal of Political Economy* 119, 6 (2011), 1061–1103. 3, 5, 9, 23, 24, 25, 26

Eric Budish, Gérard P Cachon, Judd B Kessler, and Abraham Othman. 2017. Course match: A large-scale implementation of approximate competitive equilibrium from equal incomes for combinatorial allocation. *Operations Research* 65, 2 (2017), 314–336. 1, 3, 5, 9

Eric Budish and Estelle Cantillon. 2012. The multi-unit assignment problem: Theory and evidence from course allocation at Harvard. *American Economic Review* 102, 5 (2012), 2237–71. 1, 3

Eric Budish, Ruiquan Gao, Abraham Othman, Aviad Rubinstein, and Qianfan Zhang. 2023. Practical algorithms and experimentally validated incentives for equilibrium-based fair division (A-CEEI). Unpublished manuscript. 7, 17

Eric Budish and Judd B Kessler. 2022. Can market participants report their preferences accurately (enough)? *Management Science* 68, 2 (2022), 1107–1130. 1, 3, 10, 11, 12, 13, 14, 15, 16, 19, 27, 31, 32, 36, 43, 44

Urszula Chajewska, Daphne Koller, and Ronald Parr. 2000. Making Rational Decisions Using Adaptive Utility Elicitation. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*. AAAI Press, Austin, USA, 363–369. 2, 13, 17

Wei Chu and Zoubin Ghahramani. 2005. Preference Learning with Gaussian Processes. In *Proceedings of the 22nd International Conference on Machine Learning* (Bonn, Germany) *(ICML '05)*. Association for Computing Machinery, New York, NY, USA, 137–144. https://doi.org/10.1145/1102351.1102369 4

Vincent Conitzer. 2009. Eliciting single-peaked preferences using comparison queries. *Journal of Artificial Intelligence Research* 35 (2009), 161–191. 2, 13, 17

David Delacrétaz, Scott Duke Kominers, and Alexander Teytelboym. 2019. *Matching Mechanisms for Refugee Resettlement*. Working Papers 2019-078. Human Capital and Economic Opportunity Working Group. https://ideas.repec.org/p/hka/wpaper/2019-078.html 19

Franz Diebold, Haris Aziz, Martin Bichler, Florian Matthes, and Alexander Schneider. 2014. Course allocation via stable matching. *Business & Information Systems Engineering* 6, 2 (2014), 97–110. 3

Fred Glover, Manuel Laguna, and Rafael Martí. 2018. Principles and strategies of tabu search. In *Handbook of Approximation Algorithms and Metaheuristics, Second Edition*. Chapman and Hall/CRC, London, UK, 361–377. 5

Jacob K Goeree and Charles A Holt. 2010. Hierarchical package bidding: A paper & pencil combinatorial auction. *Games and Economic Behavior* 70, 1 (2010), 146–169. 10, 27

Shengbo Guo and Scott Sanner. 2010. Real-time Multiattribute Bayesian Preference Elicitation with Pairwise Comparison Queries. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 9)*, Yee Whye Teh and Mike Titterington (Eds.). PMLR, Chia Laguna Resort, Sardinia, Italy, 289–296. https://proceedings.mlr.press/v9/guo10b.html 4

John William Hatfield. 2009. Strategy-proof, efficient, and nonbossy quota allocations. *Social Choice and Welfare* 33, 3 (2009), 505–515. 3

Jakob M Heiss, Jakob Weissteiner, Hanna S Wutte, Sven Seuken, and Josef Teichmann. 2022. NOMU: Neural Optimization-based Model Uncertainty. In *Proceedings of the 39th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 162)*. PMLR, Baltimore, USA, 8708–8758. https://proceedings.mlr.press/v162/heiss22a.html 4, 18

Sebastien M. Lahaie and David C. Parkes. 2004. Applying Learning Algorithms to Preference Elicitation. In *Proceedings of the 5th ACM Conference on Electronic Commerce* (New York, NY, USA) *(EC '04)*. Association for Computing Machinery, New York, NY, USA, 180–188. https://doi.org/10.1145/988772.988800 2

Abraham Othman, Christos Papadimitriou, and Aviad Rubinstein. 2016. The complexity of fairness through equilibrium. *ACM Transactions on Economics and Computation (TEAC)* 4, 4 (2016), 1–19. 5

Szilvia Pápai. 2001. Strategyproof and nonbossy multiple assignments. *Journal of Public Economic Theory* 3, 3 (2001), 257–271. 3

Tobias Scheffel, Georg Ziegler, and Martin Bichler. 2012. On the impact of package selection in combinatorial auctions: an experimental study in the context of spectrum auction design. *Experimental Economics* 15, 4 (2012), 667–692. 10, 27

D. Sculley. 2010. Combined Regression and Ranking. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Washington, DC, USA) *(KDD '10)*. Association for Computing Machinery, New York, NY, USA, 979–988. https://doi.org/10.1145/1835804.1835928 12, 35

Tayfun Sönmez and M Utku Ünver. 2010. Course bidding at business schools. *International Economic Review* 51, 1 (2010), 99–123. 1, 3

Jakob Weissteiner, Jakob Heiss, Julien Siems, and Sven Seuken. 2022a. Monotone-Value Neural Networks: Exploiting Preference Monotonicity in Combinatorial Assignment. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*. International Joint Conferences on Artificial Intelligence Organization, Vienna, AUT, 541–548. https://doi.org/10.24963/ijcai.2022/77 Main Track. 3, 4, 13

Jakob Weissteiner, Jakob Heiss, Julien Siems, and Sven Seuken. 2023. Bayesian Optimization-based Combinatorial Assignment. *Proceedings of the AAAI Conference on Artificial Intelligence* 37 (2023). 4

Jakob Weissteiner and Sven Seuken. 2020. Deep Learning—Powered Iterative Combinatorial Auctions. *Proceedings of the AAAI Conference on Artificial Intelligence* 34, 02 (Apr. 2020), 2284–2293. https://doi.org/10.1609/aaai.v34i02.5606 4

Jakob Weissteiner, Chris Wendler, Sven Seuken, Ben Lubin, and Markus Püschel. 2022b. Fourier Analysis-based Iterative Combinatorial Auctions. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*. International Joint Conferences on Artificial Intelligence Organization, Vienna, AUT, 549–556. https://doi.org/10.24963/ijcai.2022/78 Main Track. 4

Wharton. 2020. *Course Match User Manual*. Wharton. https://mba-inside.wharton.upenn.edu/wp-content/uploads/2020/08/Course-Match-manual-08152020.pdf 17

APPENDIX

A CARDINAL DATASET FROM STUDENTS' GUI REPORTS

In this section, we describe the process of building the cardinal dataset $D_{i,\text{card}}$ for each student $i \in N$, given their GUI reports.

From a conceptual standpoint, our idea for incorporating the students' reports in the learning framework is straightforward: For every student i, we want to use their reports to the GUI and create a bootstrap dataset $D_{i,\text{card}}$ where every datapoint is a tuple of the form (bundle, value). $D_{i,\text{card}}$ is then used to initialize $\mathcal{M}_i$.

As explained in Section 3.3, in the current implementation of Course Match, the students have two means at their disposal to express their preferences: base values and pairwise adjustments. Based on the student i's report, there are three distinct sets of bundles we can generate for our dataset $D_{i,\text{card}}$:

- **Bundles of type 1 (T1):** This set specifically includes two kind of bundles:
  (1) Bundles containing a single course, for which the corresponding student has declared a base value.
  (2) Bundles containing two courses for which the corresponding student has declared both base values plus the corresponding pairwise adjustment.

  The value of each such bundle is calculated exactly as in the current Course Match implementation, i.e., by summing the reported base values plus the reported adjustments whose courses are contained in that bundle.

  These bundles constitute the most solid pieces of information in our dataset, as for any bundle of this type, we can be sure that there cannot be any terms that the student did not report, either because they forgot to, or because it was impossible to do so in the CM reporting language.
- **Bundles of type 2 (T2)**: Each bundle of T2 contains five courses, for all of which the student reported a base value but not necessarily all corresponding pairwise adjustments. The value of each such bundle is calculated in the same way as in the current CM implementation, i.e., it is simply the sum of base values plus any adjustments reported by the student for the courses it contains.

  These bundles are important for two reasons: First, they push the distribution of the training set closer to the part of the space that matters in practice, a student participating in the mechanism is allocated a bundle of about five courses; not one or two. Additionally, these bundles implicitly "show" the learning model the combinatorial structure of the students' utility functions. For most ML algorithms tested, training them without bundles of this type lead to significantly worse generalization performance.
- **Bundles of type 3 (T3)**: Each bundle of this type contains five courses, but for at least one of them, the student did not even submit a base value when reporting their preferences in the CM language. For every such course, we impute a base value for it. Specifically, the imputed value is the expected value of the prior value distribution of a course, but now conditioned on the fact that a student did not report a base value for it in the CM language. No additional pairwise adjustments are imputed that a student did not report. Having imputed those base values, the value of each bundle of T3 can be calculated in the same way as for T1 and T2.

  Concretely, suppose that a student didn't report a base value for a course. In the current implementation, this course will be treated as if the student had declared a zero base value for it. Thus, there are two possible explanations for a missing base value: Either the student actually had a zero value for the specific course, or they had a non-zero base value, but they actually forgot to report it. Given the assumption that students tend to forget to report a base

value for courses towards the lower-end of their valuation, we model the base value v of a course that wasn't reported by the student as:

$$v = \begin{cases} 0 & \text{w.p. } p_z \\ \sim U[l, h] & \text{w.p. } 1 - p_z \end{cases} \tag{4}$$

And thus its expected value is:

$$\mathbb{E}[v] = (1 - p_z)\left(\frac{h - l}{2} + l\right) = (1 - p_z)\,\frac{h + l}{2}. \tag{5}$$

In order to generate the bundles of T3 in our experiments, we set $p_z = 0$, $l = 0$, and h equal to the lowest non-zero reported base value.

B THEORETICAL PROPERTIES OF MLCM

In this section, we show that, if the preferences are captured *approximately* via the ML models $\mathcal{M}_i$, then the same theoretical properties as for the CM Stage 1 allocation (i.e, *envy-bounded by a single good*, $(n+1)$*-maxmin share guarantee*, and *Pareto efficiency*) also hold in an *approximate* sense for the Stage 1 allocation of MLCM. First, we recall our notation.

Notation. N is the set of students, n is the number of students, M is the set of courses, m is the number of courses, k is the maximum number of courses allowed in a bundle, $u_i(\cdot)$ is the true utility function for student i, x is a bundle of courses (i.e. course schedule), a an allocation, and a_i denotes the bundle student i receives in allocation a, i.e., student i's allocation. The set of feasible allocations is represented by $\mathcal{F}$. Furthermore, we slightly overload the notation and denote by a_i *both* the indicator vector representing the allocation for student $i \in N$ (i.e., $a_i \in \{0,1\}^m$) as well as the corresponding set (i.e., $a_i \in 2^M$). With this, we denote by $j \in a_i$ that $j \in M$ is contained in a_i and by $a_i \setminus \{j\}$ the allocation a_i without the course $j \in M$ (we also use this for other generic bundles x or x').

First, we define our notion of utility function approximation, approximate fairness and welfare.

Definition B.1 (ε-approximation of a utility function). *For $\varepsilon \geq 0$, a function $\hat{u}(\cdot)$ is an ε-approximation of the true utility function $u : \{0,1\}^m \to \mathbb{R}$ if*

$$\sup_{x \in \{0,1\}^m} |\hat{u}(x) - u(x)| < \varepsilon. \tag{6}$$

Definition B.2 (Envy ε-bounded by a single good). *An allocation a satisfies envy ε-bounded by a single good if for all $i, i' \in N$*

(1) $u_i(a_i) \geq u_i(a_{i'}) - \varepsilon$ or

(2) There exists some good $j \in a_{i'}$ such that $u_i(a_i) \geq u_i(a_{i'} \setminus \{j\}) - \varepsilon$.

That is, if student i envies student i', by removing some single good from student i''s bundle we can make i's envy at most ε.

Definition B.3 (l-maximin share, Budish [2011]). *Let Z^* denote a l-maximin split, i.e.,*

$$Z^* := \underset{\substack{\{z_1,\ldots,z_l\}: z_k \in \mathcal{X} \\ \sum_{k=1}^{l} z_{kj} \leq q_j}}{\arg\max} \left(\min_{k \in \{1,\ldots,l\}} u_i(z_k) \right). \tag{7}$$

Then, student i's l-maximin share $a^{MaxiMin,l,u_i}$ for her given utility function u_i is defined as

$$a^{MaxiMin,l,u_i} := \underset{z \in Z^*}{\arg\min}\, u_i(z). \tag{8}$$

In words, $a^{MaxiMin,l,u_i} \in \{0,1\}^m$ is the course schedule an student obtains when she selects the utility maximizing feasible partition consisting of l bundles of the set of all courses given that an adversary assigns her the worst bundle from that proposed partition.

Definition B.4 ((l,ε)-maximin share). *For any $\varepsilon \geq 0$ student i's (l,ε)-maximin share is any course schedule $a^{MaxiMin,(l,\varepsilon),u_i}$ for which student i has utility at most ε less than her maximin share, i.e,*

$$u_i(a^{MaxiMin,(l,\varepsilon),u_i}) \geq u_i\left(a^{MaxiMin,l,u_i}\right) - \varepsilon. \tag{9}$$

Definition B.5 ((l,ε)-maximin share guarantee). *Any feasible allocation $a = (a_i)_{i=1}^n \in \mathcal{F} \subset \mathcal{X}^n$ where* all *students $i \in N$ get a bundle a_i they weakly prefer to their (l,ε)-maximin share $a^{MaxiMin,(l,\varepsilon),u_i}$ (w.r.t. their true utility functions, i.e., $\{u_i\}_{i\in N}$) is said to satisfy the (l,ε)-maximin share guarantee.*

In Proposition B.6, we now prove our main theoretical result.

Proposition B.6. *Let $[a^*, b, p^*]$ be an (α,β)-A-CEEI calculated using the ε-approximation of the true utility functions $\{\hat{u}_i\}_{i\in N}$, then:*

(1) If $\beta \leq \frac{1}{k-1}$ with k being the maximum number of courses per student, then a^* satisfies envy 2ε-bounded by a single good w.r.t. the true utility functions $\{u_i\}_{i\in N}$. Moreover, this bound is tight.

(2) If there exists some $\delta \geq 0$ such that $p^* \in \mathcal{P}(\delta, b)$, [15] *and $\beta < (1-\delta n)/n(1+\delta)$, then a^* satisfies the $(n+1, 2\varepsilon)$-maximin share guarantee w.r.t. the true utility functions $\{u_i\}_{i\in N}$. Moreover, this bound is tight.*

(3) Given p^*, the true utility of every student $i \in N$ for the bundle she receives in the allocation a^* (i.e., $u_i(a_i^*)$) is within 2ε of her true utility for her most preferred bundle she could afford.

(4) Given p^*, the allocation a^* is 2ε-Pareto efficient w.r.t. the true utility functions $\{u_i\}_{i\in N}$, i.e., $\nexists\ a' \in \mathcal{F}$ such that $\forall i \in N$ it holds that:

$$u_i(a_i') \geq u_i(a_i^*) - 2\varepsilon. \tag{10}$$

Proof. (1) Let $\hat{u} = (\hat{u}_1, \hat{u}_2, \ldots \hat{u}_n)$ be the profile of learned utility functions. Using Theorem 3 of Budish [2011][16], we have that for those learned utility functions the allocation a^* satisfies envy bounded by a single good i.e., for any $i, i' \in S$ either:

(a) $\hat{u}_i(a_i^*) \geq \hat{u}_i(a_{i'}^*)$ or

(b) There exists some good $j \in a_{i'}$ such that $\hat{u}_i(a_i^*) \geq \hat{u}_i(a_{i'}^* \setminus \{j\})$.

Since $\hat{u}_i(\cdot)$ is an ε-approximation of the true utility function $u_i(\cdot)$ we have that in the first case:

$$u_i(a_i^*) + \varepsilon \geq \hat{u}_i(a_i^*) \geq \hat{u}_i(a_{i'}^*) \geq u_i(a_{i'}^*) - \varepsilon \implies u_i(a_i^*) \geq u_i(a_{i'}^*) - 2\varepsilon \tag{11}$$

Similarly in the second case:

$$u_i(a_i^*) + \varepsilon \geq \hat{u}_i(a_i^*) \geq \hat{u}_i(a_{i'}^* \setminus \{j\}) \geq u_i(a_{i'}^*) - \varepsilon \implies u_i(a_i^*) \geq u_i(a_{i'}^* \setminus \{j\}) - 2\varepsilon \tag{12}$$

From Equations (11) and (12) it follows immediately that a^* satisfies envy 2ε-bounded by a single good.

Next we provide an example that shows that the bound is tight. Assume that there are 3 courses, a, b and c with capacities $q_a = 2$ and $q_b = q_c = 1$. Moreover, assume that there are 2 students with the following utility functions:

- $u_1(\{a,b\}) = u_1(\{a,b,c\}) = 1$, and 0 for any other bundle.

[15] $\mathcal{P}(\delta, b) = \{p \in [0, \max_i b_i]^m : \sum_j p_j q_j \leq \sum_i b_i(1+\delta)\}$.

[16] To apply Theorem 3 of Budish [2011], we need $\beta \leq \frac{1}{k-1}$.

- $u_2(\{a\}) = u_2(\{b\}) = 0.5 - \varepsilon, u_2(\{a,c\}) = u_2(\{b,c\}) = 0.5 + \varepsilon, u_2(\{a,b\}) = u_2(\{a,b,c\}) = 1$, and 0 for any other bundle.

Take $\varepsilon, \varepsilon' > 0$ and the learned utility functions of the 2 students to be:
- $\hat{u}_1(\{a,b\}) = 1 - \varepsilon, \hat{u}_1\{a,b,c\} = 1$, and 0 for any other bundle.
- $\hat{u}_2(\{a\}) = \hat{u}_2(\{b\}) = \hat{u}_2(\{a,c\}) = \hat{u}_2(\{b,c\}) = 0.5$
  $\hat{u}_2(\{a,b\}) = \hat{u}_2(\{a,b,c\}) = 1$, and and 0 for any other bundle.

Then, a $(0,\beta)$-A-CEEI ($\beta \leq \frac{1}{k-1} = \frac{1}{2}$) can be formed with the following elements:
- $a_1^* = \{a,b,c\}, a_2^* = \{a\}$
- $b_1 = 1 + \beta, b_2 = 1$
- $p_a^* = \beta, p_b^* = 1, p_c^* = 0$.

In this $(0,\beta)$-A-CEEI, the envy of student 2 with respect to her true utility function is exactly 2ε-bounded by a single good:

$$u_2(a_2^*) = 0.5 - \varepsilon = u_2(a_1^* \setminus \{b\}) - 2\varepsilon. \tag{13}$$

Therefore, the bound is tight.

(2) Let $\hat{u} = (\hat{u}_1, \hat{u}_2, \ldots, \hat{u}_n)$ be the profile of learned utility functions. Using Theorem 2 of [Budish, 2011],[17] we have that for those learned utility functions the allocation a^* satisfies the $(n+1)$-maximin share guarantee. Thus, for any $i \in N$:

$$\hat{u}_i(a_i^*) \geq \hat{u}_i\left(a^{\text{MaxiMin},n+1,\hat{u}_i}\right) \tag{14}$$

$$\geq \hat{u}_i\left(a^{\text{MaxiMin},n+1,u_i}\right) \tag{15}$$

$$\geq u_i\left(a^{\text{MaxiMin},n+1,u_i}\right) - \varepsilon \tag{16}$$

where (14) follows from the definition of the $(n+1)$-maximum share guarantee for the A-CEEI w.r.t. the learned utilities $\{\hat{u}_i\}_{i\in N}$ and (16) is true because $\hat{u}_i$ is per assumption an ε-approximation of u_i. Then, we have that

$$u_i(a_i^*) + \varepsilon \geq \hat{u}_i(a_i^*) \geq u_i\left(a^{\text{MaxiMin},n+1,u_i}\right) - \varepsilon, \tag{17}$$

and therefore

$$u_i(a_i^*) \geq u_i\left(a^{\text{MaxiMin},n+1,u_i}\right) - 2\varepsilon. \tag{18}$$

Hence the allocation a^* satisfies the $(n+1, 2\varepsilon)$-maximin share guarantee with respect to the true utilities $\{u_i\}_{i\in N}$.

Next, we provide an example that shows this bound is tight. Assume that there are 4 courses, a, b, c and d with capacities $q_a = q_b = q_c = q_d = 1$. Moreover, assume that there are 2 students with utility functions:
- $u_1(\{a\}) = 1, u_1(\{b\}) = 0.5 + \varepsilon, u_1(\{c\}) = 0.5 - \varepsilon, u_1(\{d\}) = 0$ and $u_1(\{b,c\}) = u_1(\{b,d\}) = u_1(\{c,d\}) = 0.5 + \varepsilon$.
- $u_2(\{a\}) = 0.8, u_2(\{a,b\}) = u_2(\{a,d\}) = 0.9, u_2(\{a,b,d\}) = 1$ and 0 for any other bundle.

Furthermore, assume that the learned utility functions of the 2 students are given as follows:
- $\hat{u}_1(\{a\}) = 1, \hat{u}_1(\{b\}) = \hat{u}_1(\{c\}) = \hat{u}_1(\{b,c\}) = \hat{u}_1(\{b,d\}) = \hat{u}_1(\{c,d\}) = 0.5$ and 0 for any other bundle.
- $\hat{u}_2(\cdot) = u_2(\cdot)$.

Then, a $(0, 0.25)$-A-CEEI ($\beta = 0.25 \leq \frac{1-\delta n}{n(1+\delta)} \overset{\delta=0}{=} \frac{1}{2}$) is given by:
- $a_1^* = \{c\}, a_2^* = \{a,b,d\}$
- $b_1 = 1, b_2 = 1 + \beta = 1.25$

[17] To apply Theorem 2 of Budish [2011], we need the existence of $\delta \geq 0$ such that $p^* \in \mathcal{P}(\delta, b)$ and $\beta < (1-\delta n)/n(1+\delta)$.

- $p_a^* = 1.1, p_b^* = p_c^* = 0.15, p_d^* = 0$.

A maximin $(n+1)$-split for student 1 w.r.t. her true utility function is $\{\{a\}, \{b\}, \{c, d\}\}$ and her true utility for her $(n+1)$-maximin share is $0.5 + \varepsilon$. Hence, the utility of student 1 with respect to her true utility function is exactly 2ε less than her $(n+1)$-maximin share of the endowment:

$$u_1(a_1^*) = 0.5 - \varepsilon \tag{19}$$

$$= u_1\left(a^{\text{MaxiMin},n+1,u_1}\right) - 2\varepsilon. \tag{20}$$

(3) For any $i \in N$, from the definition of the (α, β)-A-CEEI, it holds that

$$a_i^* = \underset{x \in \{x' \in \mathcal{X} : p^* \cdot x' \leq b_i\}}{\arg\max} \hat{u}_i(x). \tag{21}$$

Therefore,

$$u_i(a_i^*) + \varepsilon \geq \hat{u}_i(a_i^*) \tag{22}$$

$$= \hat{u}_i\left(\underset{x \in \{x' \in \mathcal{X} : p^* \cdot x' \leq b_i\}}{\arg\max} \hat{u}_i(x)\right) \tag{23}$$

$$\geq \hat{u}_i\left(\underset{x \in \{x' \in \mathcal{X} : p^* \cdot x' \leq b_i\}}{\arg\max} u_i(x)\right) \tag{24}$$

$$\geq u_i\left(\underset{x \in \{x' \in \mathcal{X} : p^* \cdot x' \leq b_i\}}{\arg\max} u_i(x)\right) - \varepsilon \tag{25}$$

Hence, we have that

$$u_i(a_i^*) \geq u_i\left(\underset{x \in \{x' \in \mathcal{X} : p^* \cdot x' \leq b_i\}}{\arg\max} u_i(x)\right) - 2\varepsilon. \tag{26}$$

(4) Assume that a^* is not 2ε- Pareto efficient. This implies that there exists an $a' \in \mathcal{F}$, which is a 2ε-Pareto improvement of a^* in economy $\left(N, M, (q_j^*)_{j=1}^m, (u_i)_{i=1}^n\right)$, i.e., the utility of every student $i \in N$ in allocation a' is more than 2ε higher than her utility in a^*. By item 3, for every $i \in N$ it holds that $p^* \cdot a_i' > p^* \cdot a_i^*$. This implies that $\sum_i p^* \cdot a_i' > \sum_i p^* \cdot a_i^*$. This is a contradiction since prices are non-negative and a^* allocates all units of positive-priced goods. □

Intuitively, item 4 of Proposition B.6 says that if we fix the prices of A-CEEI, the students cannot aggregate their budgets and then spend them in such a way that they are all benefited by more than 2ε, w.r.t. their true utility function.

Using the wording of Budish [2011], an implication of item 4 of Proposition B.6 is that the allocation induced by an A-CEEI on the ε-approximate utility functions will not admit any 2ε-Pareto-improving trades among the students, but may admit Pareto-improving trades among sets of students and the administrator.

C STUDENT PREFERENCE GENERATOR

In this section, we describe in detail our proposed student preference generator.

Our design goal for the student preference generator is two-fold: First, it should be realistic. That is, the preferences combined with the modeling of students' mistakes should closely approximate

the metrics on reported preferences given in [Budish and Kessler, 2022]. Second, we should be able to formulate the generated preferences in a succinct MIP as this is required by all stages of CM.

We build our preference generator based on the following two assumptions. First, a single student's value for a course schedule depends on their value for every single course and the complementarities/substitutabilities between the courses within a bundle. Second, the high-valued courses amongst students are correlated and fall under the category of popular courses. This is because students have the same reasoning for considering a course to be high-valued. These reasons could be a popular topic, a good instructor, an interest in the same minors, etc.

Thus, a student's utility for a bundle of courses depends on the following two things:

D1 The *individual value* of each course in that bundle.
D2 The *complementarities and substitutabilities* between courses in that bundle.

Recall, that $M \neq \emptyset$ and and $N \neq \emptyset$ denote the set of all courses and the set of students, respectively. To imitate the effect of students having correlated high-valued courses, we choose $M_p \subseteq M$ to represent the *popular courses* amongst students and $M_{np} = M \setminus M_p$ to represent the *non-popular courses*. We also assume that the set of high-valued courses for each student is a subset of the popular courses. We call these the student's *favorite courses*. For each student, their *base value* for each course in their favorite and non-favorite courses is drawn independently from $U(l_p, u_p)$ and $U(l_{np}, u_{np})$, respectively, where $U(a, b)$ denotes the uniform distribution on the interval (a, b), $0 \leq l_{np} \leq l_p$, and $0 \leq u_{np} \leq u_p$.

To simulate the substitutabilities and complementarities between courses in students' preferences, we build on the prior work on preference generators for spectrum auctions (i.e., the Local Synergy Value Model (LSVM) [Scheffel et al., 2012] and the Global Synergy Value Model (GSVM) [Goeree and Holt, 2010]).

Specifically, we assume that the courses are arranged in a latent space where the complementarity and substitutability relation between courses is a function of their distance to each other, i.e.,

- If two courses are "too close", then the contents of the courses have too much overlap and hence they are substitutes for each other.
- If they are "close" but not "too close", they are complements.
- If they are "far away", they are neither complements nor substitutes to each other.

More precisely, we assume that $M \subset \mathbb{N}^2$ and that the set of complementarities and substitutabilities are centered around a subset of the popular courses M_p, called centers, i.e., $M_c \subseteq M_p$. Then the set of substitutabilities and complementarities around a center point $c \in M_c$ are defined using the L_1 and L_∞ distances as follows:

- *Set of substitutabilities:*

$$S_c = \{m \in M | L_1(m, c) \leq r_s\} \tag{27}$$

- *Set of complementarities:*

$$C_c = \{m \in M | L_\infty(m, c) \leq r_c\} \setminus S_c \cup \{c\} \tag{28}$$

where $r_s, r_c \in \mathbb{N}_{>0}$ are the radius of the set of substitutabilities and the radius of the set of complementarities, respectively. The larger these radii, the more complementarities/substitutabilities are present in the students' preferences. We illustrate an example of this latent space in Figure 5.

Figure 5, depicts a latent space of height 5 and width 6 with the courses $M = \{1, \ldots, 30\}$, where we assume that both radii are equal to 1, i.e., $r_s = r_c = 1$ and the set of popular courses is given as $M_p = \{8, 9, 21, 29\}$. Then, the set of substitutabilities and complementarities defined by the center 9 are given by $S_c = \{9, 3, 8, 10, 15\}$ and $C_c = \{9, 2, 4, 14, 16\}$, respectively.

To model the impact of complementarities/substitutabilities on students' preferences, we use piece-wise constant step functions $\psi_c : \{0, \ldots, |C_c|\} \to \mathbb{R}_+$ and $\xi_c : \{0, \ldots, |S_c|\} \to \mathbb{R}_-$ to represent

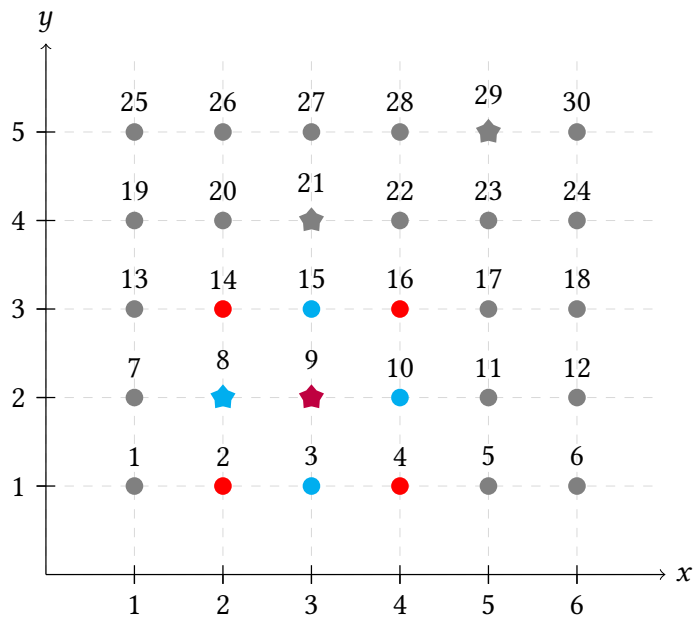


Fig. 5. A latent space with 30 courses.

the multiplicative bonuses and penalty factors for a bundle of courses if its items belong to the set of complementarities C_c or the set of substitutabilities S_c. We further assume that ψ_c is monotonically non-decreasing and ξ_c is monotonically non-increasing. For example, given a center c, $\psi_c(3)$ represents a student's multiplicative bonus for taking three courses from the set of complementarities C_c.

With all these building blocks, the utility of a student $i \in N$ for a bundle $x \in \{0,1\}^m$ is defined as:

$$\begin{aligned} u_i(x) = &\sum_{j \in x} u_i(\{j\}) \\ &+ \sum_{c \in M_c} \sum_{j \in C_c} \psi_c(\tau_c(x)) u_i(\{j\}) \\ &+ \sum_{c \in M_c} \sum_{j \in S_c} \xi_c(\kappa_c(x)) u_i(\{j\}), \end{aligned} \tag{29}$$

where $u_i(\{j\})$ represents the base values of student $i \in N$ for course j and $\tau_c(x) := |\{j \in x : j \in C_c\}|$ is the number of courses that belong both to the bundle x and the set of complementarities at center c, i.e., C_c. Analogously, $\kappa_c(x) := |\{j \in x : j \in S_c\}|$ is the number of courses that belong both to the bundle x and the set of substitutabilities at center c, i.e., S_c.

This preference generator enables to efficiently encode students' complementarities/substitabilities into a MIP (see Appendix D). Thus, it allows us to conduct experiments and compare MLCM with the original CM mechanism in a synthetic setting where we have access to the optimal allocations w.r.t. students' true preferences.

D MIP FOR THE STUDENT PREFERENCE GENERATOR

In this section, we provide a succinct MIP formulation for calculating the optimal allocation for a single student i given her preferences generated by our proposed student preference generator from Appendix C and a price vector p. This enables us to compute, for any price vector, the allocation under the students' true preferences. First, we define the individual student optimization problem.

Definition D.1 (Individual student utility optimization problem). *For student $i \in N$, utility function $u_i : \mathcal{X} \to \mathbb{R}_+$ defined in Equation (29), price vector $p \in \mathbb{R}_+^m$ and budget $b_i \in [1, 1+\beta]$, $\beta > 0$, we define the individual student utility optimization problem as*

$$\underset{\{x \in \Psi_i : x \cdot p \leq b_i\}}{\arg\max} \quad u_i(x) \tag{30}$$

In words, a solution of Equation (30) maximizes student i's true utility (as defined by our preference generator in Equation (29)) amongst all her permissible course schedules Ψ_i that are affordable at prices p when given a budget b_i.

Next, we provide in Proposition D.2 the equivalent MIP.

Proposition D.2 (Individual student MIP). *Using the notation from Appendix C, and Definition D.1 the individual student utility optimization problem defined in Equation (30) can be equivalently formulated as the following MIP:*

$$\arg\max_{x \in \Psi_i} \sum_{c \in M_c} \sum_{j \in C_c} \sum_{\tau=1}^{\bar{\tau}} \psi_c(\tau) \cdot G_{c,j,\tau} \cdot u_i(\{j\}) \tag{31}$$

$$+ \sum_{c \in M_c} \sum_{j \in S_c} \sum_{\tau=1}^{\bar{\tau}} \xi_c(\tau) \cdot J_{c,j,\tau} \cdot u_i(\{j\}) \tag{32}$$

$$+ \sum_{j \in [m]} x_j \cdot u_i(\{j\}) \tag{33}$$

s.t.

$$\sum_{\tau=1}^{\bar{\tau}} \tau \cdot G_{c,j',\tau} \leq \sum_{j \in C_c} x_j, \ \forall\ j' \in C_c, \forall\ c \in M_c \tag{34}$$

$$\sum_{\tau=1}^{\bar{\tau}} G_{c,j,\tau} \leq 1, \ \forall\ j \in C_c, \forall\ c \in M_c \tag{35}$$

$$\sum_{\tau=1}^{\bar{\tau}} \tau \cdot J_{c,j',\tau} \geq \sum_{j \in S_c} x_j, \ \forall\ j' \in S_c, \forall\ c \in M_c \tag{36}$$

$$\sum_{\tau=1}^{\bar{\tau}} J_{c,j,\tau} \leq 1, \ \forall\ j \in S_c, \forall\ c \in M_c \tag{37}$$

$$\sum_{j \in [m]} p_j \cdot x_j \leq b_i \tag{38}$$

where

- *$\bar{\tau}$ is the maximum number of courses in a schedule.*
- *M_c is the set of centers.*
- *C_c is the set of complementarities centered around c.*
- *S_c is the set of substitutabilities centered around c.*
- *$G_{c,j,\tau}$ denotes the binary variable, whether or not the course indexed by j, which belongs to the set of complementarities C_c is in x, while in total τ courses from that set are included in x.*
- *$J_{c,j,\tau}$ denotes the binary variable, whether or not the course indexed by j, which belongs to the set of substitutabilities S_c is in x, while in total τ courses from that set are included in x.*
- *b_i is the budget of student i.*

Proof. The first thing we need to show is that the problem is actually a MIP, i.e., all constraints are linear and all variables are either linear or integer. First, note that all variables are either linear or integer by their definition. Moreover, the same is true for all constraints shown, other than the $x \in \Psi_i$ constraint. Next, we show that this constraint can be encoded in a linear way.

There are 2 reasons a schedule x would not be permissible for a student i. The first one would be if it contained any courses that the student is not eligible for. Let E_i be the indexes of those courses for student i. The second constraint would be if it contained more than one courses that take place at the same time. Let H be the set of all course hours (also known as *time slots*, e.g. Wednesdays 10-11am), and T_h the set of indexes of courses that take place at time $h \in H$. Then, $x \in \Psi_i$ is equivalent to the following two *linear* constraints:

$$\sum_{j \in E_i} x_j \leq 0, \tag{39}$$

$$\sum_{j \in T_h} x_j \leq 1 \ \forall h \in H. \tag{40}$$

Constraint (39) enforces that student i cannot enroll in courses that are not permissible for her, while constraint (40) enforces that she cannot take more than one courses that take place in the same time slot. Now that we have proven that the problem defined above is a valid MIP respecting all constraints, we need to prove that its solution is a utility-maximizing course schedule for student $i \in N$, i.e, is equivalent to Equation (30).

If all binary variables $G_{c,j,\tau}$ and $J_{c,j,\tau}$ are set to the correct value for a given course schedule x, then Equations (31) to (33) encode the utility of course schedule x for a given student i, as the only additive terms not zeroed-out are those in Equation (29). Thus, to prove the correctness of the MIP, it suffices to prove that those binary variables are correctly set for any valid x and that the budget constraint is satisfied.

Constraint (37) enforces that for any $(c, j) \in M_c \times S_c$ pair at most one of the binary variables $J_{c,j,\tau}$ will be set to 1. This combined with constraint (36) enforce that if τ elements from set S_c are included in x, then for the $J_{c,j,\tau'}$ variable it must hold that $\tau' \geq \tau$. But since we have a maximization problem and the function $\xi_c(\cdot)$ is monotonically non-increasing, the smallest value of τ' that satisfies the above constraint will be set to one, thus $\tau' = \tau$. Finally, note that if $\xi_c(\tau') = \xi_c(\tau)$ for some $\tau' \geq \tau$, i.e. $\xi_c(\cdot)$ is at a constant part of its support, it can be the case that the maximizer sets $J_{c,j,\tau'} = 1$ instead of the $J_{c,j,\tau'} = 1$. However, since $\xi_c(\tau') = \xi_c(\tau)$, this is not a problem, as the objective value is the same in both cases.

Similarly, constraint (35) enforces that for any $(c, j) \in M_c \times C_c$ pair, at most one of the $G_{c,j,\tau'}$ variables will be set to one. Constraint (34) enforces that if τ of the elements of C_c are included in x, then it must hold that $\tau' \leq \tau$. But since this is a maximization problem and $\psi_c(\cdot)$ is monotonically non-decreasing, the solver will set exactly the variable $G_{c,j,\tau}$ to one where $\tau' = \tau$, as this maximizes the objective value out of all feasible solutions. If $\psi_c(\tau) = \psi(\tau')$, G_{c,j'_τ} could be set to one instead of $G_{c,j,\tau}$, but the same argument as above again applies.

Finally, constraint (38) enforces that a student cannot take any bundle with a price larger than her budget. □

E MISTAKE PROFILE CALIBRATION

In this section, we present details on our mistake profile calibration.

Results. Table 5 shows detailed results of our calibration procedure. For each row, we multiply the default reporting mistake parameters $f_b, f_a, \sigma_b, \sigma_a$ as defined in Section 5.3 by the common

constant γ. We see that our mistake calibration for the default common mistake constant $\gamma = 1$ (i.e, the rows marked in grey) closely matches the metrics reported in Budish and Kessler [2022].

Setting		#Courses with value			#Adjustments				Accuracy	If disagreement
#Pop	γ	> 0	$(0, 50)$	$[50, 100]$	Mean	Median	Min	Max	CQs	utility difference in %
9	0.5	18.75 ± 0.02	14.35 ± 0.07	4.40 ± 0.07	3.72 ± 0.12	3.0	0	17	0.93 ± 0.01	-3.81 ± 0.41
9	0.75	15.62 ± 0.02	10.31 ± 0.08	5.30 ± 0.08	2.06 ± 0.09	1.0	0	12	0.87 ± 0.01	-9.41 ± 0.62
9	0.9	13.75 ± 0.02	7.91 ± 0.09	5.84 ± 0.08	1.37 ± 0.07	1.0	0	11	0.84 ± 0.01	-13.10 ± 0.72
9	1.0	12.49 ± 0.02	6.32 ± 0.09	6.17 ± 0.09	0.98 ± 0.05	1.0	0	10	0.81 ± 0.01	-15.52 ± 0.76
9	1.1	11.20 ± 0.02	4.71 ± 0.09	6.49 ± 0.09	0.69 ± 0.04	0.0	0	7	0.78 ± 0.01	-18.11 ± 0.77
9	1.25	9.39 ± 0.02	2.58 ± 0.08	6.81 ± 0.08	0.37 ± 0.03	0.0	0	6	0.75 ± 0.01	-21.60 ± 0.80
9	1.5	6.20 ± 0.02	0.36 ± 0.04	5.84 ± 0.04	0.10 ± 0.01	0.0	0	3	0.66 ± 0.01	-26.78 ± 0.78
6	0.5	18.75 ± 0.02	13.71 ± 0.06	5.04 ± 0.06	3.83 ± 0.12	3.0	0	19	0.93 ± 0.01	-3.11 ± 0.37
6	0.75	15.62 ± 0.02	10.01 ± 0.07	5.61 ± 0.07	2.17 ± 0.09	2.0	0	13	0.87 ± 0.01	-8.02 ± 0.57
6	0.9	13.75 ± 0.02	7.80 ± 0.08	5.95 ± 0.07	1.47 ± 0.07	1.0	0	10	0.85 ± 0.01	-11.42 ± 0.67
6	1.0	12.49 ± 0.02	6.33 ± 0.08	6.16 ± 0.08	1.08 ± 0.06	1.0	0	10	0.83 ± 0.01	-13.34 ± 0.71
6	1.1	11.20 ± 0.02	4.82 ± 0.08	6.38 ± 0.08	0.73 ± 0.04	0.0	0	10	0.80 ± 0.01	-15.48 ± 0.75
6	1.25	9.39 ± 0.02	2.72 ± 0.08	6.67 ± 0.08	0.41 ± 0.03	0.0	0	10	0.76 ± 0.01	-19.12 ± 0.77
6	1.5	6.20 ± 0.02	0.38 ± 0.04	5.82 ± 0.04	0.11 ± 0.02	0.0	0	3	0.68 ± 0.01	-23.83 ± 0.78
[Budish and Kessler, 2022]		12.45	6.17	6.27	1.08	0	0	10	0.84	-13.35 ± 0.41

Table 5. Mistake profile calibration experiment for several settings defined by the number of popular courses (#Pop) and the common mistake constant γ compared to the experimental findings in [Budish and Kessler, 2022]. Our two default settings (i.e., $\gamma = 1$) are marked in grey. 2000 students in total. We show the number of courses with reported value in 3 distinct intervals, the mean, median, minimum and maximum number of adjustments in the students' reports, the accuracy of their reports as determined by asking CQs and the median scaled utility difference between the two schedules in a CQ in case of disagreement between the CQ answer and the reported preferences. Shown are average results and a 95% CI.

F HYPERPARAMETER OPTIMIZATION

In this section, we present details on our hyper-parameter optimization (HPO).

HPO Method. The parameter selection for all the models is done using the *optuna* hyper-parameter selection algorithm by Akiba et al. [2019]. The number of trials for optuna for all models were equal and was selected to be 100. The range of parameters for each model is presented in Table 6.

ML Models	Parameter	HPO-Range
NN, MVNN	Hidden Layers	[1,2,3]
	Units/Hidden Layer	[8, 16, 32, 50, 80, 120, 200, 300]
	Learning Rate	(1e-4, 1e-2)
	Epochs	(80,700)
	l_2 Regularization	(1e-15, 1e-3)
	Batch Size	[4, 8, 16, 30, 50]
XGBoost	Eta	(1e-5 , 0.2)
	Colsample Bytree	(0.4, 0.8)
	Gamma	(0, 0.02)
	Learning Rate	(0.03, 0.3)
	Max Depth	(2,6)
	N_Estimators	(10, 200)
	Subsample	(0.4, 0.6)
nuSVR	Nu	(0, 0.25)
	Gamma	(1e-4, 0.9)
Ridge	Alpha	(0,1)

Table 6. HPO-ranges used for the optuna algorithm.

Experiment Setup. As in the experiment by Budish and Kessler [2022], we consider a setting with 25 courses. We use our simulator to create 100 instances of student preferences, where we use 20 to tune the ML hyperparameters (HPs). In each of these 20 instances, we use the cardinal data $\{D_{i,\text{card}}\}_{i \in N}$ that MLCM generated based on the students' GUI reports as the training set, which we denote by GUI in the result tables (see Appendix A for details). Second, we use $\{30, 50, 100, 150\}$ *random* value queries (RAND VQs) each consisting of five courses as the training set. We used the MAE on the complement of the training set to select the reported hyper-parameters.

Results. In Tables 7 to 11 we present the selected HPs of the respective ML models for 6 popular courses. In Tables 12 to 16 we present the selected HPs of the respective ML models for 9 popular courses.

Data Type	Hidden Layers	Units per Hidden Layer	Learning Rate	Epochs	L2-Regularization	Batch Size
GUI	2	80	0.00074	102	0.00033	16
30 RAND VQs	1	200	0.00965	288	0.00062	16
50 RAND VQs	2	120	0.00666	80	0.00079	50
100 RAND VQs	3	300	0.00641	454	0.00100	4
150 RAND VQs	1	50	0.00646	197	0.00031	4

Table 7. Selected HPs for MVNN for 6 popular courses.

Data Type	Hidden Layers	Units/Hidden Layer	Learning Rate	Epochs	L2-Regularization	Batch Size
GUI	3	80	0.00710	545	0.00094	16
30 RAND VQs	2	8	0.00573	697	0.00088	4
50 RAND VQs	1	120	0.00956	538	0.00054	4
100 RAND VQs	2	8	0.00821	681	0.00043	8
150 RAND VQs	2	32	0.00989	371	0.00049	4

Table 8. Selected HPs for NN for 6 popular courses.

Data Type	Eta	Colsample Bytree	Gamma	Learning Rate	Max Depth	N_Estimators	Subsample
GUI	0.03	0.59	0	0.18	3	159	0.44
30 RAND VQs	0.13	0.64	0	0.13	2	140	0.50
50 RAND VQs	0.11	0.72	0	0.09	6	146	0.45
100 RAND VQs	0.17	0.47	0	0.18	4	170	0.59
150 RAND VQs	0.03	0.77	0	0.12	4	200	0.45

Table 9. Selected HPs for XGBoost for 6 popular courses.

Data Type	Nu	Gamma
GUI	0.24	0.05
30 RAND VQs	0.25	0.04
50 RAND VQs	0.25	0.04
100 RAND VQs	0.24	0.04
150 RAND VQs	0.24	0.05

Table 10. Selected HPs for nuSVR with Gaussian kernel for 6 popular courses.

G GENERALIZATION PERFORMANCE RESULTS WITH 25 COURSES AND 6 POPULAR COURSES

Experiment Setup. As in the experiment by Budish and Kessler [2022], we consider a setting with 25 courses. We use our simulator to create 100 instances of student preferences, where we use 20 to tune the ML hyperparameters (see Appendix F for details), and 80 for testing. We perform two

Data Type	Alpha
GUI	0.22
30 RAND VQs	0.39
50 RAND VQs	0.38
100 RAND VQs	0.41
150 RAND VQs	0.45

Table 11. Selected HPs for Ridge for 6 popular courses.

Data Type	Hidden Layers	Units/Hidden Layer	Learning Rate	Epochs	L2-Regularization	Batch Size
GUI	2	32	0.00979	342	0.00009	8
30 RAND VQs	1	200	0.00998	604	0.00070	30
50 RAND VQs	2	50	0.00586	414	0.00072	4
100 RAND VQs	3	50	0.00949	401	0.00085	8
150 RAND VQs	2	120	0.00165	305	0.00084	4

Table 12. Selected HPs for MVNN for 9 popular courses.

Data Type	Hidden Layers	Units/Hidden Layer	Learning Rate	Epochs	L2-Regularization	Batch Size
GUI	2	300	0.00669	247	0.00099	4
30 RAND VQs	2	300	0.00472	700	0.00096	16
50 RAND VQs	2	8	0.00730	591	0.00048	16
100 RAND VQs	1	50	0.00682	657	0.00078	16
150 RAND VQs	1	300	0.00496	367	0.00075	4

Table 13. Selected HPs for NN for 9 popular courses.

Data Type	Eta	Colsample Bytree	Gamma	Learning Rate	Max Depth	N_Estimators	Subsample
GUI	0.00	0.52	0	0.09	4	165	0.56
30 RAND VQs	0.18	0.68	0	0.14	6	112	0.40
50 RAND VQs	0.06	0.57	0	0.21	3	85	0.54
100 RAND VQs	0.10	0.64	0	0.10	2	189	0.42
150 RAND VQs	0.05	0.68	0	0.24	3	93	0.43

Table 14. Selected HPs for XGBoost for 9 popular courses.

Data Type	Nu	Gamma
GUI	0.22	0.05
30 RAND VQs	0.24	0.04
50 RAND VQs	0.24	0.04
100 RAND VQs	0.13	0.05
150 RAND VQs	0.25	0.04

Table 15. Selected HPs for nuSVR with Gaussian kernel for 9 popular courses.

experiments. First, we use the cardinal data $\{D_{i,\text{card}}\}_{i \in N}$ that MLCM generated based on the students' GUI reports as the training set, which we denote by GUI in the result tables (see Appendix A for

Data Type	Alpha
GUI	0.22
30 RAND VQs	0.22
50 RAND VQs	0.27
100 RAND VQs	0.36
150 RAND VQs	0.45

Table 16. Selected HPs for Ridge for 9 popular courses.

details). Second, we use $\{30, 50, 100, 150\}$ *random* value queries (RAND VQs) each consisting of five courses as the training set. To test the trained models, we use the complement of the training set as the test set and report the mean absolute error (MAE) and Kendall tau (KT) of each ML model.

Results. In Tables 17 and 18, we provide the generalization performance results for the *mean absolute error (MAE)* and the *Kendall tau* rank correlation with 25 courses in total and 6 popular courses.

Data Type	Ridge	nuSVR	XGBoost	NN	MVNN
GUI	37.45± 1.17	37.63± 1.13	38.38± 1.30	38.25± 1.09	37.38± 1.45
30 RAND VQs	22.35± 1.07	26.64± 0.85	25.82± 0.86	25.05± 0.89	20.65± 1.02
50 RAND VQs	15.08± 0.92	17.01± 0.69	19.96± 0.67	17.77± 0.77	13.46± 0.78
100 RAND VQs	11.37± 0.72	10.80± 0.60	15.74± 0.53	11.80± 0.56	7.80± 0.57
100 RAND VQs	10.56± 0.69	9.42± 0.56	13.68± 0.42	7.81± 0.50	4.79± 0.38

Table 17. MAE on the test set. 25 courses in total and 6 popular courses. Winners are marked in grey.

Data Type	Ridge	nuSVR	XGBoost	NN	MVNN
GUI	0.34± 0.02	0.34± 0.02	0.33± 0.02	0.31± 0.02	0.30± 0.03
30 RAND VQs	0.65± 0.02	0.61± 0.01	0.59± 0.01	0.60± 0.01	0.68± 0.02
50 RAND VQs	0.78± 0.01	0.76± 0.01	0.70± 0.01	0.73± 0.01	0.80± 0.01
100 RAND VQs	0.83± 0.01	0.84± 0.01	0.77± 0.01	0.83± 0.01	0.88± 0.01
150 RAND VQs	0.84± 0.01	0.86± 0.01	0.81± 0.01	0.88± 0.01	0.93± 0.01

Table 18. Kendall tau on the test set (larger is better). 25 courses in total and 6 popular courses. Winners marked in grey.

H GENERALIZATION PERFORMANCE RESULTS WITH 25 COURSES AND 9 POPULAR COURSES

Experiment Setup. We use the exact same experiment setup as described in Appendix G, except for the number of popular courses which we set to 9 when generating students' preferences with our proposed student preference generator defined in Section 5.1 and Appendix C.

Results. In Tables 19 and 20, we provide the generalization performance results for the *mean absolute error (MAE)* and the *Kendall tau* rank correlation with 25 courses in total and 9 popular courses.

Data Type	Ridge	nuSVR	XGBoost	NN	MVNN
GUI	34.01± 1.18	34.85± 1.13	34.99± 1.23	34.85± 1.12	34.52± 1.42
30 RAND VQs	19.36± 1.04	21.76± 0.81	23.01± 0.88	23.25± 0.88	18.64± 0.99
50 RAND VQs	12.88± 0.83	14.31± 0.59	15.79± 0.54	15.71± 0.65	11.93± 0.65
100 RAND VQs	9.87± 0.68	9.40± 0.50	15.75± 0.55	9.32± 0.56	5.84± 0.53
150 RAND VQs	9.18± 0.61	7.89± 0.44	13.72± 0.41	5.62± 0.48	3.50± 0.38

Table 19. MAE on the test set. 25 courses in total and 9 popular courses. Winners are marked in grey.

Data Type	Ridge	nuSVR	XGBoost	NN	MVNN
GUI	0.35± 0.02	0.34± 0.02	0.32± 0.02	0.30± 0.02	0.31± 0.02
30 RAND VQs	0.68± 0.02	0.65± 0.01	0.60± 0.02	0.59± 0.01	0.69± 0.02
50 RAND VQs	0.79± 0.01	0.78± 0.01	0.74± 0.01	0.75± 0.01	0.81± 0.01
100 RAND VQs	0.84± 0.01	0.85± 0.01	0.74± 0.01	0.86± 0.01	0.90± 0.01
150 RAND VQs	0.85± 0.01	0.87± 0.01	0.78± 0.01	0.9± 0.01	0.94± 0.01

Table 20. Kendall tau on the test set (larger is better). 25 courses in total and 9 popular courses. Winners are marked in grey.

I INTEGRATING COMPARISON QUERIES INTO MVNNS

In this section, we describe the details how we simultaneously train the MVNNs on GUI reports (regression data) and CQs (classification data).

Sculley [2010] proposed a method called *combined ranking and regression (CRR)* to train linear regression models using *both* regression and classification data at the same time (in our application the data generated from the students answers to the GUI is the regression data and the students' answers to comparison queries is the classification data). For this they use a trade-off parameter $\alpha \in [0, 1]$ to combine a regression loss and a ranking loss into one loss. More specifically, given α, a convex regression loss function $l_{reg}(\cdot)$, and a convex classification loss function $l_{class}(\cdot)$, we define the hybrid loss l_h as follows:

$$l_h(\cdot) = \alpha l_{reg}(\cdot) + (1 - \alpha) l_{class}(\cdot). \tag{41}$$

Using this hybrid loss, we fit the parameters of the MVNNs using Algorithm 2, which trains the MVNN using Adam on both the data from the GUI (regression data) and the data from the CQs (classification data).

The main idea of Algorithm 2 is to choose the data from GUI or the CQs to update the parameters of MVNN. In Algorithm 2, with probability α a data point from the regression data set is chosen (Line 4). Similarly, a data point from the classification data is chosen with probability $1 - \alpha$ (Line 8). If the data from the CQs is selected, we use the Sigmoid function $f(x) = \frac{1}{1+e^{-x}}$ to convert the real-valued outputs of MVNN$(\cdot)$ to the $[0, 1]$-interval, where the value of $\frac{1}{1+e^{-(\hat{y}_1 - \hat{y}_2)}}$ represents the predicted probability that bundle x_1 is more valuable than bundle x_2 for the student (Line 13).

ALGORITHM 2: Combined ranking and regression for MVNN

Input: $\{X_{reg}, y_{reg}\}$, $\{X_{class}, y_{class}\}$
Parameters: $\alpha \in [0, 1]$, epochs t, learning rate η
Output: Parameters of trained MVNN

```
θ_0 ← initialize parameters of the MVNN(·)
for i = 1 to t do
    pick z uniformly at random from [0, 1]
    if z < α then
        (x, y) ~^Unif {X_reg, y_reg}
        ŷ = MVNN(x)
        loss = l_reg(y, ŷ)
    else
        (x, y) ~^Unif {X_class, y_class}
        (x_1, x_2) = x
        ŷ_1 = MVNN(x_1)
        ŷ_2 = MVNN(x_2)
        ŷ = 1/(1+e^-(ŷ_1 - ŷ_2))
        loss = l_class(y, ŷ)
    end if
    θ_i = ADAM(θ_{i-1}, loss, η)
end for
return θ_t
```

ADAM is then used to update the parameters of MVNN. The algorithm terminates after t epochs and returns the selected parameters.

J INFERRING MISSING BASE VALUES - A WORKED EXAMPLE

In Example J.1, we show how MLCM can infer missing base values, a common error observed in the lab experiment by Budish and Kessler [2022].

Preference Model	Courses 1	2	3	4	Utility Maximizing Schedule a_1^*	Utility $u_1(a_1^*)$	Elicited CQ
u_1	85	70	40	0	{1,3}	125	
u_1^{GUI}	75	76	0	0	{2}	70	
$\mathcal{M}_1^{t=0}$	75	76	38	38	{2,3}	110	$\{2,3\} \succ \{2,4\}$
$\mathcal{M}_1^{t=1}$	75	76	41	33	{2,3}	110	$\{1,3\} \succ \{2,3\}$
$\mathcal{M}_1^{t=2}$	78	71	44	36	{1,3}	125	$\{2,3\} \succ \{1,4\}$
$\mathcal{M}_1^{t=3}$	78	72	45	32	{1,3}	125	$\{1,4\} \succ \{2,4\}$
$\mathcal{M}_1^{t=4}$	78	72	45	32	{1,3}	125	

Table 21. Worked example illustrating the ML-based preference elicitation algorithm. Each row represents the linear coefficients (corresponding to the base values) that uniquely define the corresponding function, the current utility maximizing schedule a_1^*, its corresponding utility $u_1(a_1^*)$ and the answer to the CQ.

Example J.1 (Inferring Missing Base Values). *In this example, we assume that the student forgot to report a base value for courses 3 and 4. This is treated in the original CM reporting language*

as inserting a base value equal to zero in the GUI, i.e., $u_1^{GUI}(\{3\}) = u_1^{GUI}(\{4\}) = 0$ *(as can be seen in the second row of Table 21). First, note that our cardinal dataset generation procedure (see Appendix A for details) implies that any schedule for which a student forgot to report a base value has the same initial predicted base value (as defined in Equation (5)), which is not necessarily zero but lower than the base value of any non-forgotten course. This results in* $\mathcal{M}_1^{t=0}(\{3\}) = \mathcal{M}_1^{t=0}(\{4\}) = 38$ *instead of zero.*

After this point, MLCM proceeds again in the same way as in Example 4.3 until the student stops answering CQs.

K WELFARE EXPERIMENTS DETAILS

For all welfare experiments we selected the MVNN hyperparameters shown in Table 22. .

Hidden Layers	Units per Hidden Layer	Learning Rate	Epochs	L2-Regularization	Batch Size
3	20	0.001	400	6e-5	8

Table 22. Selected hyperparameters for MVNNs in MLCM.

Computing Infrastructure. All welfare experiments were conducted on a compute cluster running Debian GNU/Linux 10 with Intel Xeon E5-2650 v4 2.20GHz processors with 24 cores and 128GB RAM and Intel E5 v2 2.80GHz processors with 20 cores and 128GB RAM and Python 3.7.10.

L WELFARE RESULTS FOR OTHER SETTINGS

In this section, we provide additional welfare results for the following three settings defined by a supply ratio (SR) and a number of popular courses (#Pop):

- SR = 1.50 and #Pop = 9 (see Table 23),
- SR = 1.50 and #Pop = 6 (see Table 24),
- SR = 1.25 and #Pop = 9 (see Table 25),

where we use the same experimental setup as described in the main paper in Section 7.2.

Results. From Table 23, Table 24, and Table 25, we see that the results (both average and minimum student utility) are better for SR = 1.5 and worse for 6 popular courses but qualitatively similar compared to the results presented in the main paper in Section 7.2.

Namely, for SR = 1.5 and 9 popular courses MLCM (10 ML-BASED CQs) increases average utility from 79.2% to 86.5% (a 8.8% increase) and minimum utility from 41.6% to 50.4% (a 21.2% increase). At the most extreme setting of SR = 1.25 and 6 popular courses, MLCM (10 ML-BASED CQs) increases average utility compared to CM from 83.4% to 86.7% (a 4% increase) and minimum utility from 50.0% to 55.0% (a 10.0% increase). For all settings, as the number of CQs increases, the performance of MLCM improves further.

M STATISTICAL SIGNIFICANCE TESTS

To test whether our results from Section 7 are statistically significant, we perform a *multivariate analysis of variance (MANOVA)* test, using as the independent variable the mechanism and as dependent variables (a) average student utility, and (b) the minimum student utility. We use as significance level for this test a value of 0.05. The MANOVA test determined that there is a statistically significant treatment effect between the different mechanisms. Given this, we then performed a Post-Hoc tukey test for all pairs of mechanisms. For those tests, we used one dependent

	Avgerage Student Utility			Minimum Student Utility			Overs.	Time
Mechanism	**Stage 1**	**Stage 2**	**Stage 3**	**Stage 1**	**Stage 2**	**Stage 3**	**Stage 1**	**in h**
CM* (Full Preferences)	100.0 ± 0.0	98.5 ± 0.3	99.7 ± 0.2	71.5 ± 1.0	71.1 ± 1.0	71.4 ± 1.0	3.3 ± 0.6	3.3
CM (No Mistakes)	98.4 ± 0.3	97.3 ± 0.3	98.3 ± 0.3	71.2 ± 1.0	70.7 ± 1.0	71.1 ± 1.0	2.5 ± 0.5	2.9
RSD	-	-	78.2 ± 0.5	-	-	36.2 ± 1.2	-	0.0
CM	79.3 ± 0.5	78.7 ± 0.4	79.2 ± 0.5	41.7 ± 1.1	41.7 ± 1.1	41.6 ± 1.1	1.3 ± 0.3	1.4
MLCM (1 ML-based CQ)	79.2 ± 0.5	78.7 ± 0.5	79.1 ± 0.5	41.0 ± 1.2	40.8 ± 1.2	40.9 ± 1.2	1.2 ± 0.3	13.6
MLCM (5 ML-based CQs)	83.8 ± 0.4	83.2 ± 0.4	83.7 ± 0.4	47.3 ± 1.2	46.8 ± 1.2	47.0 ± 1.1	1.1 ± 0.2	14.1
MLCM (10 ML-based CQs)	86.6 ± 0.4	86.0 ± 0.4	86.5 ± 0.4	50.3 ± 1.2	50.0 ± 1.2	50.4 ± 1.2	1.1 ± 0.3	12.4
MLCM (15 ML-based CQs)	88.4 ± 0.4	87.8 ± 0.4	88.3 ± 0.4	51.8 ± 1.3	51.4 ± 1.3	51.6 ± 1.3	1.2 ± 0.3	13.2
MLCM (20 ML-based CQs)	89.3 ± 0.4	88.6 ± 0.4	89.2 ± 0.4	54.2 ± 1.2	53.7 ± 1.2	53.9 ± 1.2	1.3 ± 0.3	16.7
MLCM (20 Random CQs)	78.7 ± 0.5	78.2 ± 0.5	78.6 ± 0.5	41.0 ± 1.2	40.8 ± 1.2	40.9 ± 1.2	1.4 ± 0.3	16.7

Table 23. Comparison of RSD, CM and MLCM (also using random CQs) in Stages 1–3 for a supply ratio of 1.5, 9 popular courses, and default parameterization for reporting mistakes. We normalize all results by the average utility of CM* after Stage 1. Shown are averages in % over 100 runs and 95% CIs. Additionally, we present the oversubscription (in number of seats) after Stage 1 (Overs.) and total runtime (in hours) per run.

	Avgerage Student Utility			Minimum Student Utility			Overs.	Time
Mechanism	**Stage 1**	**Stage 2**	**Stage 3**	**Stage 1**	**Stage 2**	**Stage 3**	**Stage 1**	**in h**
CM* (Full Preferences)	100.0 ± 0.0	95.7 ± 0.7	98.4 ± 0.5	77.5 ± 0.7	74.0 ± 0.9	75.8 ± 0.9	8.7 ± 1.6	3.8
CM (No Mistakes)	98.7 ± 0.4	93.7 ± 0.7	96.6 ± 0.6	76.5 ± 0.8	72.7 ± 0.9	73.9 ± 0.9	10.8 ± 1.5	3.2
RSD	-	-	81.0 ± 0.7	-	-	34.8 ± 1.3	-	0.0
CM	83.8 ± 0.6	81.3 ± 0.7	83.4 ± 0.6	49.4 ± 1.4	48.1 ± 1.4	48.8 ± 1.4	5.0 ± 0.9	1.4
MLCM (1 ML-based CQ)	83.3 ± 0.6	81.1 ± 0.7	83.0 ± 0.6	50.2 ± 1.3	48.3 ± 1.3	49.8 ± 1.3	4.4 ± 0.8	21.4
MLCM (5 ML-based CQs)	87.1 ± 0.6	83.7 ± 0.8	86.1 ± 0.6	55.0 ± 1.3	51.4 ± 1.4	52.7 ± 1.4	5.7 ± 1.0	20.8
MLCM (10 ML-based CQs)	89.2 ± 0.5	85.6 ± 0.7	88.2 ± 0.5	57.7 ± 1.3	54.5 ± 1.2	55.9 ± 1.3	5.5 ± 0.9	21.0
MLCM (15 ML-based CQs)	90.3 ± 0.6	86.2 ± 0.8	88.9 ± 0.6	59.7 ± 1.2	55.1 ± 1.4	56.9 ± 1.3	5.8 ± 1.0	22.7
MLCM (20 ML-based CQs)	90.8 ± 0.5	87.2 ± 0.7	89.9 ± 0.6	59.2 ± 1.3	56.4 ± 1.2	57.6 ± 1.2	5.6 ± 0.9	23.3
MLCM (20 Random CQs)	83.0 ± 0.6	80.6 ± 0.6	82.5 ± 0.6	50.0 ± 1.3	48.9 ± 1.4	49.5 ± 1.4	4.4 ± 0.7	22.1

Table 24. Comparison of RSD, CM and MLCM (also using random CQs) in Stages 1–3 for a supply ratio of 1.5, 6 popular courses, and default parameterization for reporting mistakes. We normalize all results by the average utility of CM* after Stage 1. Shown are averages in % over 100 runs and 95% CIs. Additionally, we present the oversubscription (in number of seats) after Stage 1 (Overs.) and total runtime (in hours) per run.

	Avgerage Student Utility			Minimum Student Utility			Overs.	Time
Mechanism	**Stage 1**	**Stage 2**	**Stage 3**	**Stage 1**	**Stage 2**	**Stage 3**	**Stage 1**	**in h**
CM* (Full Preferences)	100.0 ± 0.0	94.4 ± 1.0	97.7 ± 0.9	78.0 ± 0.7	74.0 ± 1.1	75.5 ± 1.0	13.7 ± 2.1	5.0
CM (No Mistakes)	98.4 ± 0.5	92.7 ± 1.3	96.0 ± 1.1	77.1 ± 0.7	73.2 ± 1.3	75.1 ± 1.2	11.6 ± 2.0	4.3
RSD	-	-	80.5 ± 0.8	-	-	31.3 ± 1.4	-	0.0
CM	83.7 ± 0.6	81.2 ± 0.7	83.4 ± 0.7	49.8 ± 1.3	49.1 ± 1.2	50.0 ± 1.3	5.4 ± 0.9	2.0
MLCM (1 ML-based CQ)	83.1 ± 0.6	80.6 ± 0.8	82.4 ± 0.7	50.3 ± 1.4	48.2 ± 1.3	49.1 ± 1.4	5.2 ± 0.8	27.7
MLCM (5 ML-based CQs)	86.4 ± 0.6	82.4 ± 1.2	84.6 ± 1.1	54.6 ± 1.3	51.3 ± 1.5	52.1 ± 1.5	5.6 ± 0.8	30.1
MLCM (10 ML-based CQs)	88.4 ± 0.6	84.2 ± 1.4	86.7 ± 1.3	56.9 ± 1.4	54.0 ± 1.4	55.0 ± 1.5	6.1 ± 1.1	29.8
MLCM (15 ML-based CQs)	89.6 ± 0.6	85.6 ± 1.2	88.2 ± 1.1	59.1 ± 1.3	55.9 ± 1.6	58.2 ± 1.6	6.0 ± 1.0	30.9
MLCM (20 ML-based CQs)	90.9 ± 0.6	86.5 ± 1.2	88.9 ± 1.1	60.6 ± 1.3	56.4 ± 1.7	58.9 ± 1.6	6.6 ± 1.1	31.5
MLCM (20 Random CQs)	83.0 ± 0.6	79.8 ± 0.9	81.9 ± 0.8	49.8 ± 1.3	47.7 ± 1.3	48.9 ± 1.4	5.7 ± 1.0	28.7

Table 25. Comparison of RSD, CM and MLCM (also using random CQs) in Stages 1–3 for a supply ratio of 1.25, 6 popular courses, and default parameterization for reporting mistakes. We normalize all results by the average utility of CM* after Stage 1. Shown are averages in % over 100 runs and 95% CIs. Additionally, we present the oversubscription (in number of seats) after Stage 1 (Overs.) and total runtime (in hours) per run.

variable (i.e., either average or minimum student utility), with a significance level of 0.025 where

the null hypothesis is that there is no treatment effect between Group1 and Group2, i.e., $\mathcal{H}_0 : \mu_{Group1} = \mu_{Group2}$. The results for each setting are provided in Tables 26 to 33.

Group1	Group2	Meandiff	p-adj	lower	upper	reject
CM	MLCM (10 ML-based CQs)	0.0563	0.0010	0.0417	0.0708	True
CM	MLCM (20 ML-based CQs)	0.0899	0.0010	0.0754	0.1044	True
CM	MLCM (20 Random CQs)	-0.0075	0.5326	-0.0220	0.0070	False
CM	RSD	-0.0216	0.0010	-0.0361	-0.0071	True
MLCM (10 ML-based CQs)	MLCM (20 ML-based CQs)	0.0337	0.0010	0.0191	0.0482	True
MLCM (10 ML-based CQs)	MLCM (20 Random CQs)	-0.0638	0.0010	-0.0783	-0.0492	True
MLCM (10 ML-based CQs)	RSD	-0.0778	0.0010	-0.0924	-0.0633	True
MLCM (20 ML-based CQs)	MLCM (20 Random CQs)	-0.0974	0.0010	-0.1119	-0.0829	True
MLCM (20 ML-based CQs)	RSD	-0.1115	0.0010	-0.1260	-0.0970	True
MLCM (20 Random CQs)	RSD	-0.0141	0.0324	-0.0286	0.0004	False

Table 26. Post-hoc tukey test for average student utility. Supply ratio 1.25, 9 popular courses. Significance level for Reject column was set to 0.025.

Group1	Group2	Meandiff	p-adj	lower	upper	reject
CM	MLCM (10 ML-based CQs)	0.0627	0.0010	0.0353	0.0900	True
CM	MLCM (20 ML-based CQs)	0.0937	0.0010	0.0664	0.1210	True
CM	MLCM (20 Random CQs)	-0.0152	0.4661	-0.0425	0.0122	False
CM	RSD	-0.1224	0.0010	-0.1498	-0.0951	True
MLCM (10 ML-based CQs)	MLCM (20 ML-based CQs)	0.0310	0.0069	0.0037	0.0584	True
MLCM (10 ML-based CQs)	MLCM (20 Random CQs)	-0.0778	0.0010	-0.1052	-0.0505	True
MLCM (10 ML-based CQs)	RSD	-0.1851	0.0010	-0.2125	-0.1578	True
MLCM (20 ML-based CQs)	MLCM (20 Random CQs)	-0.1089	0.0010	-0.1362	-0.0815	True
MLCM (20 ML-based CQs)	RSD	-0.2161	0.0010	-0.2435	-0.1888	True
MLCM (20 Random CQs)	RSD	-0.1073	0.0010	-0.1346	-0.0800	True

Table 27. Post-hoc tukey test for minimum student utility. Supply ratio 1.25, 9 popular courses. Significance level for Reject column was set to 0.025.

Group1	Group2	Meandiff	p-adj	lower	upper	reject
CM	MLCM (10 ML-based CQs)	0.0724	0.0010	0.0628	0.0820	True
CM	MLCM (20 ML-based CQs)	0.0993	0.0010	0.0897	0.1089	True
CM	MLCM (20 Random CQs)	-0.0066	0.2454	-0.0162	0.0030	False
CM	RSD	-0.0108	0.0072	-0.0204	-0.0013	True
MLCM (10 ML-based CQs)	MLCM (20 ML-based CQs)	0.0269	0.0010	0.0173	0.0365	True
MLCM (10 ML-based CQs)	MLCM (20 Random CQs)	-0.0790	0.0010	-0.0886	-0.0694	True
MLCM (10 ML-based CQs)	RSD	-0.0832	0.0010	-0.0928	-0.0737	True
MLCM (20 ML-based CQs)	MLCM (20 Random CQs)	-0.1059	0.0010	-0.1155	-0.0963	True
MLCM (20 ML-based CQs)	RSD	-0.1102	0.0010	-0.1197	-0.1006	True
MLCM (20 Random CQs)	RSD	-0.0043	0.6544	-0.0138	0.0053	False

Table 28. Post-hoc tukey test for average student utility. Supply ratio 1.5, 9 popular courses. Significance level for Reject column was set to 0.025.

N MISTAKE ABLATION EXPERIMENT FOR SUPPLY RATIO 1.5

In this section, we present three further reporting mistake ablation experiments for the following choices of supply ratios and number of popular courses, respectively:

- A supply ratio of 1.25 and 6 popular courses (see Figure 6).
- A supply ratio of 1.5 and 9 popular courses (see Figure 7).
- A supply ratio of 1.5 and 6 popular courses (see Figure 8).

group1	group2	meandiff	p-adj	lower	upper	reject
CM	MLCM (10 ML-based CQs)	0.0882	0.001	0.0627	0.1137	True
CM	MLCM (20 ML-based CQs)	0.1229	0.001	0.0974	0.1484	True
CM	MLCM (20 Random CQs)	-0.0066	0.900	-0.0321	0.0189	False
CM	RSD	-0.0537	0.001	-0.0792	-0.0282	True
MLCM (10 ML-based CQs)	MLCM (20 ML-based CQs)	0.0347	0.001	0.0092	0.0602	True
MLCM (10 ML-based CQs)	MLCM (20 Random CQs)	-0.0949	0.001	-0.1204	-0.0694	True
MLCM (10 ML-based CQs)	RSD	-0.1419	0.001	-0.1674	-0.1164	True
MLCM (20 ML-based CQs)	MLCM (20 Random CQs)	-0.1295	0.001	-0.1550	-0.1040	True
MLCM (20 ML-based CQs)	RSD	-0.1766	0.001	-0.2021	-0.1511	True
MLCM (20 Random CQs)	RSD	-0.0470	0.001	-0.0725	-0.0215	True

Table 29. Post-hoc tukey test for minimum student utility. Supply ratio 1.5, 9 popular courses. Significance level for Reject column was set to 0.025.

group1	group2	meandiff	p-adj	lower	upper	reject
CM	MLCM (10 ML-based CQs)	0.0326	0.0010	0.0125	0.0527	True
CM	MLCM (20 ML-based CQs)	0.0551	0.0010	0.0350	0.0752	True
CM	MLCM (20 Random CQs)	-0.0152	0.1631	-0.0352	0.0049	False
CM	RSD	-0.0288	0.0010	-0.0489	-0.0087	True
MLCM (10 ML-based CQs)	MLCM (20 ML-based CQs)	0.0225	0.0079	0.0024	0.0426	True
MLCM (10 ML-based CQs)	MLCM (20 Random CQs)	-0.0478	0.0010	-0.0678	-0.0277	True
MLCM (10 ML-based CQs)	RSD	-0.0614	0.0010	-0.0815	-0.0414	True
MLCM (20 ML-based CQs)	MLCM (20 Random CQs)	-0.0703	0.0010	-0.0903	-0.0502	True
MLCM (20 ML-based CQs)	RSD	-0.0839	0.0010	-0.1040	-0.0639	True
MLCM (20 Random CQs)	RSD	-0.0137	0.2532	-0.0337	0.0064	False

Table 30. Post-hoc tukey test for average student utility. Supply ratio 1.25, 6 popular courses. Significance level for Reject column was set to 0.025.

group1	group2	meandiff	p-adj	lower	upper	reject
CM	MLCM (10 ML-based CQs)	0.0497	0.0010	0.0191	0.0803	True
CM	MLCM (20 ML-based CQs)	0.0886	0.0010	0.0580	0.1192	True
CM	MLCM (20 Random CQs)	-0.0113	0.7813	-0.0419	0.0193	False
CM	RSD	-0.1873	0.0010	-0.2180	-0.1567	True
MLCM (10 ML-based CQs)	MLCM (20 ML-based CQs)	0.0389	0.0016	0.0083	0.0695	True
MLCM (10 ML-based CQs)	MLCM (20 Random CQs)	-0.0610	0.0010	-0.0916	-0.0304	True
MLCM (10 ML-based CQs)	RSD	-0.2370	0.0010	-0.2676	-0.2064	True
MLCM (20 ML-based CQs)	MLCM (20 Random CQs)	-0.0999	0.0010	-0.1305	-0.0693	True
MLCM (20 ML-based CQs)	RSD	-0.2760	0.0010	-0.3066	-0.2453	True
MLCM (20 Random CQs)	RSD	-0.1761	0.0010	-0.2067	-0.1454	True

Table 31. Post-hoc tukey test for minimum student utility. Supply ratio 1.25, 6 popular courses. Significance level for Reject column was set to 0.025.

group1	group2	meandiff	p-adj	lower	upper	reject
CM	MLCM (10 ML-based CQs)	0.0479	0.0010	0.0351	0.0607	True
CM	MLCM (20 ML-based CQs)	0.0651	0.0010	0.0523	0.0780	True
CM	MLCM (20 Random CQs)	-0.0088	0.2500	-0.0216	0.0041	False
CM	RSD	-0.0241	0.0010	-0.0369	-0.0113	True
MLCM (10 ML-based CQs)	MLCM (20 ML-based CQs)	0.0172	0.0010	0.0044	0.0301	True
MLCM (10 ML-based CQs)	MLCM (20 Random CQs)	-0.0567	0.0010	-0.0695	-0.0438	True
MLCM (10 ML-based CQs)	RSD	-0.0720	0.0010	-0.0848	-0.0591	True
MLCM (20 ML-based CQs)	MLCM (20 Random CQs)	-0.0739	0.0010	-0.0867	-0.0611	True
MLCM (20 ML-based CQs)	RSD	-0.0892	0.0010	-0.1020	-0.0764	True
MLCM (20 Random CQs)	RSD	-0.0153	0.0037	-0.0281	-0.0025	True

Table 32. Post-hoc tukey test for average student utility. Supply ratio 1.5, 6 popular courses. Significance level for Reject column was set to 0.025.

GROUP1	GROUP2	MEANDIFF	P-ADJ	LOWER	UPPER	REJECT
CM	MLCM (10 ML-BASED CQs)	0.0715	0.001	0.0434	0.0995	TRUE
CM	MLCM (20 ML-BASED CQs)	0.0882	0.001	0.0602	0.1163	TRUE
CM	MLCM (20 RANDOM CQs)	0.0076	0.900	-0.0205	0.0356	FALSE
CM	RSD	-0.1396	0.001	-0.1676	-0.1115	TRUE
MLCM (10 ML-BASED CQs)	MLCM (20 ML-BASED CQs)	0.0167	0.388	-0.0113	0.0448	FALSE
MLCM (10 ML-BASED CQs)	MLCM (20 RANDOM CQs)	-0.0639	0.001	-0.0920	-0.0359	TRUE
MLCM (10 ML-BASED CQs)	RSD	-0.2110	0.001	-0.2391	-0.1830	TRUE
MLCM (20 ML-BASED CQs)	MLCM (20 RANDOM CQs)	-0.0807	0.001	-0.1087	-0.0526	TRUE
MLCM (20 ML-BASED CQs)	RSD	-0.2278	0.001	-0.2558	-0.1997	TRUE
MLCM (20 RANDOM CQs)	RSD	-0.1471	0.001	-0.1752	-0.1191	TRUE

Table 33. Post-hoc tukey test for minimum student utility. Supply ratio 1.5, 6 popular courses. Significance level for REJECT column was set to 0.025.

Results. Similarly to the results presented in the main paper in Figure 3 (supply ratio of 1.25 and 9 popular courses), we see that, as γ increases, the performance of both, CM and MLCM, monotonically decreases. This should not come as a surprise, as for values of γ larger than 1 the students make more mistakes when reporting their preferences to the GUI. MLCM significantly outperforms CM for all $\gamma \in [0.75, 1.5]$. Moreover, as γ increases, the relative performance gap of the two mechanisms gets larger. These results further highlight the robustness of our design to changes in the mistake profile of the students.

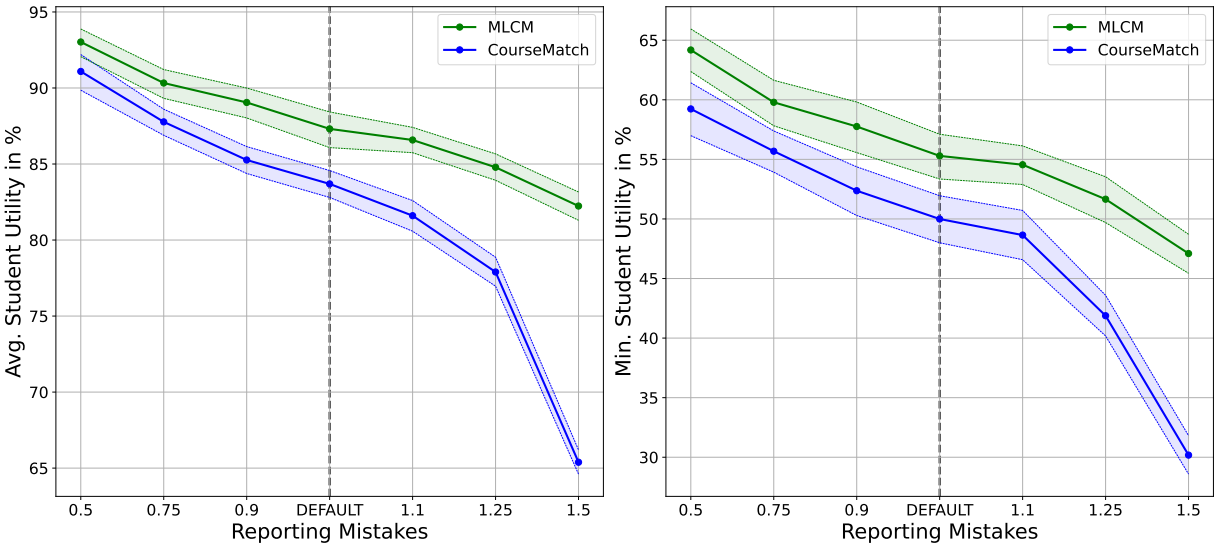


Fig. 6. Reporting mistakes ablation experiment for a supply ratio of 1.25 and 6 popular courses. Shown are average results in % for the final allocation over 50 runs including 95% CI.

O SHOULD INDIVIDUAL STUDENTS OPT INTO THE ML FEATURE?

In this section, we first provide additional results for a supply ratio of 1.5 for the experiment measuring the expected utility gain of a student if she were the only one to opt into MLCM, as described in Section 7.4. We present those results in Table 34. Furthermore, we perform a similar test, measuring the expected gain of a student opting into MLCM, if *every* other student also opted in. We present those results in Table 35.

No Other Student to Opt into the ML Feature. Similarly to the results presented in the main paper (see Table 4 for a supply ratio of 1.25), Table 34 shows that for a supply ratio equal to 1.5 the *expected*

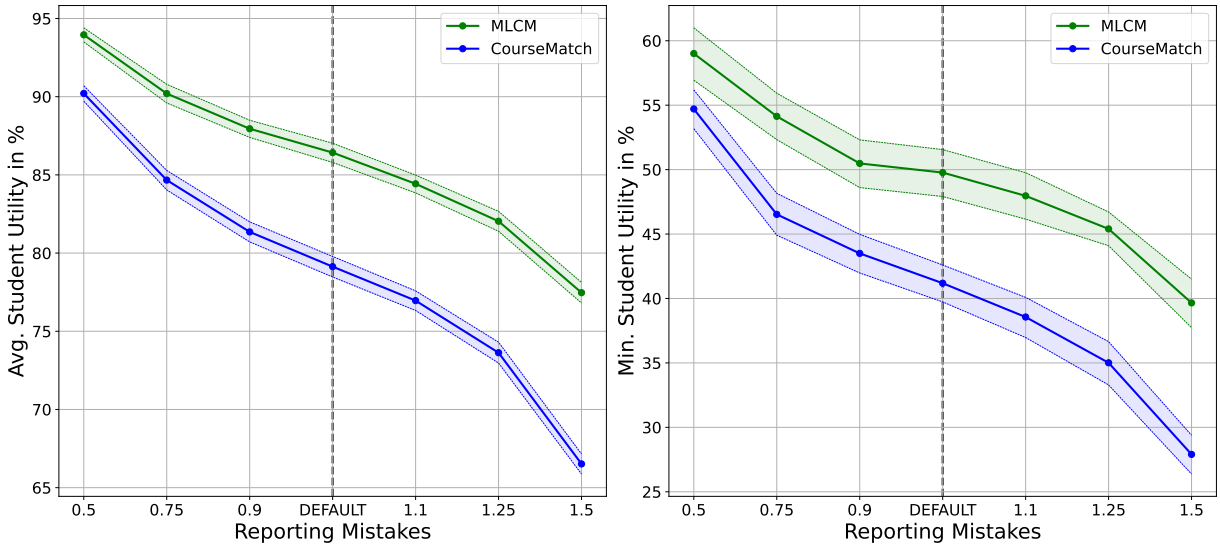


Fig. 7. Reporting mistakes ablation experiment for a supply ratio of 1.5 and 9 popular courses. Shown are average results in % for the final allocation over 50 runs including 95% CI.

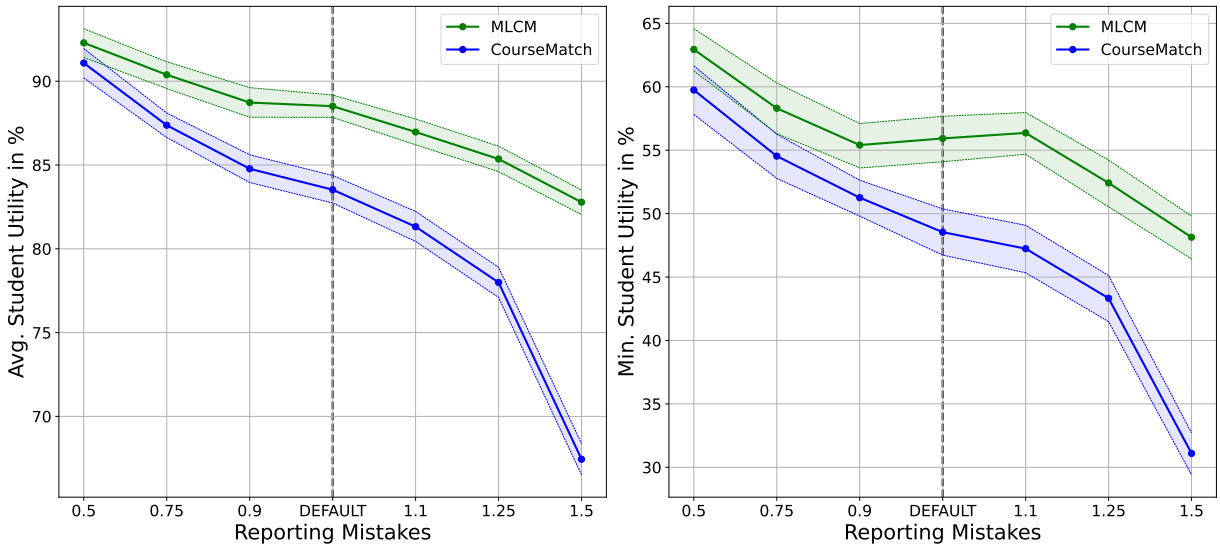


Fig. 8. Reporting mistakes ablation experiment for a supply ratio of 1.5 and 6 popular courses. Shown are average results in % for the final allocation over 50 runs including 95% CI.

relative gain from opting in is at least 7.9% (across all settings). Furthermore, the student prefers the "MLCM schedule" in at least 68.45% of the cases, while she prefers the "CM schedule" in at most 7.8% of the cases. As the number of comparison queries (CQs) the student answers increases, the benefit from opting into MLCM's ML feature as the first student becomes even larger. Finally, the improvement is larger in for a setting with more popular courses.

All Other Students to Opt into the ML Feature. Now suppose that all students have decided to opt into MLCM's ML feature, except for one. How much would that last student benefit if she decided to also "opt in"? As before, we answer this question by running both MLCM and CM twice – once

Setting			Preferred Mechanism			Gain from Opting Into MLCM		
SR	#PoP	#CQs	MLCM	CM	Indiff.	Expected	if pref MLCM	if pref CM
1.50	9	10	73.85%	4.70%	21.45%	10.9%	15.5%	-10.9%
1.50	9	15	80.45%	2.80%	16.75%	13.6%	17.2%	-6.7%
1.50	9	20	84.75%	2.90%	12.35%	15.2%	18.3%	-9.4%
1.50	6	10	68.45%	7.80%	23.75%	7.9%	12.5%	-8.5%
1.50	6	15	74.65%	7.00%	18.35%	9.4%	13.4%	-8.9%
1.50	6	20	78.60%	6.80%	14.60%	10.3%	13.8%	-7.6%

Table 34. Expected gain of opting into MLCM's ML feature when no other student opts in. Shown are average results across 2000 students per setting (SR,#PoP,#CQs). CIs for all metrics are ≈ 0.

where all but one student have opted in, and once where *all* students have opted in.[18] We use 20 instances and 100 students per instance. We report averages over those 2000 students. Table 35 shows those results for both supply ratios, 1.25 and 1.5. We see that the *expected relative gain* from opting in is at least 7.6% (across all settings). Furthermore, the student prefers the "MLCM schedule" in at least 64% of the cases, while she prefers the "CM schedule" in at most 11.15% of the cases. As the student answers more CQs, the benefit from opting into MLCM's ML feature becomes even larger. Finally, the improvement is larger for more popular courses and for a larger SR.

Setting			Preferred Mechanism			Gain from Opting Into MLCM		
SR	#PoP	#CQs	MLCM	CM	Indiff.	Expected	if pref MLCM	if pref CM
1.25	9	10	68.20%	9.00%	22.80%	10.1%	16.0%	-9.4%
1.25	9	15	74.00%	7.45%	18.55%	12.1%	17.2%	-8.3%
1.25	9	20	79.50%	7.10%	13.40%	14.4%	18.8%	-7.0%
1.25	6	10	64.00%	11.15%	24.85%	7.6%	13.5%	-9.6%
1.25	6	15	69.95%	10.70%	19.35%	8.7%	13.8%	-8.4%
1.25	6	20	73.05%	11.15%	15.80%	9.9%	14.9%	-8.0%
1.50	9	10	72.50%	5.45%	22.05%	10.7%	15.5%	-9.8%
1.50	9	15	79.25%	3.20%	17.55%	13.2%	17.1%	-8.1%
1.50	9	20	82.35%	5.05%	12.60%	14.6%	18.3%	-9.1%
1.50	6	10	66.60%	9.15%	24.25%	7.7%	12.7%	-7.9%
1.50	6	15	71.70%	9.55%	18.75%	8.8%	13.6%	-9.0%
1.50	6	20	74.90%	9.85%	15.25%	9.6%	13.9%	-7.8%

Table 35. Expected gain of opting into MLCM's ML feature when all other students also opt in. Shown are average results across 2000 students per setting (SR,#PoP,#CQs). CIs for all metrics are ≈ 0.

P ADDITIVE PREFERENCES

In this section, we repeat all of our experiments for the special case of students having *additive* true preferences, i.e., we use our student preference generator (see Section 5.1 and Appendix C) to generate additive/linear true students' utility functions. Again, we calibrated the mistake profile of the students so that both their accuracy and the reported utility difference in case of disagreements match those determined in the lab experiment of Budish and Kessler [2022] (see Table 36).

[18] As before, we report Stage 1 results but now use the fixed price vector that would result if all students chose to opt in.

Setting		#Courses with value			#Adjustments				Accuracy	If disagreement
#Pop	γ	> 0	$(0, 50)$	$[50, 100]$	Mean	Median	Min	Max	CQs	utility difference in %
9	0.5	17.20 ± 0.02	9.96 ± 0.10	7.24 ± 0.09	0.00 ± 0.00	0.0	0	0	0.95 ± 0.01	-1.47 ± 0.31
9	0.75	13.39 ± 0.03	5.66 ± 0.11	7.73 ± 0.11	0.00 ± 0.00	0.0	0	0	0.91 ± 0.01	-6.44 ± 0.73
9	0.9	11.03 ± 0.01	3.00 ± 0.11	8.03 ± 0.11	0.00 ± 0.00	0.0	0	0	0.87 ± 0.01	-11.11 ± 0.89
9	1	9.49 ± 0.03	1.53 ± 0.09	7.96 ± 0.09	0.00 ± 0.00	0.0	0	0	0.84 ± 0.01	-14.03 ± 1.00
9	1.1	7.92 ± 0.02	0.50 ± 0.06	7.42 ± 0.06	0.00 ± 0.00	0.0	0	0	0.78 ± 0.01	-17.70 ± 1.07
9	1.25	5.62 ± 0.03	0.04 ± 0.01	5.58 ± 0.03	0.00 ± 0.00	0.0	0	0	0.69 ± 0.02	-22.18 ± 1.07
9	1.5	1.75 ± 0.03	0.00 ± 0.00	1.75 ± 0.03	0.00 ± 0.00	0.0	0	0	0.59 ± 0.02	-17.01 ± 1.36
Budish and Kessler [2022]		12.45	6.17	6.27	1.08	0	0	10	0.84	-13.35 ± 0.41

Table 36. Mistake profile calibration experiment for *additive* preferences for several settings defined by the number of popular courses (#Pop) and the common mistake constant γ compared to the experimental findings in [Budish and Kessler, 2022]. Our default settings (i.e., $\gamma = 1$) is marked in grey. 1000 students in total. We show the number of courses with reported value in 3 distinct intervals, the mean, median, minimum and maximum number of adjustments in the students' reports, the accuracy of their reports as determined by asking CQs and the median scaled utility difference between the two schedules in a CQ in case of disagreement between the CQ answer and the reported preferences. Shown are average results and a 95% CI.

For these new preferences we present

- the main welfare results for supply ratios 1.25 and 1.5 (Table 37 and Table 38),
- the reporting mistakes ablation study for supply ratios 1.25 and 1.5 (Figure 9 and Figure 10),
- the first- and last-student to opt into MLCM experimental study (Table 39 and Table 40).

Overall our results in the following three sections show that MLCM achieves qualitatively the same results when the true students' utilities are restricted to be additive/linear. These results further highlight the robustness of our design, and its adaptability to different environments.

P.1 Welfare Results for Additive Preferences

In this subsection, we present the main welfare results when students' true utility functions are restricted to be *additive/linear*. Please see Section 7.2 for details on the experiment setup.

In Table 37, we present results for SR = 1.25 (which is very close to Wharton's SR; see [Budish and Kessler, 2022]) and 9 popular courses. The results are qualitatively very similar for a SR of 1.5. We normalize all results by the average utility of CM* (Full Preferences) after Stage 1. Note that in this setting with additive preferences, the mechanisms CM* (Full Preferences) and CM (No Mistakes) completely coincide. The metrics of interest are the average and minimum student utility after Stage 3 (i.e., for the final allocation). We see that MLCM (10 ML-BASED CQs) significantly outperforms CM, both in average and minimum student utility. In particular, MLCM (10 ML-BASED CQs) increases average utility from 84.7% to 90.6% (a 7% increase) and minimum utility from 53.3% to 65.6% (a 23% increase). As the number of CQs increases, the performance of MLCM improves further.

In these tables, we show one additional version of MLCM, MLCM-LR. It is the same MLCM mechanism but now the machine learning model of each student is an MVNN with zero hidden layers, thus it can only capture additive preferences (as is the case for this setting). The performance achieved with these networks is on par with our original MLCM design, yet at the same time, the computational cost of calculating a final allocation is the same as for CM, as indicated by the last column of Table 37. Thus, for a setting where the school in question only wishes to capture such preferences, the welfare improvement of MLCM compared to CM comes at no additional computational cost.

Mechanism	Average Student Utility			Minimum Student Utility			Oversubs	Time
	Stage 1	Stage 2	Stage 3	Stage 1	Stage 2	Stage 3	Stage 1	(in h)
CM* (Full Preferences)	100.0 ± 0.0	89.6 ± 2.5	93.0 ± 2.2	84.5 ± 0.6	74.7 ± 2.3	77.3 ± 2.2	10.9 ± 1.6	2.6
CM (No Mistakes)	100.0 ± 0.0	89.6 ± 2.5	93.0 ± 2.2	84.5 ± 0.6	74.7 ± 2.3	77.3 ± 2.2	10.9 ± 1.6	2.6
CM	85.8 ± 0.4	82.3 ± 0.6	84.7 ± 0.4	54.9 ± 1.0	51.8 ± 1.1	53.3 ± 1.0	5.2 ± 0.9	1.2
MLCM (1 ML-based CQs)	88.7 ± 0.2	85.1 ± 0.8	87.5 ± 0.6	61.3 ± 1.0	58.5 ± 1.2	60.4 ± 1.2	5.6 ± 0.8	26.1
MLCM (5 ML-based CQs)	91.3 ± 0.2	86.9 ± 1.0	89.2 ± 0.9	65.0 ± 1.1	60.5 ± 1.4	62.3 ± 1.3	5.9 ± 1.0	25.8
MLCM (10 ML-based CQs)	93.2 ± 0.3	88.1 ± 1.2	90.6 ± 1.0	68.3 ± 1.1	62.9 ± 1.5	65.6 ± 1.2	7.2 ± 1.3	25.9
MLCM (15 ML-based CQs)	94.1 ± 0.2	88.9 ± 1.4	91.5 ± 1.3	69.8 ± 1.0	65.0 ± 1.6	66.7 ± 1.5	6.0 ± 1.0	27.3
MLCM (20 ML-based CQs)	94.9 ± 0.2	89.3 ± 1.3	92.1 ± 1.1	71.1 ± 1.0	65.4 ± 1.6	67.9 ± 1.4	6.7 ± 1.1	28.2
MLCM-LR (1 ML-based CQ)	88.6 ± 0.2	84.7 ± 0.8	87.2 ± 0.6	60.5 ± 1.0	57.5 ± 1.1	58.7 ± 1.1	5.8 ± 0.9	1.1
MLCM-LR (5 ML-based CQs)	90.7 ± 0.2	86.6 ± 0.8	89.1 ± 0.6	63.4 ± 1.0	60.3 ± 1.4	61.8 ± 1.2	6.6 ± 1.1	1.0
MLCM-LR (10 ML-based CQs)	91.7 ± 0.2	87.0 ± 1.3	89.5 ± 1.1	65.5 ± 1.1	61.2 ± 1.8	63.2 ± 1.6	5.6 ± 0.8	1.0
MLCM-LR (10 ML-based CQs)	93.0 ± 0.2	88.2 ± 1.4	90.5 ± 1.3	67.3 ± 1.1	63.2 ± 1.7	65.4 ± 1.5	6.2 ± 0.9	1.0
MLCM-LR (20 ML-based CQs)	93.8 ± 0.2	88.7 ± 1.2	91.3 ± 1.0	69.1 ± 1.0	64.6 ± 1.5	66.6 ± 1.4	5.6 ± 0.7	1.0

Table 37. Comparison of CM and MLCM in Stages 1–3 for *additive* preferences, a supply ratio of 1.25, 9 popular courses, and default parameterization for reporting mistakes. We normalize all results by the average utility of CM* after Stage 1. Shown are averages in % over 100 runs and 95% CIs. Additionally, we present the oversubscription (in number of course seats) after Stage 1 (Overs.) and total runtime (in hours) per run.

Mechanism	Average Student Utility			Minimum Student Utility			Oversubs	Time
	Stage 1	Stage 2	Stage 3	Stage 1	Stage 2	Stage 3	Stage 1	(in h)
CM* (Full Preferences)	100.0 ± 0.0	96.4 ± 0.5	98.6 ± 0.4	84.3 ± 0.6	81.0 ± 0.8	82.7 ± 0.7	12.8 ± 1.8	1.8
CM (No Mistakes)	100.0 ± 0.0	96.4 ± 0.5	98.6 ± 0.4	84.3 ± 0.6	81.0 ± 0.8	82.7 ± 0.7	12.8 ± 1.8	1.8
CM	88.4 ± 0.4	84.9 ± 0.6	87.4 ± 0.4	57.5 ± 1.1	54.4 ± 1.1	56.7 ± 1.1	4.1 ± 0.7	1.1
MLCM (1 ML-based CQ)	90.2 ± 0.3	87.8 ± 0.4	89.7 ± 0.3	62.0 ± 1.0	61.1 ± 1.0	62.1 ± 1.0	3.8 ± 0.7	20.0
MLCM (5 ML-based CQs)	92.5 ± 0.2	89.6 ± 0.4	91.7 ± 0.3	66.0 ± 1.0	63.1 ± 1.1	64.3 ± 1.1	5.4 ± 0.9	21.1
MLCM (10 ML-based CQs)	93.9 ± 0.3	91.0 ± 0.5	93.2 ± 0.3	68.8 ± 1.0	66.3 ± 1.0	67.5 ± 1.0	5.2 ± 0.8	20.9
MLCM (15 ML-based CQs)	94.8 ± 0.3	92.0 ± 0.4	94.2 ± 0.3	70.6 ± 1.0	67.9 ± 1.0	69.1 ± 1.0	5.0 ± 0.9	22.1
MLCM (20 ML-based CQs)	95.5 ± 0.3	92.4 ± 0.5	94.5 ± 0.4	71.7 ± 1.1	69.1 ± 1.0	70.3 ± 0.9	5.5 ± 0.9	25.6
MLCM-LR (1 ML-based CQ)	90.2 ± 0.3	87.6 ± 0.4	89.5 ± 0.3	61.8 ± 1.1	60.5 ± 1.1	61.5 ± 1.0	4.9 ± 0.8	0.7
MLCM-LR (5 ML-based CQs)	91.5 ± 0.3	89.1 ± 0.4	90.9 ± 0.3	63.9 ± 1.1	61.7 ± 1.2	63.1 ± 1.2	4.8 ± 0.8	0.8
MLCM-LR (10 ML-based CQs)	92.5 ± 0.2	90.2 ± 0.5	91.9 ± 0.3	65.4 ± 1.1	63.6 ± 1.1	64.4 ± 1.1	4.7 ± 0.8	0.8
MLCM-LR (10 ML-based CQs)	93.7 ± 0.3	91.0 ± 0.4	93.0 ± 0.3	68.1 ± 0.9	66.3 ± 0.9	66.9 ± 0.9	4.7 ± 0.8	0.8
MLCM-LR (20 ML-based CQs)	94.4 ± 0.3	92.1 ± 0.4	94.0 ± 0.3	69.6 ± 0.9	67.4 ± 1.1	69.1 ± 1.0	4.9 ± 0.9	0.8

Table 38. Comparison of CM and MLCM in Stages 1–3 for *additive* preferences, a supply ratio of 1.5, 9 popular courses, and default parameterization for reporting mistakes. We normalize all results by the average utility of CM* after Stage 1. Shown are averages in % over 100 runs and 95% CIs. Additionally, we present the oversubscription (in number of course seats) after Stage 1 (Overs.) and total runtime (in hours) per run.

P.2 Reporting Mistakes Ablation Study for Additive Preferences

In this subsection, we present in Figures 9 and 10 the reporting mistakes ablation study when students' true utility functions are restricted to be *additive/linear*. Please see Section 7.3 for details on the experiment setup.

Similar to the results in Section 7.3, as γ increases, the performance of both CM and MLCM monotonically decreases. MLCM significantly outperforms CM for all $\gamma \in [0.5, 1.5]$. As γ increases, the relative performance gap of the two mechanisms gets significantly larger. Those results could be further improved by retuning MLCM's hyperparameters for each value of γ.

P.3 Should Individual Students Opt Into MLCM?

In this section, we present the experimental results measuring a students expected gain from opting into MLCM when students' true utility functions are restricted to be *additive/linear*. Recall, that we use 20 instances and 100 students per instance. We report averages over those 2000 students. Please see Section 7.4 for more details on the experiment setup.

No Other Student to Opt into the ML Feature. Table 39 shows the results when all other students opt out of MLCM's ML feature. We observe that the *expected relative gain* from opting in is at least 8.4% (across both supply ratios). Furthermore, the student prefers the "MLCM schedule" in at least 71.5% of the cases, while she prefers the "CM schedule" in at most 6.3% of the cases. As the number

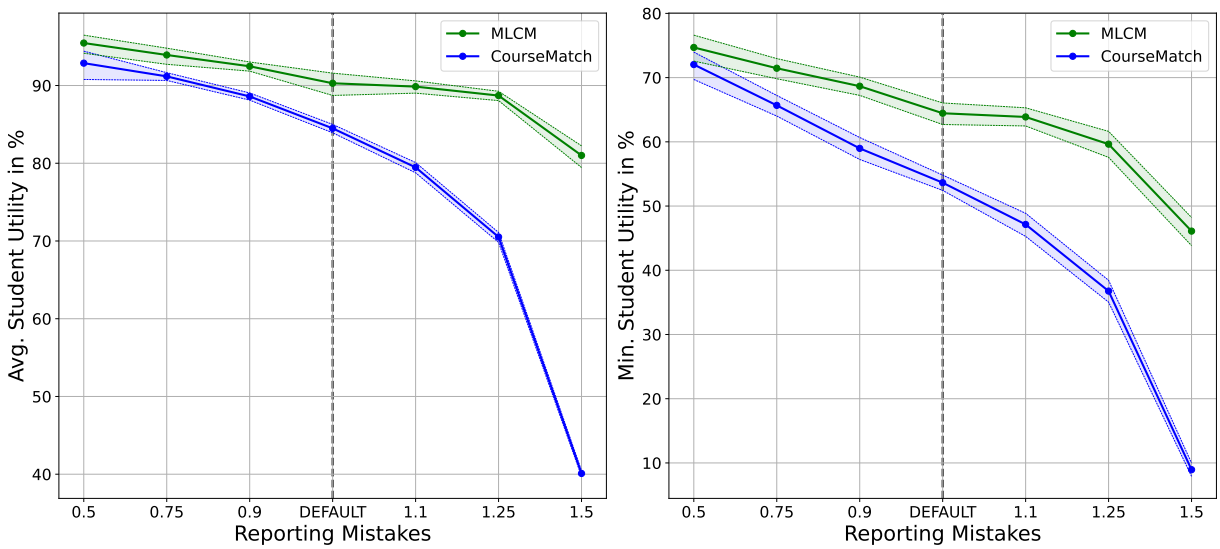


Fig. 9. Reporting mistakes ablation experiment for *additive* preferences for a supply ratio of 1.25 and 9 popular courses. Shown are average results in % for the final allocation over 50 runs including 95% CI.

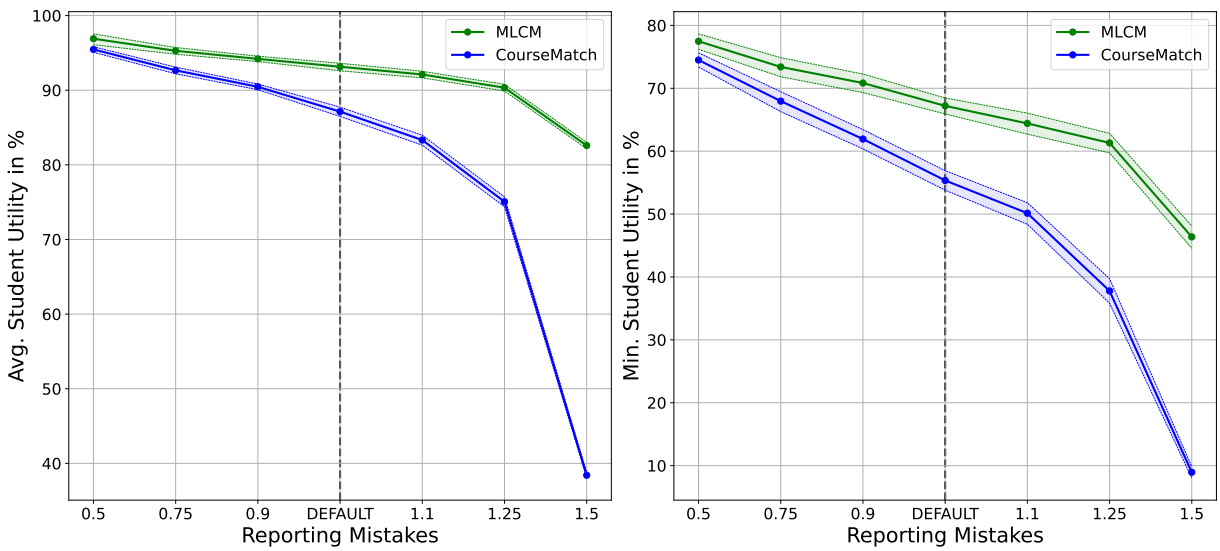


Fig. 10. Reporting mistakes ablation experiment for *additive* preferences for a supply ratio of 1.5 and 9 popular courses. Shown are average results in % for the final allocation over 50 runs including 95% CI.

of CQs the student answers increases, the benefit from opting into MLCM's ML feature becomes even larger.

Setting			Preferred Mechanism			Gain from Opting Into MLCM		
SR	#PoP	#CQs	MLCM	CM	Indiff.	Expected	if pref MLCM	if pref CM
1.25	9	10	77.60%	5.75%	16.65%	10.98%	14.54%	-5.46%
1.25	9	15	81.35%	6.35%	12.30%	12.32%	15.52%	-4.93%
1.25	9	20	85.30%	6.10%	8.60%	13.58%	16.21%	-4.08%
1.50	9	10	71.55%	6.30%	22.15%	8.42%	12.30%	-6.02%
1.50	9	15	77.85%	5.45%	16.70%	9.81%	12.95%	-4.39%
1.50	9	20	81.70%	6.50%	11.80%	10.56%	13.30%	-4.89%

Table 39. Expected gain of opting into MLCM's ML feature when no other student opts in for *additive* preferences. Shown are average results across 2000 students per setting (SR,#PoP,#CQs). CIs for all metrics are ≈ 0.

All Other Students to Opt into the ML Feature. Table 40 shows the results when all other students opt into MLCM's ML feature. We observe that the *expected relative gain* from opting in is at least 7.6% (across both supply ratios). Furthermore, the student prefers the "MLCM schedule" in at least 68.5% of the cases, while she prefers the "CM schedule" in at most 8.4% of the cases. As the number of CQs the student answers increases, the benefit from opting into MLCM's ML feature becomes even larger.

Setting			Preferred Mechanism			Gain from Opting Into MLCM		
SR	#PoP	#CQs	MLCM	CM	Indiff.	Expected	if pref MLCM	if pref CM
1.25	9	10	74.55%	8.4%	17.05%	10.05%	14.12%	-5.80%
1.25	9	15	77.6%	8.4%	14.00%	10.94%	14.73%	-5.91%
1.25	9	20	81.7%	8.15%	10.15%	11.79%	14.91%	-4.67%
1.50	9	10	68.5%	7.95%	23.55%	7.59%	11.70%	-5.75%
1.50	9	15	73.95%	8.65%	17.40%	8.54%	12.20%	-5.95%
1.50	9	20	77.8%	8.65%	13.55%	9.06%	12.19%	-5.56%

Table 40. Expected gain of opting into MLCM's ML feature when all other students also opt in for *additive* preferences. Shown are average results across 2000 students per setting (SR,#PoP,#CQs). CIs for all metrics are ≈ 0.

Curriculum Vitae

Personal Information

Name	Jakob Weissteiner
Date of Birth	November 22, 1992
Place of Birth	Grieskirchen, Austria
Nationalities	Austrian

Education

February 2019 – February 2023	*Doctoral program at the University of Zurich* Department of Informatics Computation and Economics Research Group
September 2016 – September 2018	*M.Sc. in Quantitative Finance (passed with distinction)* Vienna University of Economics and Business Vienna, Austria
October 2015 – January 2018	*M.Sc. in Mathematics (passed with distinction)* Specialization: Financial Mathematics Technical University of Vienna Vienna, Austria
October 2012 – September 2015	*B.Sc. in Mathematics (passed with distinction)* Specialization: Financial and Actuarial Mathematics Technical University of Vienna Vienna, Austria

Professional Experience

October 2021 – September 2022	*Workflow Chair* ACM Conference on Economics and Computation (EC'22) Zurich, Switzerland
September 2017 – July 2018	*Junior Data Scientist* Raiffeisen Bank International AG Department of Advanced Analytics Vienna, Austria
October 2015 – March 2017	*Freelancer* Raiffeisen Bank International AG Department of Investment Finance Vienna, Austria