Efficient Algorithms, Spring 2021

1. Introduction & Course Organisation

Prof. Dan Olteanu

DaST

Data • (Systems+Theory)

University of Zurich UZH

February 22, 2021

https://lms.uzh.ch/url/RepositoryEntry/16979230773

Why Take This Course?

Many computational problems in Computer Science are deeply related

- They are widely taught and researched **separately**
- They are formalised **differently** and come with **specific** algorithmic solutions

Why Take This Course?

Many computational problems in Computer Science are deeply related

- They are widely taught and researched **separately**
- They are formalised **differently** and come with **specific** algorithmic solutions

Yet, they all admit **one** natural and elegant formalisation and algorithmic solution

Why Take This Course?

Many computational problems in Computer Science are deeply related

- They are widely taught and researched **separately**
- They are formalised **differently** and come with **specific** algorithmic solutions

Yet, they all admit **one** natural and elegant formalisation and algorithmic solution

This course is:

- a journey from such problems to their unified formalisation and solution
- the first of its kind – to the best of our knowledge
- closely follows research done by many (ourselves included) across CS

A Very Small Sample of the Computational Problems under Consideration

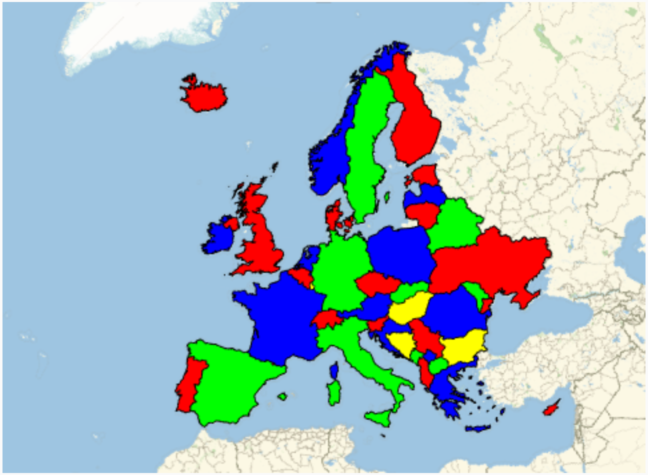
Constraint Satisfaction Problems (CSPs)

	3							
			1	9	5			
		8					6	
8				6				
4			8					1
				2				
	6					2	8	
			4	1	9			5
							7	

Sudoku: Find a completion of the grid with numbers 1..9 such that each number occurs once in a row, column, and 3x3 block

A Very Small Sample of the Computational Problems under Consideration

Satisfiability (SAT, #SAT)



Map colouring: Can we colour the map of Europe using 4 colours?

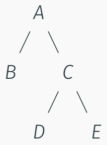
How many satisfying assignments of colours to European countries are there?

A Very Small Sample of the Computational Problems under Consideration

Query Evaluation in Relational Databases (QE)

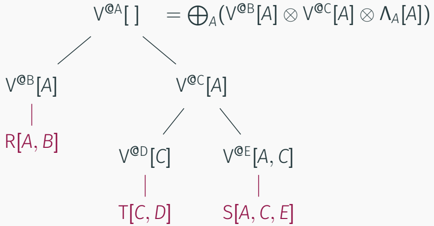
Variable order

$$dep(A) = \emptyset$$
$$dep(B) = \{A\}$$
$$dep(C) = \{A\}$$
$$dep(D) = \{C\}$$
$$dep(E) = \{A, C\}$$



$\Rightarrow$

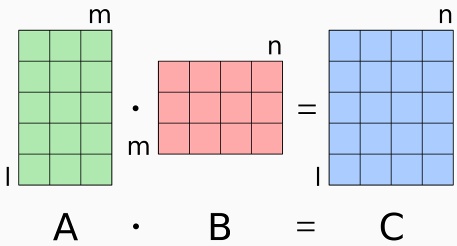
View tree



Given a database with relations R, S, and T, find the result of their natural join

A Very Small Sample of the Computational Problems under Consideration

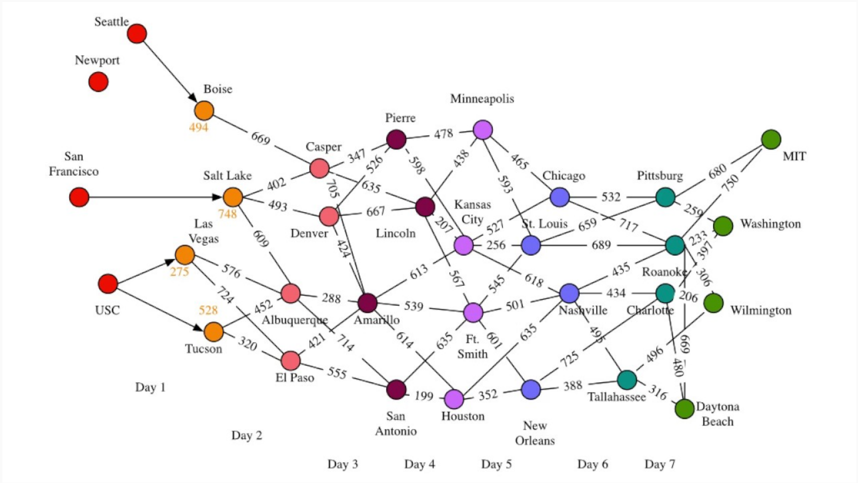
Matrix Chain Multiplication (MCM)



Given a chain of matrices with matching dimensions, compute the matrix representing their multiplication

A Very Small Sample of the Computational Problems under Consideration

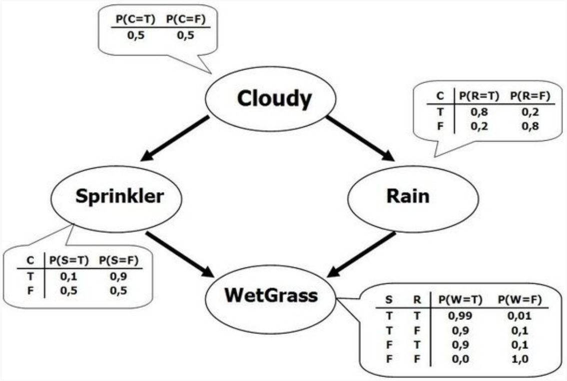
Viterbi Algorithm (VA)



Find the most likely sequence of hidden states that results in a sequence of observed events, e.g., in hidden Markov models.

A Very Small Sample of the Computational Problems under Consideration

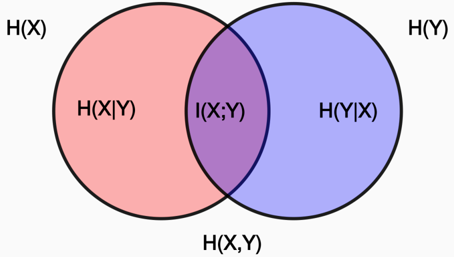
Inference in Bayesian Networks and Probabilistic Graphical Models



What is the probability that the grass is wet given that the sprinkler was off and it did not rain?

A Very Small Sample of the Computational Problems under Consideration

Cost Functions and Statistics for Machine Learning (ML)



$$K_{XX} = \begin{bmatrix} \mathrm{E}[(X_1 - \mathrm{E}[X_1])(X_1 - \mathrm{E}[X_1])] & \mathrm{E}[(X_1 - \mathrm{E}[X_1])(X_2 - \mathrm{E}[X_2])] & \cdots & \mathrm{E}[(X_1 - \mathrm{E}[X_1])(X_n - \mathrm{E}[X_n])] \\ \mathrm{E}[(X_2 - \mathrm{E}[X_2])(X_1 - \mathrm{E}[X_1])] & \mathrm{E}[(X_2 - \mathrm{E}[X_2])(X_2 - \mathrm{E}[X_2])] & \cdots & \mathrm{E}[(X_2 - \mathrm{E}[X_2])(X_n - \mathrm{E}[X_n])] \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{E}[(X_n - \mathrm{E}[X_n])(X_1 - \mathrm{E}[X_1])] & \mathrm{E}[(X_n - \mathrm{E}[X_n])(X_2 - \mathrm{E}[X_2])] & \cdots & \mathrm{E}[(X_n - \mathrm{E}[X_n])(X_n - \mathrm{E}[X_n])] \end{bmatrix}$$

Compute the empirical entropy, the pairwise mutual information and covariance of the variables describing the data.

Did You Get the Gist?

How to Uniformly Approach These Problems?

1. Unified language able to express a wide range of such problems

How to Uniformly Approach These Problems?

1. Unified language able to express a wide range of such problems

2. Algorithm to efficiently compute any expression in the language

How to Uniformly Approach These Problems?

1. Unified language able to express a wide range of such problems

   - Our language in this course: Functional Aggregate Queries [Ngo et al 2016]
   - Several previous incarnations in AI & Information Theory, e.g.,

     Marginalise a Product Function [Aji & McEliece 2000]

2. Algorithm to efficiently compute any expression in the language

How to Uniformly Approach These Problems?

1. Unified language able to express a wide range of such problems

- Our language in this course: Functional Aggregate Queries [Ngo et al 2016]
- Several previous incarnations in AI & Information Theory, e.g.,

  Marginalise a Product Function [Aji & McEliece 2000]

2. Algorithm to efficiently compute any expression in the language

- Decompose the problem into smaller tasks based on its structure

  [Gottlob et al 1990]
- Arbitrary structure is reduced to tree structure worst-case optimally

  [Ngo et al 2012, Veldhuizen 2014]
- Best known algorithms used for tree structures [Yannakakis 1982]

Learning Outcomes

On completion of the course you should be able to

- Learn to formalise problems using various formalisms
- Analyse computational complexity of problems using a toolbox of techniques that exploit the algebraic and combinatorial structure of the problems
- Familiarise themselves with simple yet powerful algorithms
- Implement and benchmark such algorithms

Course Information

Website: `https://lms.uzh.ch/url/RepositoryEntry/16979230773`

Zoom: `https://uzh.zoom.us/j/95108110165?pwd=MTdRMU5QeDhjMXQ5RThlYWdndTJIZz09`

- Meeting ID: 951 0811 0165, Passcode: 261401
- You need to log in to UZH zoom to access the course room
- By default, no sound and video for student participants
- All sessions recorded and posted on OLAT
- Questions asked via chat room, addressed in the session
- All questions and answers will be copied to the course forum on OLAT

Sessions:

- Mondays 16.15 - 17.45
- Several Mondays are off: April 5; April 19; May 24
- 12 sessions in total: 9 lectures, 2 exercises, 1 practical

Course Materials

It is the first time this course is given!

Materials posted on OLAT:

1. Research monographs/surveys/papers on relevant topics
   - Cover a wealth of ideas across Computer Science
   - Cover theory, algorithmic development, and systems aspects
2. Slides made available before the lectures

Lecture Topic 1: Expressing Computational Problems

- Semirings and rings. Static versus dynamic computation over rings
- FAQs: Functional Aggregate Queries over semirings
- Examples:
    - Relational databases: Query evaluation
    - CSP: Constraint satisfaction problems
    - SAT, count-SAT: Satisfiability problems
    - PGM: Inference and learning in probabilistic graphical models
    - Linear algebra: Matrix chain multiplication, discrete Fourier transform
    - Machine learning: Cost functions and gradients for learning over structured data
    - . . .

Lecture Topic 2: Structural Decompositions and Width Measures

- Tree decompositions
- Hypertree decompositions
- Decompositions as variable orders
- Width measures: treewidth, hypertree width, fractional hypertree width, FAQ width, submodular width
- Special cases: Acyclic (α, β, γ, berge). Free-connex. Hierarchical

Lecture Topic 3: Solving Joins Optimally

- Instance output size versus worst-case output size
- Suboptimality of state-of-the-art join algorithms for cyclic queries
- Worst-case size of a join. The AGM bound
- The Yannakakis algorithm for acyclic joins
- Worst-case optimal join algorithms: LeapFrog TrieJoin and Generic Join
- Complexity refinement: Preprocessing time and enumeration delay

Lecture Topic 4: Solving Functional Aggregate Queries

- FDB Algorithm for FAQ over one semiring
  - Generalises worst-case joins from listing to factorised representations of output
  - Top-down dynamic programming with memoisation
- InsideOut Algorithm for FAQ over arbitrary semirings
  - Bottom-up dynamic programming, equivalent to FDB for one semiring

Lecture Topic 5: Solving Functional Aggregate Queries under Updates

- The Incremental View Maintenance (IVM) Problem
- First-order versus higher-order delta processing
- IVM Algorithms: DBToaster, F-IVM
- Adaptive update execution: IVM$^{\epsilon}$
    - Worst-case optimal counting of triangles in graphs
    - Trade-offs between update time and enumeration delay

Lectures: Format

Lecturers: **Prof. Dan Olteanu** and **Dr. Ahmet Kara**

- Online lectures
- Lectures recorded and posted in OLAT
- Q&A: Live during lecture, asynchronous in OLAT forum

Classes

Tutor: **Dr. Ahmet Kara**

Class Topics:

1. Express Problems as Functional Aggregate Queries; Decompositions
2. Solving Functional Aggregate Queries

Class Format:

- Exercise sheets made available at least a week before the class, not marked
- Exercises discussed in class, recordings posted in OLAT
- Live Q&A in class + Q&A in OLAT forum

Practical Task

Demonstrator: **Haozhe Zhang**

1. Implement the worst-cast optimal join algorithm Leapfrog Triejoin (LFTJ)
2. Benchmark its performance on public and private joins and databases
3. Extend LFTJ with semiring computation to support aggregates over joins
   e.g., count queries, queries with projections, and covariance matrix

Practical Task: Automated Tests and Leaderboard

- We will maintain a leaderboard with the runtime performance scores of your query evaluation engines on private databases
- This may guide your effort on improving your engines
- You submit your code, it gets compiled and tested automatically
- Feedback on correctness and runtime performance
- If the correctness test is passed, then then runtime performance is added to the leaderboard.

What Makes a Successful Practical Submission

- Success = Pass the correctness tests for all given joins on both the public and private databases
- The private databases have the same database schemas as the public databases, but different content. They are not disclosed to the students but will be used to test the correctness and runtime performance of your code.
- Submission deadline for the practical: EOD (Zurich time) on June 12, 2021
- Each student must work independently on their practical. It is not allowed to use existing code written by others (fellow students or otherwise).
- A successful practical submission before the deadline is a prerequisite for sitting the written exam and therefore passing the course.

Practical Task: Bonus points

- Bonus points awarded to students whose code performs above average
- Allocation of bonus points to be decided after the submission deadline
- Max bonus points = 20
- Bonus points will be added to the points (0-100) obtained in the written exam

Written Exam

- Date: **Monday 21/06/2021, 16:15 - 17:45**
- Open book, online (expected via EPIS)
- Expect to be asked to formalise a problem in FAQ, analyse its complexity by decomposing it and propose an efficient strategy to evaluate it.
- Expect similar in difficulty yet not identical exam questions
    - you need to know what you are doing
    - copying someone else's solution is futile and time consuming

Is this Course Right for You?

⚠ The topics addressed in this course are mathematically rigorous, making use of computational complexity, algorithm design and analysis, formal proof & reasoning

Lots of complexity results and derivations

Ideas from many distinct areas of Computer Science, some at the frontier of research

Prerequisites:

- Good background in: Computer Science at BSc level

  E.g., **Foundations of Computing I + II, Informatik II**

- **Programming skills needed**: See the practical task

Join Us!

DaST

Data•(Systems+Theory)

Student projects & theses:

`https://www.ifi.uzh.ch/en/dast/projects.html`

Research positions:

`https://www.ifi.uzh.ch/en/dast/jobs.html`