Finding Clearing Payments in Financial Networks with Credit Default Swaps is PPAD-complete[*]

Steffen Schuldenzucker Sven Seuken Stefano Battiston
University of Zurich

First version: June 6, 2016
This version: December 28, 2016

Abstract

We consider the problem of clearing a system of interconnected banks that have been exposed to a shock on their assets. Eisenberg and Noe (2001) showed that when banks can only enter into simple debt contracts with each other, then a clearing vector of payments can be computed in polynomial time. In this paper, we show that the situation changes radically when banks can also enter into *credit default swaps* (*CDSs*), i.e., financial derivative contracts that depend on the default of another bank. We prove that computing an approximate solution to the clearing problem with sufficiently small constant error is PPAD-complete. To do this, we demonstrate how financial networks with debt and CDSs can encode arithmetic operations such as addition and multiplication. Further analysis of our construction reveals that already determining which banks are in default is a PPAD-complete problem. This makes apparent the origin of the additional complexity introduced by CDSs: that banks may profit from the ill-being of other banks.

1 Introduction

We consider systems of banks (or other financial institutions) that are connected by financial contracts. Due to a shock, some of the banks may not be able to meet their obligations towards other banks, thus forcing them into bankruptcy (or *default*). We study the *clearing problem* in this setting, i.e., the problem of computing a collection of payments between each pair of banks that are in accordance with standard bankruptcy law. Since banks' contractual relationships banks can be complex and are often cyclic, designing good clearing mechanisms is a nontrivial task.[1]

In their seminal paper, Eisenberg and Noe (2001) devised an efficient clearing mechanism for financial systems. Their mechanism relies on the assumption that banks can only enter

[*] Steffen Schuldenzucker, Sven Seuken: Department of Informatics, University of Zurich, Switzerland, Email: {schuldenzucker,seuken}@ifi.uzh.ch. Stefano Battiston: Department of Banking and Finance, University of Zurich, Switzerland, Email: stefano.battiston@uzh.ch. We would like to thank (in alphabetical order) Vitor Bosshard, Gianluca Brero, Yu Cheng, Marc Chesney, Constantinos Daskalakis, Marco d'Errico, Timo Mennle, Thomas Noe, and Joseph Stiglitz for helpful comments on this work. Furthermore, we are thankful for the feedback received from an anonymous reviewer at ITCS'17. Any errors remain our own. This project has received funding from the European Union's Horizon 2020 research and innovation programme under the DOLFINS project, grant agreement No 640772.

[1] We liberally borrow from our own earlier work (Schuldenzucker et al., 2016) for parts of the introduction, related work, and the formal model.

into simple *debt contracts*, i.e., loans from one bank to another. We argue, however, that the growing importance of financial derivative contracts makes it necessary to reconsider the question if today's financial networks can still always be efficiently cleared. Specifically, *credit default swaps (CDSs)*, which are contracts that are only triggered when a reference entity goes into default, have received only little attention in a network context so far. Market participants use CDSs to insure themselves against a default of the reference entity or to place a speculative bet on this event. Because the reference entity can itself be a financial institution, CDSs create new dependencies that do not exist in pure debt networks.

In prior work (Schuldenzucker et al., 2016), we have shown that if no money is lost in the bankruptcy process (i.e., banks do not incur *default costs*), then clearing payments always exist. However, our proof is via a non-constructive fixed point argument so that the question remained open if one could devise an efficient algorithm to actually *find* a clearing payment vector in this case.

In the present paper, we answer this question in the negative: we show that the problem FindClearing of finding an approximately clearing vector of payments in a financial network with debt and CDSs and without default costs is PPAD-complete even for a sufficiently small constant error bound. This implies that the problem does not have a polynomial-time approximation scheme unless P=PPAD and thus needs to be considered computationally infeasible in the worst case. We extend our result by showing that already the problem of determining the set of banks that default in any (approximate) solution is PPAD-complete.

More in detail, we proceed as follows: we first describe a simplified variant of our model from (Schuldenzucker et al., 2016) that only applies to the case without default costs (Section 3). We next argue that since solutions to the clearing problem can be irrational, FindClearing needs to be considered as an approximation problem. We define the notion of an *ε-approximate clearing vector* and we show that it makes the FindClearing problem well-posed and a member of PPAD (Section 4). Having done this, we describe our main contribution, namely a reduction from the problem of finding an approximate solution of a generalized circuit to FindClearing, establishing that FindClearing is PPAD-hard. We do this by composing *financial system gadgets*, i.e., fragments of financial networks that encode specific operations such as addition, subtraction, scaling, comparison, and Boolean operations like NOT and OR (Section 5). Our financial system gadgets have a particularly simple structure; so simple in fact that as soon as one knows which banks are in default and which are not, one can efficiently compute a clearing vector. From this we conclude that it is already PPAD-complete to determine which banks are in default, and that this even remains true in the special case where liabilities never form cycles, CDS writers never default, and banks are not at all exposed to their contract counterparties (Section 6).

Our results contribute to the literature on complexity in financial networks (Battiston et al., 2016). By measuring complexity in terms of computational complexity, we are able to accurately describe the effect of introducing a new class of financial products into the system. Our hardness result has practical relevance for *stress tests* in which regulators such as the European Central Bank evaluate the stability of the financial system under an array of adverse economic scenarios. We argue that, because of the complex interdependencies in real financial networks, future stress tests should take into account network effects, which would essentially require regulators to compute clearing payments. However, as we show, this problem is computationally infeasible. While we do not specify below which approximation quality ε this happens and it may indeed be the case that the clearing problem can be solved efficiently for values of ε demanded in practice, the fact that there is no polynomial-time approximation scheme (unless P=PPAD) tells us that such algorithms, if they exist, need to be discovered for individual values of ε: there is no general recipe to construct an efficient stress testing algorithm given the desired ε.

2 Related Work

Prior work on financial networks has primarily focused on financial contagion, i.e., how small shocks may amplify to system-wide losses and which network topologies are particularly susceptible to such effects. Researchers have investigated trade-offs between (stabilizing) diversification and (destabilizing) contagion effects based on network completeness (Allen and Gale, 2000), portfolio diversification and integration (Elliott et al., 2014), or the magnitude of shocks (Acemoglu et al., 2015). Several measures for an individual bank's contribution to systemic risk have been proposed: a distance measure by Acemoglu et al. (2015) and two network algorithms: DebtRank by Battiston et al. (2012) and LASER by Hu et al. (2012).

The clearing problem was first studied by Eisenberg and Noe (2001), who showed that in debt networks, clearing payments always exist and can be computed in polynomial time. Rogers and Veraart (2013) extended their result to debt networks with default costs. Since all aforementioned pieces of work use a weighted graph as the underlying model of the financial network, they cannot accurately represent the *ternary* relationship introduced by a credit default swap between the holder, the writer, and the reference entity. We filled this gap in prior work (Schuldenzucker et al., 2016) by devising a new model that *could* represent networks of debt and CDSs. We then showed that the clearing problem in these networks is significantly more complex than in the debt-only case: if default costs are present, then clearing payments may not exist and it is NP-hard to decide if they do. In this work, we study the case without default costs.

A field that has only developed recently is the application of computational complexity theory to financial markets. Arora et al. (2010) and Zuckerman (2011) investigated the cost of asymmetric information in financial derivatives markets with computationally bounded agents. Braverman and Pasricha (2014) provided computational hardness results on fair pricing of compound options. Hemenway and Khanna (2015) showed that in Elliott et al.'s model, it is computationally infeasible to determine the distribution of a given total negative shock to the banks that has the worst impact in terms of value. In contrast, we prove that in financial networks with CDSs, it is already computationally infeasible to determine the impact of a *known* distribution of shocks to banks.

The PPAD complexity class was introduced by Papadimitriou (1994) to capture the hardness of a class of total search problems. Daskalakis et al. (2005) developed the proof technique of reduction from generalized circuits to show that finding a Nash equilibrium in a graphical game with exponential accuracy is PPAD-complete. Their result was subsequently strengthened and extended by Chen et al. (2006), Daskalakis (2013), and Rubinstein (2015) while at the same time, the theory of generalized circuits was refined. Our work builds on these techniques, in particular on Rubinstein's (2015) hardness result for generalized circuits with constant accuracy. To the best of our knowledge, we are the first to implement generalized circuits using financial networks and, together with our prior work on the case with default costs, we are the first to present a computational complexity result for the clearing problem.

3 Formal Model

Our model is based on the model by Eisenberg and Noe (2001), which was restricted to debt contracts. We define an extension to credit default swaps. We adjust the notation where necessary. The model used in this paper is a simplified version of the one used in (Schuldenzucker et al., 2016), where default costs were also modeled. In this paper, we only consider financial systems without default costs.

Consider a two-period model:

- *Period 0:* Each bank receives an initial endowment called its *external assets*. Banks enter into bilateral contracts with each other. No bank is in default.
- *Period 1:* Banks' external assets change due to an exogenous shock. All banks make payments according to their contractual commitments from period 0 and the new external assets.

We define the elements of the financial system in period 1.

Banks and External Assets. We denote by N a finite set of n *banks*. For any bank $i \in N$ let $e_i \geq 0$ denote the *external assets* of i as of period 1. Let $e = (e_i)_{i \in N}$ denote the vector of all external assets.

Contracts. There are two types of contracts: *debt contracts* and *credit default swap contracts (CDSs)*. Every contract gives rise to a conditional obligation to pay a certain amount, called a *liability*, from its *writer* to its *holder*. Banks that are unable to fulfill their obligations are said to be *in default*. The *recovery rate* r_i of a bank i is the share of its liabilities it is able to pay. Thus, $r_i = 1$ if i is not in default and $r_i < 1$ if i is in default. Let $r = (r_i)_{i \in N}$ denote the vector of all recovery rates.

A *debt contract* obliges the writer i to unconditionally pay a certain amount to the holder j in period 1. This amount is called the *notional* of the contract and is denoted by $c^{\emptyset}_{i,j}$. A *credit default swap* obliges the writer i to make a conditional payment to the holder j in period 1. The amount of this payment depends on the default of a third bank k, called the *reference entity*. Specifically, the payment amount of the CDS contract from i to j with reference entity k and *notional* $c^k_{i,j}$ is $c^k_{i,j} \cdot (1 - r_k)$.

Note that when banks enter into contracts, there would typically be an initial payment. For example, debt contracts arise because the holder lends an amount of money to the writer, and holders of CDSs pay a premium to obtain them. In our model, any initial payments have been made in period 0 and are implicitly reflected by the external assets.

The contractual relationships between all banks are represented by a 3-dimensional matrix $c = (c^k_{i,j})_{i \in N,\, j \in N,\, k \in N \cup \{\emptyset\}}$. The entry $c^{\emptyset}_{i,j}$ is the total notional of the debt contracts from i to j and the entry $c^k_{i,j}$ for $k \in N$ is the total notional of CDS contracts from i to j with reference entity k. Zero entries indicate the absence of the respective contract. The set of contracts can alternatively be represented as an edge-weighted directed hypergraph.

We require that no bank enters a contract with itself (i.e., $c^k_{i,i} = 0$ for all $k \in N \cup \{\emptyset\}$ and $i \in N$). We further require that any bank that is a reference entity in a CDS must be a writer of some debt contract (i.e., if $\sum_{k,l \in N} c^i_{k,l} > 0$, then $\sum_{j \in N} c^{\emptyset}_{i,j} > 0$ for all $i \in N$). Both requirements are needed to rule out pathological cases.They are always assumed to hold in the following.

For any bank i, the *creditors of i* are the banks that are holders of contracts for which i is the writer, i.e., the banks to which i owes money. Conversely, the *debtors of i* are the writers of contracts of which i is the holder, i.e., the banks by which i is owed money. Note that the two sets can overlap: for example, a bank could hold a CDS on one reference entity while writing a CDS on another reference entity, both with the same counterparty.

Financial System Without Default Costs. A *financial system without default costs* (or, for the purpose of this paper just a *financial system*) is a tuple (N, e, c) where N is a set of banks, e is a vector of external assets, and c is a 3-dimensional matrix of contracts. The *length* of a financial system is the total number of bits needed to describe the tuple, including all numeric values.

Liabilities, Payments, and Assets. For two banks i, j and a vector of recovery rates r, the *liabilities of i to j at r* are the amount of money that i has to pay to j if recovery rates in the financial system are given by r, denoted by $l_{i,j}(r)$. They arise from the aggregate of all debt- and CDS contracts from i to j.

$$l_{i,j}(r) := c^{\emptyset}_{i,j} + \sum_{k \in N} (1 - r_k) \cdot c^k_{i,j}$$

The *total liabilities of i at r* are the aggregate of all liabilities that i has toward other banks given the recovery rates r, denoted by $l_i(r)$.

$$l_i(r) := \sum_{j \in N} l_{i,j}(r)$$

The actual *payment* $p_{i,j}(r)$ from i to j at r can be lower than $l_{i,j}(r)$ if i is in default. By the *principle of proportionality*, a bank that is in default makes payments for its contracts in proportion to the respective liabilities.

$$p_{i,j}(r) := r_i \cdot l_{i,j}(r)$$

The *total assets* $a_i(r)$ of a bank i at r consist of its external assets e_i and the incoming payments to i.

$$a_i(r) := e_i + \sum_{j \in N} p_{j,i}(r)$$

Clearing Recovery Rate Vector. Following Eisenberg and Noe (2001), we call a recovery rate vector r *clearing* if the payments $p_{i,j}(r)$ satisfy the following three principles:

1. *Absolute Priority:* Banks with sufficient assets pay their liabilities in full.
2. *Limited Liability:* Banks with insufficient assets to pay their liabilities are in default and pay all of their assets to creditors.
3. *Proportionality:* In case of default, payments to creditors are made in proportion to the respective liability.

The principle of proportionality is automatically fulfilled by the definition of the payments $p_{i,j}(r)$. The other two principles lead to the following formal definition.

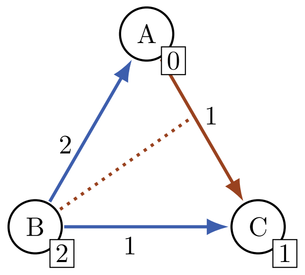
Definition 3.1 (Clearing Recovery Rate Vector)**.** A recovery rate vector r is called *clearing* for a financial system without default costs $X = (N, e, c)$ if for all banks $i \in N$ we have

$$\sum_{j \in N} p_{i,j}(r) \; = \; \min\left(a_i(r), l_i(r)\right). \tag{1}$$

We also call a clearing recovery rate vector a *solution*.

Remark 3.2 (Clearing Payments and Recovery Rates). It is an equivalent point of view whether one considers clearing *payments* or clearing *recovery rates*. The previous definition is equivalent to requiring that the recovery rates of banks with positive liabilities are 1 in case they are not in default ($a_i(r) \geq l_i(r)$) and $a_i(r)/l_i(r)$ in case they are in default ($a_i(r) < l_i(r)$). The recovery rates of banks with zero liabilities are left unconstrained between 0 and 1. While this might seem unintuitive at first, it corresponds exactly to the definition of the recovery rate: r_i is the share of its total liabilities that bank i can pay. If there are no liabilities, then this value is not well-defined. Forcing the recovery rate to 1 in these cases would introduce an artificial discontinuity because $\frac{a_i(r)}{l_i(r)}$ may converge to a value strictly below 1 while $l_i(r) \to 0$.

Figure 1 Example financial system.



Example and Visual Language. Figure 1 shows a visual representation of an example financial system. There are three banks $N = \{A, B, C\}$, drawn as circles, with external assets of $e_A = 0$, $e_B = 2$, and $e_C = 1$, drawn as rectangles on the banks. Debt contracts are drawn as blue arrows from the writer to the holder and they are annotated with the notionals $c^{\emptyset}_{B,A} = 2$ and $c^{\emptyset}_{B,C} = 1$. CDS contracts are drawn as orange arrows where a dashed line connects to the reference entity and are also annotated with notionals: $c^{B}_{A,C} = 1$. A clearing recovery rate vector for this example is given by $r_A = 1$, $r_B = \frac{2}{3}$, and $r_C = 1$. The liabilities arising from this recovery rate vector are $l_{B,A}(r) = 2$, $l_{B,C}(r) = 1$, and $l_{A,C}(r) = \frac{1}{3}$. The clearing payments are $p_{B,A}(r) = \frac{4}{3}$, $p_{B,C} = \frac{2}{3}$, and $p_{A,C}(r) = \frac{1}{3}$. This is the only solution for this system.

We stress that we are not concerned with the question whether or not it is "rational" for the banks to form a certain financial system: contracts might have been entered for reasons exogenous to the system, or simply for cash transfers at time 0.

Theorem 3.3 (Schuldenzucker et al. (2016))**.** *Any financial system without default costs has a clearing recovery rate vector.*

Proof Outline. The proof rests on the fact that the right-hand side of equation (1) in Definition 3.1 is a continuous function of r. Let $[0,1]^N$ be the set of the $|N|$-tuples with elements in $[0,1]$, labeled by the elements of N. Consider the function ρ defined by

$$\rho : [0,1]^N \to 2^{[0,1]^N}$$
$$\rho(r) := \bigtimes_{i=1}^{N} \rho_i(r)$$

where

$$\rho_i : [0,1]^N \to 2^{[0,1]}$$
$$\rho_i(r) := \begin{cases} \{\min(1, \frac{a_i(r)}{l_i(r)})\} & \text{if } l_i(r) > 0 \\ [0,1] & \text{if } l_i(r) = 0. \end{cases}$$

By Remark 3.2 a recovery rate vector r is clearing iff it is a fixed point of the set-valued function ρ, i.e., iff $r \in \rho(r)$. To show that such a fixed point exists, one applies Kakutani's fixed point theorem for set-valued functions with a closed graph (a generalization of Brouwer's fixed point theorem for continuous functions). $\square$

4 Defining the FindClearing Search Problem

We have just seen a non-constructive proof that a solution for a given financial system always exists. In this section, we define the corresponding total search problem. This is not a trivial task: there are financial systems where all solutions are irrational numbers (a simple example is provided in Appendix A) and these solutions have infinitely many non-periodic digits, so a computer could not even *output* a solution in finite time. Thus, the problem "given a financial system, find a clearing recovery rate vector" is not well-posed and the best we can hope for is an algorithm that computes a recovery rate vector that is in some sense *approximately* clearing. More in detail, we need to relax our problem such that there always exist "approximate" solutions that are *polynomial* in length: otherwise, any algorithm would take super-polynomial time in the worst case even to output an approximate solution.

There are many ways to relax the definition of clearing recovery rate vectors to receive a concept of an approximate solution. The approach we will use in this paper is by relaxing the function ρ from the proof of Theorem 3.3. For $x \in \mathbb{R}$ let $[x] := \min(1, \max(0, x))$. For $\varepsilon \geq 0$ write $y = x \pm \varepsilon$ to mean that $|x - y| \leq \varepsilon$ if x and y are scalars and $\|x - y\| \leq \varepsilon$ if x and y are vectors, where $\|\cdot\|$ is the supremum norm. We also use the notation "$\pm\varepsilon$" in compound expressions such as $[x \pm \varepsilon]$ to indicate a range of possible values. This notation formally corresponds to interval arithmetic. For $\varepsilon \geq 0$ and $i \in N$ let

$$\rho_i^\varepsilon(r) : [0,1]^N \to 2^{[0,1]}$$
$$\rho_i^\varepsilon(r) := \begin{cases} [\frac{a_i(r)}{l_i(r)} \pm \varepsilon] & \text{if } l_i(r) > 0 \\ [0,1] & \text{if } l_i(r) = 0 \end{cases}$$

and let $\rho^\varepsilon : [0,1]^N \to [0,1]^N$ be defined accordingly.

Definition 4.1 (Approximately Clearing Recovery Rate Vector)**.** Fix a financial system without default costs and let $\varepsilon \geq 0$. A recovery rate vector r is called *ε-approximately clearing* or an *ε-solution* if it is a fixed point of the set-valued function ρ^ε, i.e., if $r \in \rho^\varepsilon(r)$. For clarity, we refer to (not approximate) solutions as *exact solutions*.

The following proposition gives a useful equivalent definition of ε-solutions.

Proposition 4.2 (Alternative Definition of Approximate Solutions)**.** *r is an ε-solution iff for all i we have one of the following cases, where b) and c) are non-exclusive:*

a) $l_i(r) = 0$

b) $l_i(r) > 0$, $\frac{a_i(r)}{l_i(r)} \geq 1 - \varepsilon$, *and* $r_i = 1$

c) $l_i(r) > 0$, $\frac{a_i(r)}{l_i(r)} \leq 1 + \varepsilon$, *and* $r_i = \frac{a_i(r)}{l_i(r)} \pm \varepsilon$

In particular, if $\frac{a_i(r)}{l_i(r)} \geq 1 + \varepsilon$*, then we must have* $r_i = 1$ *and if* $\frac{a_i(r)}{l_i(r)} < 1 - \varepsilon$*, then we must have* $r_i < 1$*. Further, if r is an ε-solution and $l_i(r) > 0$, then* $r_i = [\frac{a_i(r)}{l_i(r)}] \pm \varepsilon$*, but the converse does not necessarily hold.*

The proof of the proposition is by simple algebra and is omitted. Our definition of an approximate solution has many desirable properties from an economic and technical point of view. We provide a discussion in Appendix B.

It is easy to see that for any $\varepsilon > 0$, there always exists an ε-solution of finite length. To guarantee that there is also an ε-solution of *polynomial* length, we make an additional assumption that we call *nondegeneracy*.[2] We can then state our search problem.

Definition 4.3 (Nondegenerate Financial System)**.** A financial system without default costs $X = (N, e, c)$ is called *nondegenerate* if each bank that writes a CDS

a) also writes a debt contract or

b) has strictly positive external assets.

Definition 4.4 (ε-FindClearing Problem)**.** For any parameter $\varepsilon > 0$, ε-FindClearing is the following total search problem: given a nondegenerate financial system without default costs, find an ε-solution.

The following lemma establishes that under the assumption of nondegeneracy, sufficiently "short" approximate solutions always exist in the vicinity of exact solutions, thus making ε-FindClearing a well-posed search problem. The converse is not in general true: there can be additional approximate solutions that are not close to any exact solution. While this is unfortunate, it appears to be unavoidable for an approximate solution concept; for example, the well established concept of approximate Nash equilibria also has this property.

Lemma 4.5. *If $X = (N, e, c)$ is a nondegenerate financial system without default costs and $\varepsilon > 0$, then there exists an ε-solution of length polynomial in the length of X and the length of ε.*

Proof Outline (full proof in Appendix C). We define a function F such that any ε-approximate fixed point of F gives rise to an ε-solution of X and we prove that F has a polynomial Lipschitz constant. By Brouwer's fixed point theorem, F has a fixed point and by rounding this fixed point to a grid of polynomial step size, we receive an ε-solution. ☐

Theorem 4.6. *For any $\varepsilon > 0$, the problem ε-FindClearing is in PPAD.*

Proof Outline (full proof in Appendix C). We use the function F from the proof of the previous lemma and the fact that the problem of finding an approximate fixed point of a Lipschitz continuous function is in PPAD. ☐

5 FindClearing is PPAD-hard

Our main contribution in this paper is the proof that FindClearing is PPAD-hard, and thus PPAD-complete for a sufficiently small constant ε.

Theorem 5.1. *There exists an $\varepsilon > 0$ such that the ε-FindClearing problem is PPAD-hard.*

The theorem immediately implies:

Corollary 5.2. *There is no polynomial-time approximation scheme that computes an ε-solution for a given financial system without default costs and given ε, unless P = PPAD.*

Towards a proof of the theorem, we proceed in two steps: we first introduce a variant of Rubinstein's (2015) generalized circuit framework and we show that the problem of finding an approximate solution of a generalized circuit in this framework is still well-posed and PPAD-complete (Section 5.1). We then reduce this problem to FindClearing (Section 5.2).

[2] It is an open question whether or not ε-solutions of polynomial length are still guaranteed to exist when this assumption is not made.

5.1 Generalized Circuits

A generalized circuit consists of a collection of interconnected arithmetic or Boolean gates. In contrast to regular arithmetic or Boolean circuits, generalized circuits may contain cycles, making the problem of finding a solution (or stable state) of the circuit a non-trivial fixed point problem. Rubinstein (2015) introduced a framework for generalized circuits that is already well-suited for our purposes. To make our reduction to financial systems as simple as possible, we use a reduced set of gates.

Definition 5.3 (Generalized Circuit and Approximate Solution). A *generalized circuit* is a collection of *nodes* and *gates*, where each node is labeled *input* of any number of gates (including zero) and *output* of at most one gate. Inputs to the same gate are distinguishable from each other. Each gate has one of the following types:

- For each $\zeta \in [0,1]$ the *constant gate* C_ζ with no inputs and one output.
- Arithmetic gates: *addition* and *subtraction* gates, denoted C_+ and C_-, with two inputs and one output; for each $\zeta > 0$ the *scale by* ζ gate $C_{\times\zeta}$ with one input and one output.
- For each $\zeta \in (0,1)$ the *compare to* ζ *gate* $C_{>\zeta}$ with one input and one output.
- Boolean gates: $C_\neg$ with one input and one output and $C_\vee$ with two inputs and one output.

The *length* of a generalized circuit is given by the number of nodes, the size of the mapping from nodes to inputs and outputs of gates, and the length of any ζ values involved.

If $\varepsilon \geq 0$ and C is a generalized circuit, then an ε*-approximate solution* (or ε*-solution*) to C is a mapping that assigns to each node v of C a value $x[v] \in [0,1]$ such that at any gate of type g with inputs $a_1, \dots, a_k$ and output v the respective condition from Figure 2 holds.

Definition 5.4 (ε-GCircuit Problem). For any parameter $\varepsilon > 0$, ε-GCircuit is the following total search problem: given a generalized circuit, find an ε-solution.

Note how the comparison gadget $C_{>\zeta}$ is *brittle*: its value is arbitrary if $x[a_1]$ is close to ζ. This property is crucial for our second step of describing generalized circuits via financial systems because the function $\frac{a_i}{l_i}$ that ultimately defines an approximate solution is always continuous while a non-brittle comparison gadget, yielding low values for $x[a_1] < \zeta$ and high values for $x[a_1] \geq \zeta$, would correspond to a discontinuous function. We further use *approximate Boolean values* $0 \pm \varepsilon$ and $1 \pm \varepsilon$ instead of exact Boolean values 0 and 1 since the latter are not attainable if there can be ε errors at each bank. Note how chains of Boolean gadgets do not accumulate errors, but chains of arithmetic gadgets do.

It is well accepted in the literature that GCircuit is well-posed and in PPAD. We provide the following simple lemma for our variant of GCircuit for completeness. The proof can be found in Appendix D.

Lemma 5.5.

1. *If C is a generalized circuit and $\varepsilon > 0$, then there exists an ε-solution for C of length polynomial in the length of C and the length of ε.*
2. *For any $\varepsilon > 0$, the ε-*GCircuit *problem is in PPAD.*

PPAD-hardness of the GCircuit problem for constant ε follows by reduction from Rubinstein's (2015) variant:

Figure 2 Conditions to hold at the different gates in an ε-solution of a generalized circuit

$$\begin{array}{lll}
g = C_\zeta & \Rightarrow & x[v] = \zeta \pm \varepsilon \\
g = C_+ & \Rightarrow & x[v] = [x[a_1] + x[a_2]] \pm \varepsilon \\
g = C_- & \Rightarrow & x[v] = [x[a_1] - x[a_2]] \pm \varepsilon \\
g = C_{\times\zeta} & \Rightarrow & x[v] = [\zeta x[a_1]] \pm (1+\zeta)\,\varepsilon \\
g = C_{>\zeta} & \Rightarrow & x[a_1] \le \zeta - \varepsilon \Rightarrow x[v] = 0 \pm \varepsilon \\
 & & x[a_1] \ge \zeta + \varepsilon \Rightarrow x[v] = 1 \pm \varepsilon \\
g = C_\neg & \Rightarrow & x[a_1] = 0 \pm \varepsilon \Rightarrow x[v] = 1 \pm \varepsilon \\
 & & x[a_1] = 1 \pm \varepsilon \Rightarrow x[v] = 0 \pm \varepsilon \\
g = C_\vee & \Rightarrow & x[a_1] = 0 \pm \varepsilon \text{ and } x[a_2] = 0 \pm \varepsilon \Rightarrow x[v] = 0 \pm \varepsilon \\
 & & x[a_1] = 1 \pm \varepsilon \text{ or } \quad x[a_2] = 1 \pm \varepsilon \Rightarrow x[v] = 1 \pm \varepsilon
\end{array}$$

Theorem 5.6. *There exists an $\varepsilon > 0$ such that the ε-GCircuit problem is PPAD-hard.*

Proof. Rubinstein (2015) proved that the following variant of the ε-GCircuit problem is PPAD-hard for some ε:

1. Scaling is only allowed[3] by values $\zeta \le 1$ and has error $\pm\varepsilon$ instead of $\pm(1+\zeta)\varepsilon$.
2. There are two additional, redundant gates: $C_=$ is a gate that (approximately) copies its input and $C_\wedge$ implements an approximate AND operator.
3. The comparison gate compares two inputs rather than compare one input to a constant.

For the first point, note that if $\zeta \le 1$, then our $C_{\times\zeta}$ gate has error $(1+\zeta)\varepsilon \le 2\varepsilon$ and thus we can achieve Rubinstein's error bound by considering an $\frac{\varepsilon}{2}$-solution instead. The second point does not make the problem any harder because we can express $C_=$ as $C_{\times 1}$ and $C_\wedge$ via the identity $x \wedge y = \neg(\neg x \vee \neg y)$.

Towards the third point, we show how to emulate the behavior of a binary comparison gate. Let a_1 and a_2 be the inputs and v the output of the would-be binary comparison gate. The expected behavior is that $x[v] = 0 \pm \varepsilon$ if $x[a_1] \le x[a_2] - \varepsilon$ and $x[v] = 1 \pm \varepsilon$ if $x[a_1] \ge x[a_2] + \varepsilon$.

We rewrite the expression $x[a_1] < x[a_2]$ to use only comparison to a constant in a way that is robust against ε errors and cut-off at 0 and 1: construct, by combining the appropriate gates, a sub-circuit corresponding to the expression $(\frac{1}{2} + (a_1 - a_2)) - (a_2 - a_1)$ and call the output node of that circuit u. If $\varepsilon' > 0$ and $x[\cdot]$ is an ε'-solution, then $x[u] = \tilde{u} \pm 4\varepsilon'$ where

$$\tilde{u} \;=\; \left[\left[\frac{1}{2} + [x[a_1] - x[a_2]]\right] - [x[a_2] - x[a_1]]\right] \;=\; \left[\frac{1}{2} + x[a_1] - x[a_2]\right].$$

Note that $x[a_1] < x[a_2] \Leftrightarrow \tilde{u} < \frac{1}{2}$. Add a $C_{>\frac{1}{2}}$ gate with input u and output v.

[3] This assumption can be found in the full version of the paper (Rubinstein, 2016).

Now assume WLOG that $\varepsilon \leq \frac{1}{2}$, let $\varepsilon' = \frac{\varepsilon}{5}$, and let $x[\cdot]$ be an ε'-solution. Then

$$\begin{aligned} x[a_1] \leq x[a_2] - \varepsilon \quad \Rightarrow \quad & \tilde{u} \leq \frac{1}{2} - \varepsilon = \frac{1}{2} - 4\varepsilon' - \varepsilon' \\ \Rightarrow \quad & x[u] \leq \frac{1}{2} - \varepsilon' \\ \Rightarrow \quad & x[v] \leq \varepsilon' \leq \varepsilon \end{aligned}$$

Analogously $x[a_1] \geq x[a_2] + \varepsilon \Rightarrow x[v] \geq 1 - \varepsilon$.

Altogether, we can construct from any circuit C in Rubinstein's (2015) framework a circuit C' in our reduced framework such that the $\frac{\varepsilon}{5}$-solutions of C' are ε-solutions of C. This concludes the proof. □

Remark 5.7 (Irrelevance of ε Multiples). It is clear that one may replace any individual occurrence of ε in Figure 2 by a multiple $\lambda\varepsilon$ where $\lambda > 0$ is any constant while preserving the statement of the theorem. This is important because it will allow us to construct our financial system gadgets corresponding to gates without having to worry about the exact multiples of ε.

5.2 Reduction from Generalized Circuits to Financial Systems

We now reduce the GCircuit problem to the FindClearing problem. To do so, we construct *financial system gadgets*, i.e., fragments of financial systems where the recovery rate of an *output bank* is given (approximately) by a function of certain *input banks*.

Definition 5.8 (Financial System Gadget). A *financial system gadget G* is a polynomial-time computable function mapping a financial system without default costs $X = (N, e, c)$ to a new financial system $X' = (N', e', c')$ in the following way:

- Given are X, a set of *input banks* $a_1, \dots, a_k \in N$ where k depends on the gadget and an *output bank* $v \in N$ such that v has no assets or liabilities in X, i.e., $e_i = c^k_{i,j} = c^k_{j,i} = 0$ for all $j \in N$ and $k \in N \cup \{\emptyset\}$.
- X' consists of X together with new banks and contracts.
- For any ε-solution r' of X', the restriction $r := r'|_N$ is an ε-solution for X.
- For any ε-solution r of X, there is an ε-solution r' of X' such that $r_i = r'_i$ for all $i \in N \setminus \{v\}$.

We usually label input banks a and b instead of a_1 and a_2 for the sake of readability.

We will now describe our gadgets: addition gadgets, scaling and comparison gadgets, and Boolean gadgets. Some of the gadgets, shown in Figure 3–6, are *fundamental* while the others are defined as combinations of the fundamental ones. Our gadgets add assets and liabilities to the output bank and CDS references to the input banks. This ensures that gadgets only restrict the recovery rate of the output bank based on the recovery rates of the input banks, but not vice versa, and gadgets applied to different output banks do not conflict. In a final step, we iteratively apply our gadgets starting from a financial system with no contracts to receive a financial system that corresponds to a given generalized circuit. Keep in mind that by Remark 5.7, our gadgets only need to be accurate up to a constant multiple of ε, not necessarily up to ε. All gadgets lead to nondegenerate financial systems.

Figure 3 Constant Gadget: extension of an existing financial system with output bank v by new banks s, t and contracts such that $r_v = \zeta$.

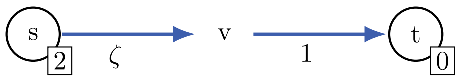
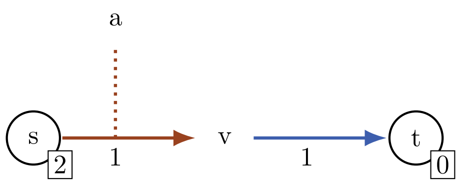


Figure 4 Inverter Gadget: extension of an existing financial system with input bank a and output bank v by new banks s, t and contracts that translate $1 - r_a$.



5.2.1 Addition Gadgets

The simplest gadget establishes a fixed recovery rate at the output bank:

Lemma 5.9 (Constant Gadget)**.** *Let $\zeta \in [0, 1]$. There is a financial system gadget with no input banks and output bank v such that if r is an ε-solution, then $r_v = \zeta \pm \varepsilon$.*

Proof. Consider the gadget in Figure 3. We have $\frac{a_s(r)}{l_s(r)} \geq 2 \geq 1 + \varepsilon$, so $r_s = 1$. Thus, s pays in full and $a_v(r) = \zeta$ and $l_v(r) = 1 \geq a_v(r)$, so in an ε-solution $r_v = \frac{a_v(r)}{l_v(r)} \pm \varepsilon = \zeta \pm \varepsilon$. $\square$

An important building block for the following constructions is a gadget that "inverts" the recovery rate of a bank.

Lemma 5.10 (Inverter Gadget)**.** *There is a financial system gadget with input bank a and output bank v such that if r is an ε-solution, then $r_v = 1 - r_a \pm \varepsilon$.*

Proof. Consider the gadget in Figure 4. Since $l_v(r) = 1$ we have in any ε-solution that $r_v = a_v(r) \pm \varepsilon$ and $a_v(r) = 1 - r_a$. $\square$

We can now define the sum and difference gadgets:

Lemma 5.11 (Sum Gadget)**.** *There is a financial system gadget with input banks a and b and output bank v such that if r is an ε-solution, then $r_v = [r_a + r_b] \pm 3\varepsilon$.*

Figure 5 Sum Gadget: extension of an existing financial system with input banks a and b and output bank v by new banks s, t and contracts that translate $r_a + r_b$.

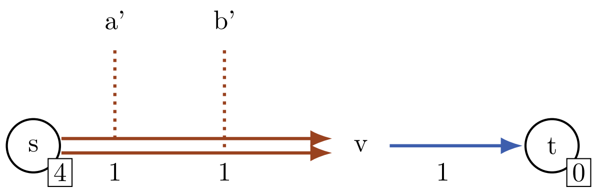
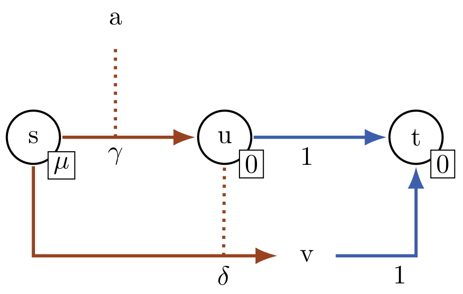


Figure 6 Amplifier Gadget: extension of an existing financial system with input bank a by new banks s, t, u, v and contracts that translate the function f from Lemma 5.13. Let $\mu = 2(\gamma + \delta)$.



Proof. Apply inverter gadgets (Lemma 5.10) to both a and b and call the output banks a' and b', respectively. Now consider the gadget in Figure 5. We have

$$\begin{aligned} r_v &= [1 - r_{a'} + 1 - r_{b'}] \pm \varepsilon \\ &= [r_a + r_b \pm 2\varepsilon] \pm \varepsilon \\ &= [r_a + r_b] \pm 3\varepsilon. \end{aligned}$$ ☐

Lemma 5.12 (Difference Gadget). *There is a financial system gadget with input banks a and b and output bank v such that if r is an ε-solution, then $r_v = [r_a - r_b] \pm 3\varepsilon$.*

Proof. Apply an inverter gadget (Lemma 5.10) to a and call the output bank a'. Apply the gadget in Figure 5 to a' and $b' := b$ and call the output bank u. From the proof of the previous lemma we know that

$$r_u = [1 - r_a + r_b] \pm 2\varepsilon$$

where the error is by one ε lower because we used one inverter gadget less. Now apply an inverter to u and call the output bank v. To show that r_v is as desired, we distinguish two cases:

- If $r_a \leq r_b$, then $1 - r_a + r_b \geq 1$, so $r_u \geq 1 - 2\varepsilon$ and thus $r_v = 1 - r_u \pm \varepsilon \leq 3\varepsilon$ as required.
- If $r_a \geq r_b$, then $1 - r_a + r_b \leq 1$, so $r_u = 1 - r_a + r_b \pm 2\varepsilon$ and thus $r_v = 1 - r_u \pm \varepsilon = r_b - r_a \pm 3\varepsilon$ as required. ☐

5.2.2 Scaling and Comparison

Towards the scaling and comparison gadgets, we introduce a versatile tool that can be used to re-scale and shift recovery rates.

Lemma 5.13 (Amplifier Gadget). *Let K and L be real numbers such that $K < L$, $K < 1$, and $L > 0$. Note that $K \leq 0$ and $L \geq 1$ are allowed. Let*

$$\begin{aligned} f &: [0,1] \to [0,1] \\ f(r_a) &:= \left[\frac{1}{L-K} r_a - \frac{K}{L-K} \right]. \end{aligned}$$

f connects the points $(-\infty, 0)$, $(K, 0)$, $(L, 1)$, and $(\infty, 1)$ via straight lines.

There is a financial system gadget with input bank a and output bank v such that if r is an ε-solution, then $r_v = f(r_a) \pm (\delta + 1)\varepsilon$ where $\delta = \frac{1-K}{L-K}$. The construction can be performed in time polynomial in the lengths of L and K.

Proof. Consider the gadget in Figure 6 with

$$\gamma := \frac{1}{1-K}$$
$$\delta := \frac{1-K}{L-K}.$$

Let r be an ε-solution. We show that $r_v = f(r_a) \pm (\delta + 1)\varepsilon$. We have

$$r_u = [\gamma(1 - r_a)] \pm \varepsilon$$
$$r_v = [\delta(1 - r_u)] \pm \varepsilon.$$

By replacing the first relation into the second one, we receive

$$\begin{aligned} r_v &\in [\delta\,(1 - ([\gamma(1 - r_a)] \pm \varepsilon))] \pm \varepsilon \\ &\subseteq [\delta\,(1 - [\gamma(1 - r_a)])] \pm (\delta + 1)\,\varepsilon \\ &= [\delta\,(1 - (\gamma(1 - r_a)))] \pm (\delta + 1)\,\varepsilon \\ &= [\delta - \delta\gamma + \delta\gamma r_a] \pm (\delta + 1)\,\varepsilon \;=\; \left[-\frac{K}{L-K} + \frac{1}{L-K} r_a\right] \pm (\delta + 1)\,\varepsilon \end{aligned}$$

where the third line is because $[\delta(1 - z)] = [\delta(1 - [z])]$ for any $z \in [0, \infty)$ and the last line is by simple algebra. Thus, r_v is as desired. □

We receive a scaling gadget by choosing $K = 0$:

Corollary 5.14 (Scale by Constant Gadget)**.** *Let $\zeta > 0$. There is a financial system gadget with input bank a and output bank v such that if r is an ε-solution, then $r_v = [\zeta r_a] \pm (1+\zeta)\varepsilon$. The construction can be performed in time polynomial in the length of ζ.*

Proof. Use an amplifier gadget (Lemma 5.13) with $K = 0$ and $L = \frac{1}{\zeta}$. Then $f(r_a) = [\zeta r_a]$ and $\delta = \zeta$. □

We receive a gadget that acts like the brittle comparison gate $C_{>\zeta}$ by choosing K and L closely together around a central point ζ. The gadget is less "brittle" the closer K and L are together, but this also increases the value δ and thus the output error of the gadget. To compensate for this, we first introduce a gadget that converts a wide range of values to approximate Boolean values with threshold 3ε.

Corollary 5.15 (Reset Gadget)**.** *There is a financial system gadget with input bank a and output bank v such that if r is an ε-solution, then if $r_a \leq \frac{1}{4}$, then $r_v = 0 \pm 3\varepsilon$ and if $r_a \geq \frac{3}{4}$, then $r_v = 1 \pm 3\varepsilon$.*

Proof. Apply the amplifier gadget (Lemma 5.13) with $K = \frac{1}{4}$ and $L = \frac{3}{4}$. We have $\delta + 1 = \frac{5}{2} < 3$. □

Corollary 5.16 (Brittle Comparison to Constant Gadget)**.** *Let $\zeta \in [0, 1]$. There is a financial system gadget with input bank a and output bank v such that if $\varepsilon \leq 1/18$ and r is an ε-solution, then if $r_a \leq \zeta - 3\varepsilon$, then $r_v = 0 \pm 3\varepsilon$ and if $r_a \geq \zeta + 3\varepsilon$, then $r_v = 1 \pm 3\varepsilon$. The construction can be performed in time polynomial in the length of ζ.*

Proof. We apply two constructions involving the amplifier gadget (Lemma 5.13): first we apply an amplifier to a as an input bank with $K := \zeta - 3\varepsilon$ and $L := \zeta + 3\varepsilon$. Call the output bank u. We have $\delta = \frac{1-K}{L-K} = \frac{1-\zeta+3\varepsilon}{6\varepsilon} \leq \frac{1+3\varepsilon}{6\varepsilon} = \frac{1}{6\varepsilon} + \frac{1}{2}$. So this gadget has output error $(\delta+1)\varepsilon \leq \frac{1}{6} + \frac{1}{2}\varepsilon + \varepsilon \leq \frac{1}{4}$. Thus, if $r_a \leq K$, then $r_u \leq \frac{1}{4}$ and if $r_a \geq L$, then $r_u \geq \frac{3}{4}$. Now apply a reset gadget (Corollary 5.15) to u as the input bank to receive the desired lower output error of 3ε. □

5.2.3 Boolean Gadgets

We can re-use the addition gadgets from above to build Boolean gadgets, translating OR into "+" and NOT into "$1 - x$" (inversion). We use the reset gadget to prevent errors from propagating.

Lemma 5.17 (Boolean Gadgets)**.** *There are financial system gadgets with input banks a and b and output bank v such that if $\varepsilon \leq 1/36$ and r is an ε-solution, then*

i) (OR) If $r_a = 0 \pm 3\varepsilon$ and $r_b = 0 \pm 3\varepsilon$, then $r_v = 0 \pm 3\varepsilon$.
If $r_a = 1 \pm 3\varepsilon$ or $r_b = 1 \pm 3\varepsilon$, then $r_v = 1 \pm 3\varepsilon$.

ii) (NOT) If $r_a = 0 \pm 3\varepsilon$, then $r_v = 1 \pm 3\varepsilon$.
If $r_a = 1 \pm 3\varepsilon$, then $r_a = 0 \pm 3\varepsilon$.

Proof. i) Apply a sum gadget (Lemma 5.11) to a and b and call the output bank u. Now apply a reset gadget (Corollary 5.15) to u and call the output bank v. We know that $r_u = [r_a + r_b] \pm 3\varepsilon$. If $r_a \geq 1 - 3\varepsilon$ or $r_b \geq 1 - 3\varepsilon$, then $r_u \geq 1 - 6\varepsilon \geq \frac{3}{4}$, so $r_v \geq 1 - 3\varepsilon$. If $r_a, r_b \leq 3\varepsilon$, then $r_u \leq 9\varepsilon \leq \frac{1}{4}$, so $r_v \leq 3\varepsilon$.

ii) Apply similarly an inverter gadget (Lemma 5.10) and then a reset gadget. One easily checks that the construction behaves as desired. □

5.2.4 Completing the PPAD-hardness Proof

We combine our gadgets to model generalized circuits, thus reducing GCircuit to FindClearing and proving PPAD-hardness of FindClearing:

Proof of Theorem 5.1. Let $\varepsilon > 0$ be arbitrary. We reduce ε-GCircuit to ε'-FindClearing where $\varepsilon' := \frac{\varepsilon}{3}$. Let be given a generalized circuit C with n nodes. Construct a financial system via the following algorithm.

- Start with a system X^0 consisting of n banks that have no assets or liabilities. Identify these n banks with the nodes of C.
- Consider the gates of C in any order. For each $k = 1, \dots, n$ do the following:
  - Consider the k-th gate of C. Let g be the type, $a_1, \dots, a_k$ the inputs, and v the output of this gate.
  - Apply the gadget from above corresponding to g to X^{k-1} with input banks $a_1, \dots, a_k$ and output bank v. Call the resulting financial system X^k.
- Let $X := X^n$.

For $k = 0, \dots, n$ let C^k be C restricted to the first k gates. We show by induction on k that the ε'-solutions of X^k correspond to ε-solutions of C^k. For $k = 0$, the statement is clear. For $k > 0$, and assuming the statement for $k - 1$, it follows from the fact that the bank corresponding to the output of the k-th gate has no assets or liabilities in X^{k-1} and then from the definition of a financial system gadget and our above lemmas. By definition of the gadgets, each X^k, and thus X, is nondegenerate. □

Remark 5.18. The intermediate systems X^k in the above construction may violate our assumption that any bank that is a reference entity in a CDS must be a writer of some debt contract (cf. Section 3). This happens when gadgets refer to a reference entity that is an output bank of another gadget that has not yet been executed. We can circumvent this problem by temporarily replacing such banks by a financial sub-system that fulfills all our assumptions and in which one of the banks can attain any recovery rate in some solution.[4] Alternatively, one checks that not having the assumption does not lead to any problems in the proof.

6 Origin of the Computational Complexity

Given the results from the previous section, one may wonder why exactly the computational complexity arises in financial networks with CDSs and why it did not arise in debt-only systems. Understanding this is important to devise policies that aim to reduce complexity in the financial system in the future. In general financial systems with credit default swaps, many possible origins of computational complexity come to mind:

a) Banks' liabilities may form a cycle, creating "feedback loops" where banks are highly sensitive to changes in the assets of the other banks. These cycles may even be interlinked when banks have liabilities to more than one creditor, where the principle of proportionality leads to a strong coupling between a cycle and the rest of the system.

b) In the definition of approximately clearing recovery rates, there are two sources of non-linearities:

   - In CDSs, having both counterparty risk (i.e., the dependence of banks on the recovery rates of their debtors) and fundamental risk (i.e., the dependence of CDS holders and writers on the recovery rates of reference entities) introduces quadratic terms into the definition of the assets a_i of a bank.
   - The liabilities of CDS writers depend on the recovery rates of other banks, which introduces terms of form $\frac{1}{1-r_k}$ into the function $\frac{a_i}{l_i}$.

c) The complexity could come from determining which banks are in default while computing the values of recovery rates could be easy.

We show that of these points, a) and b) cannot alone be the origin of the complexity while we answer the last point in the affirmative. We first notice that all our gadgets, and thus the financial systems we use to show PPAD-hardness, share three properties that make them particularly simple financial systems:

1. *Acyclic Liabilities:* The graph of writer-holder relationships of contracts is acyclic.
2. *Outside Insurers:* CDS writers are highly capitalized banks: their external assets are significantly (by factor $2 \geq 1+\varepsilon$, for any relevant ε) higher than the total notional of their contracts written and thus, they have recovery rate 1 in any ε-solution.

[4] Such a financial system is described in (Schuldenzucker et al., 2016, Figure 3, $\delta = \gamma = 1$).

3. *No Counterparty Risk:* Contracts are either written by a highly capitalized source bank s or held by a sink bank t with zero liabilities.[5]

The first property implies that cycles of liabilities cannot be the reason for PPAD-hardness. The second property implies that we never have both counterparty and fundamental risk at the same time in our construction, so banks' assets never contain quadratic terms. It further implies that banks that are at the risk of defaulting do not write CDSs and thus their liabilities are constant, so $\frac{a_i}{l_i}$ never contains terms of form $\frac{1}{1-r_k}$. The third property implies that proportionality is not relevant.

To answer the remaining question if it is easier to compute the set of defaulting banks than to compute approximate recovery rates, we introduce the notion of a *default set* as the set of banks that do not pay their liabilities in full at a given recovery rate vector:

Definition 6.1 (Default Set)**.** If r is a recovery rate vector, define the *default set* of r as

$$D(r) := \{i \in N \mid r_i < 1\}.$$

The following theorem and corollary show that in the setting of our construction, it is easy to determine recovery rates once the default set is known, so hardness of the problem must stem from having to compute default sets. We receive as an additional result that our construction in fact has an *exact* solution of polynomial length (but finding it is PPAD-hard).

Theorem 6.2. *Given $\varepsilon \geq 0$, a financial system without default costs with outside insurers, and the default set of any ε-solution, one can compute an ε-solution in time polynomial in the length of the financial system.*

Proof. Let $\varepsilon \geq 0$. Let $X = (N, e, c)$ be a financial system with outside insurers (with respect to ε) and let $M \subseteq N$ be the set of banks i for which $e_i < (1+\varepsilon)(\sum_j c^{\emptyset}_{i,j} + \sum_{j,k} c^{k}_{i,j})$. By assumption, banks in M do not write CDSs, so $l_i(r) =: l_i$ is a constant for all $i \in M$. Assume WLOG $l_i > 0$ for all $i \in M$. CDSs held by banks in M are further only written by banks outside M, so that for the assets of a bank $i \in M$ we have in any ε-solution r that

$$a_i(r) = e_i + \sum_{j \in N \setminus M} c^{\emptyset}_{j,i} + \sum_{j \in M} c^{\emptyset}_{j,i} r_j + \sum_{\substack{j \in N \setminus M \\ k \in M}} c^{k}_{j,i}(1 - r_k)$$

is a linear term in r. The expression does not contain any r_j with $j \in N \setminus M$ because we have $r_j = 1$ for these j.

[5] To ease presentation, we assume in the following that all source banks s and all sink banks t, respectively, are the same. This does not change the solutions for the other banks.

Let now $D \subseteq N$ and consider the following linear program with variables $r_M := (r_i)_{i \in M}$:

$$\begin{aligned}
&& \min \tilde{\varepsilon} \text{ s.t.} && (2)\\
&& \tilde{\varepsilon} \geq 0 &&\\
&\text{For all } i \in M: & 0 \leq r_i \leq 1 &&\\
&\text{For all } i \in M \setminus D: & &&\\
&& \frac{a_i(r_M)}{l_i} \geq 1 - \tilde{\varepsilon} &&\\
&& r_i = 1 &&\\
&\text{For all } i \in M \cap D: & &&\\
&& \frac{a_i(r_M)}{l_i} \leq 1 + \tilde{\varepsilon} &&\\
&& r_i = \frac{a_i(r_M)}{l_i} \pm \tilde{\varepsilon} &&
\end{aligned}$$

The two cases $i \in M \setminus D$ and $i \in M \cap D$ correspond to parts b) and c) of Proposition 4.2, respectively. One checks that any ε-solution with default set D is a feasible solution of the LP with objective value $\tilde{\varepsilon} \leq \varepsilon$. Vice versa, any such solution to the LP gives rise to an ε-solution of the financial system by setting the recovery rates of banks outside M to 1. The default set of this ε-solution may not be D, though.

Now, if D if the default set of some ε-solution, then the LP for D must be feasible with optimal value $\leq \varepsilon$. Since the LP has polynomial size in the financial system, we can compute in polynomial time an optimal solution r to the LP via the ellipsoid method. By optimality, r must have value $\leq \varepsilon$ and thus be an ε-solution. $\square$

Corollary 6.3. *There exists an $\varepsilon > 0$ such that the following problem is PPAD-complete: given a nondegenerate financial system without default costs, find a set of banks that is the default set of some ε-solution. The problem remains PPAD-complete when restricted to financial systems with acyclic liabilities, outside insurers, and no counterparty risk.*

Proof. The statement follows immediately from Theorem 6.2 and the fact that our construction in the PPAD-hardness proof had the mentioned properties. $\square$

Corollary 6.4. *Any financial system without default costs with outside insurers (with respect to $\varepsilon = 0$) has an exact solution of polynomial length; finding one is PPAD-hard.*

Proof. The existence statement follows from Theorem 6.2 for $\varepsilon = 0$ when applied to the default set of any exact solution. PPAD-hardness is clear. $\square$

Coming back to our discussion on the origin of the complexity, we highlight the unique property of CDSs that banks can have an "inverse relationship" or *short position* on each other: the holder of a CDS profits from the ill-being of the reference entity, which allows us to implement operations such as logical negation. This effect is only present when CDSs are held by banks in a *naked* fashion, i.e., without holding a corresponding debt contract from the reference entity. We have shown that even the simplest kinds of financial networks, if they contain naked CDSs, can lead to PPAD-hardness. Without naked CDSs however, it is hard to imagine how anything close to our construction could be carried out. We conjecture that financial networks in which naked CDSs are not allowed admit a polynomial-time algorithm, similar to debt-only networks. Due to non-linearities in the function $\frac{a_i}{l_i}$, developing such an algorithm is not straightforward and beyond the scope of this paper.[6]

7 Conclusion

In this paper, we have studied the problem of computing clearing payments in financial networks with debt and credit default swap (CDS) contracts and without default costs. We have shown that compared to debt-only networks, the addition of CDSs turns the clearing problem from being solvable exactly in polynomial time into an approximation problem that is PPAD-complete even when the desired approximation quality is kept constant. Consequently, no polynomial-time approximation scheme exists unless P=PPAD. We have shown that even though general financial networks with CDSs could exhibit high computational complexity for many different reasons, the clearing problem is already PPAD-complete in the special case with acyclic liabilities, outside insurers, and no counterparty risk. In fact, already determining which banks are in default and which are not is a PPAD-complete problem.

Future work should investigate if the clearing problem can be solved in polynomial time when naked CDS positions are not allowed. We conjecture that this is the case. Another important task for future research is to find algorithms for the general case that may not have polynomial worst-case running time, but are fast in practice. These algorithms could work by iterating over default sets in a systematic fashion. All algorithms that are not restricted to outside insurers must in addition be able to deal with non-linearities in the function $\frac{a_i}{l_i}$.

References

D. Acemoglu, A. Ozdaglar, and A. Tahbaz-Salehi. Systemic risk and stability in financial networks. *American Economic Review*, 105(2):564–608, Feb 2015.

F. Allen and D. Gale. Financial contagion. *Journal of political economy*, 108(1):1–33, 2000.

S. Arora, B. Barak, M. Brunnermeier, and R. Ge. Computational complexity and information asymmetry in financial products. In *ICS*, pages 49–65, 2010.

S. Battiston, M. Puliga, R. Kaushik, P. Tasca, and G. Caldarelli. Debtrank: Too central to fail? financial networks, the fed and systemic risk. *Scientific reports*, 2, 2012.

S. Battiston, J. D. Farmer, A. Flache, D. Garlaschelli, A. G. Haldane, H. Heesterbeek, C. Hommes, C. Jaeger, R. May, and M. Scheffer. Complexity theory and financial regulation. *Science*, 351 (6275):818–819, 2016. doi: 10.1126/science.aad0299.

M. Braverman and K. Pasricha. The computational hardness of pricing compound options. In *Proceedings of the 5th conference on Innovations in theoretical computer science*, pages 103–104. ACM, 2014. Working paper at `http://eccc.hpi-web.de/report/2014/013/`.

X. Chen, X. Deng, and S. h. Teng. Computing nash equilibria: Approximation and smoothed complexity. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 603–612, Oct 2006. doi: 10.1109/FOCS.2006.20.

C. Daskalakis. On the complexity of approximating a nash equilibrium. *ACM Transactions on Algorithms (TALG)*, 2013. Special Issue for SODA 2011, Invited.

C. Daskalakis, P. W. Goldberg, and C. H. Papadimitriou. The complexity of computing a nash equilibrium. *Electronic Colloquium on Computational Complexity (ECCC)*, (115), 2005.

[6] Of course, the restriction of the clearing problem to no naked CDSs and outside insurers would not contain any non-linearities, but it is also trivial: such financial networks are equivalent to debt-only networks. Cf. Appendix B and (Schuldenzucker et al., 2016) for a discussion of naked CDSs.

C. Daskalakis, P. W. Goldberg, and C. H. Papadimitriou. The complexity of computing a nash equilibrium. *Commun. ACM*, 52(2):89–97, feb 2009. ISSN 0001-0782. doi: 10.1145/1461928.1461951.

L. Eisenberg and T. H. Noe. Systemic risk in financial systems. *Management Science*, 47(2): 236–249, 2001.

M. Elliott, B. Golub, and M. O. Jackson. Financial networks and contagion. *American Economic Review*, 104(10):3115–53, 2014.

B. Hemenway and S. Khanna. Sensitivity and computational complexity in financial networks. Working Paper, Mar 2015. URL `http://arxiv.org/abs/1503.07676`.

D. Hu, J. L. Zhao, Z. Hua, and M. C. Wong. Network-based modeling and analysis of systemic risk in banking systems. *MIS Quarterly*, 36(4):1269–1291, 2012.

C. H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *Journal of Computer and System Sciences*, 48(3):498 – 532, 1994. ISSN 0022-0000. doi: http://dx.doi.org/10.1016/S0022-0000(05)80063-7.

L. Rogers and L. A. Veraart. Failure and rescue in an interbank network. *Management Science*, 59(4):882–898, 2013.

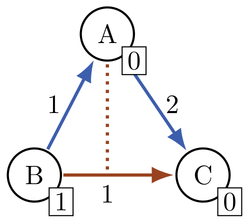
A. Rubinstein. Inapproximability of nash equilibrium. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, STOC '15, pages 409–418, Portland, Oregon, USA, 2015. ACM. ISBN 978-1-4503-3536-2.

A. Rubinstein. Inapproximability of nash equilibrium. Working Paper, 2016.

S. Schuldenzucker, S. Seuken, and S. Battiston. Clearing payments in financial networks with credit default swaps [extended abstract]. In *Proceedings of the 17th ACM Conference on Economics and Computation*, EC '16, Maastricht, The Netherlands, 2016. ACM. Current Working Paper: `http://www.ifi.uzh.ch/ce/publications/Clearing_CDSs.pdf`.

D. Zuckerman. Pseudorandom financial derivatives. In *Proceedings of the 12th ACM Conference on Electronic Commerce*, EC '11, pages 315–320, San Jose, California, USA, 2011. ACM. ISBN 978-1-4503-0261-6.

Figure 7 Financial System without default costs where the unique solution is irrational.



APPENDIX

A Irrational Solutions

Example A.1 (Irrational Solutions). Figure 7 shows a financial system the unique solution of which is irrational. To see this, note that by the contract structure r is clearing iff

$$r_A = \frac{1}{2} r_B, \qquad r_B = \frac{1}{2 - r_A},$$

and r_C is left unconstrained. One easily checks that the unique solution in $[0, 1]^2$ to this system of equations is given by

$$r_A = 2 - \sqrt{2}, \qquad r_B = 1 - \frac{1}{\sqrt{2}}.$$

B Properties of Approximate Solutions

Our definition of an approximate solution is well motivated from an economic point of view: assume that a bank A holds a debt contract of notional γ from bank B as well as a CDS on B from a highly capitalized bank C with the same notional. This contract pattern is called a *covered CDS* and it was the original use case CDSs were designed for: the CDS insures the debt contract. While nowadays, a large part of the CDSs are traded *naked* (i.e., they do not have this property), the covered case serves as a benchmark to which extent our solution concept is natural.

We describe the effect of ε errors in recovery rates on the three banks. If the insurer C is highly capitalized (its assets are greater than its liabilities by a factor $1 + \varepsilon$), then C never defaults ($r_C = 1$) and the assets of A are

$$\gamma r_B + \gamma(1 - r_B) r_C = \gamma.$$

That is, the covered CDS acts as a "full" insurance that eliminates A's dependence on B. This property is not affected by ε errors in the recovery rates of any bank. On the other hand, the writer C of the CDS might incur higher or lower liabilities due to errors in r_B, but this difference is bounded by $\varepsilon\gamma$. Finally, the recovery rate of B might be up to ε lower or higher than $\frac{a_B(r)}{l_B(r)}$. If it is lower, then B may keep up to $\varepsilon\gamma$ of its assets even though it is in default. If it is higher however, then B must make up to $\varepsilon\gamma$ in payments from money it does not have. This money would have to come from an external entity such as a government institution or the clearing mechanism itself. This is why clearing mechanisms should seek ε-solutions where ε is small compared to the inverse notionals in the system.

The following elementary properties serve as an indication that our definition of an approximate solution is also natural from a technical point of view. They are all easy to validate.

Proposition B.1 (Natural Properties of Approximate Solutions)**.** *Fix a financial system without default costs.*

1. *Any r is a 1-solution. r is a 0-solution iff it is an exact solution.*
2. *If $\varepsilon \leq \varepsilon'$, then any ε-solution is also an ε'-solution.*
3. *r is an ε-solution iff r is an ε'-solution for all $\varepsilon' > \varepsilon$.*
4. *Given r and ε, one can check in polynomial time if r is an ε-solution via interval arithmetic.*

C Proofs from Section 4

The following lemma lets us express ε-FindClearing as the problem of finding an approximate fixed point of a certain Lipschitz continuous function. Then the lemma and theorem follow using standard techniques.

Lemma C.1. *Given a nondegenerate financial system without default costs $X = (N, e, c)$ and $\varepsilon > 0$ define the function*

$$F : [0,\, 1+\varepsilon]^N \to [0,\, 1+\varepsilon]^N$$
$$F_i(s) := \begin{cases} \left[\frac{a_i([s])}{l_i([s])}\right]^{1+\varepsilon} & \text{if } l_i([s]) > 0 \\ 1+\varepsilon & \text{if } l_i([s]) = 0 \end{cases}$$

where $[x]^{1+\varepsilon} := \min(1+\varepsilon,\, \max(0,\, x))$ *and* $[s] := ([s_1],\, \dots,\, [s_n])$.
Then the following hold:

1. *F is Lipschitz continuous with a Lipschitz constant polynomial-time computable from X and ε.*
2. *If s is an ε-approximate fixed point of F, then $[s]$ is an ε-solution of X.*

Proof. **Part 1:** It is sufficient to show that each F_i has an appropriate Lipschitz constant. So let $i \in N$. By nondegeneracy, bank i must fall into one of three cases: it either writes no contracts at all, or writes a debt contract, or has positive external assets. If i writes no contracts, then F_i is constant $1 + \varepsilon$.

If i writes a debt contract, then $l_i([s]) > 0$ for all s, so

$$F(s) = \left[\frac{a_i([s])}{l_i([s])}\right]^{1+\varepsilon} = \left([\cdot]^{1+\varepsilon} \circ \frac{a_i}{l_i} \circ [\cdot]\right)(s).$$

The functions $[\cdot]^{1+\varepsilon}$ and $[\cdot]$ are Lipschitz with constant 1. For $\frac{a_i}{l_i}$, we find a bound on the partial derivatives. We have

$$\frac{\partial \frac{a_i}{l_i}}{\partial r_k} = \frac{\frac{\partial a_i}{\partial r_k} l_i - a_i \frac{\partial l_i}{\partial r_k}}{l_i^2}$$
$$= \frac{(l_{k,i} - \sum_j r_j c_{j,i}^k) \cdot l_i + a_i \cdot \sum_j c_{i,j}^k}{l_i^2}.$$

where the second line is easily seen by expanding a_i and l_i. The numerator is bounded from above in absolute value by

$$N_k^i := \left(c_{k,i}^{\emptyset} + \sum_j c_{k,i}^j\right) \cdot \left(\sum_j c_{i,j}^{\emptyset} + \sum_{j,l} c_{i,j}^l\right) + \left(e_i + \sum_j c_{j,i}^{\emptyset} + \sum_{j,l} c_{j,i}^l\right) \cdot \sum_j c_{j,i}^k$$

and the denominator is bounded from below by $D^i := (\sum_j c_{i,j}^{\emptyset})^2$. Thus, the partial derivative is bounded by $\frac{N_k^i}{D^i}$ and this bound is polynomial in X.

If i has positive external assets, then let $L_i := \{s \mid l_i([s]) > e_i\}$. For $s \notin L_i$, we have $F_i(s) = 1 + \varepsilon$ and further $F_i(s) \to 1 + \varepsilon$ as $l_i([s]) \to e_i$. On L_i, one receives a Lipschitz constant for the restriction of $[\frac{a_i([s])}{l_i([s])}]^{1+\varepsilon}$ to L_i by applying the same reasoning as above with $D_i := e_i^2$. Thus, F_i is the continuous union of two Lipschitz continuous functions and thus itself Lipschitz with the constant being the maximum of the two Lipschitz constants, namely $\max_k \frac{N_k^i}{D_i}$.

Part 2: Let s be an ε-approximate fixed point of F. Let $i \in N$ and let $\tilde{s}_i := \frac{a_i(r)}{l_i(r)} \in [0, \infty)$. We have $F_i(s) = [\tilde{s}_i]^{1+\varepsilon}$ and thus

$$\begin{aligned} s_i &= [\tilde{s}_i]^{1+\varepsilon} \pm \varepsilon \\ \Rightarrow \quad [s_i] &\in \left[[\tilde{s}_i]^{1+\varepsilon} \pm \varepsilon\right] = [\tilde{s}_i \pm \varepsilon] \end{aligned}$$

where the last equality is easily seen by case distinction on $\tilde{s}_i \geq 1+\varepsilon$ and $\tilde{s}_i < 1+\varepsilon$. Thus, $[s]$ is an ε-solution at i. □

Proof of Lemma 4.5. Let X and ε be given and consider the function F from Lemma C.1. Let K be the Lipschitz constant and recap that K is polynomial in X and ε. Since F is continuous on a compact domain, by Brouwer's fixed point theorem, it has an (exact) fixed point s. Let $\delta = \frac{\varepsilon}{K+1}$. Let s' be defined by $s'_i := \delta\lfloor\delta^{-1}s_i\rfloor$. That is, s'_i is s_i rounded to multiples of δ. s' has length $n \cdot L$ where L is the length of δ, and L is polynomial in the lengths of X and ε.[7] Further,

$$\begin{aligned} \|s' - F(s')\| &\leq \|s' - F(s)\| + \|F(s') - F(s)\| \\ &= \|s' - s\| + \|F(s') - F(s)\| \\ &\leq \delta + K\delta = (1+K)\delta = \varepsilon. \end{aligned}$$

Hence, s' is an ε-approximate fixed point of F and thus an ε-solution. □

Proof of Theorem 4.6. Proof by reduction to the PPAD-complete generic Brouwer problem (Daskalakis et al., 2009):

> Given an efficient algorithm for the evaluation of a function $F : [0,1]^m \to [0,1]^m$, a Lipschitz constant K for F, and an accuracy $\varepsilon > 0$, compute a point x such that $\|F(x) - x\| \leq \varepsilon$.

We apply the generic Brouwer problem to the function F from Lemma C.1. One checks that one may replace the domain $[0,1]$ by $[0, 1+\varepsilon]$ without changing the problem in any significant way (e.g., by scaling inputs and outputs of F by a factor $1+\varepsilon$ and replacing ε by $\frac{\varepsilon}{1+\varepsilon} \geq \frac{1}{2}\varepsilon$). Again by Lemma C.1, we know that the output of the Brouwer problem gives rise to an ε-solution for X. □

[7] We assume here that numbers are encoded as fractions of binary integers. Alternatively, one could choose δ to be the largest power of two $\leq \frac{\varepsilon}{K+1}$.

D Proofs from Section 5.1

Proof of Lemma 5.5. We show that the approximate solutions of a circuit correspond to the approximate fixed points of a certain Lipschitz continuous function. The statement of the lemma then follows like in the proofs of Lemma 4.5 and Theorem 4.6.

For given C and ε define *gate functions* $f_g : [0,1]^k \to [0,1]$, where $k \in \{0,1,2\}$, as follows:

$$\begin{aligned} f_{C_\zeta} &:= \zeta \\ f_{C_+}(a,b) &:= [a+b] \\ f_{C_-}(a,b) &:= [a-b] \\ f_{C_{\times\zeta}}(a) &:= [\zeta \cdot a] \\ f_{C_{>\zeta}}(a) &:= \left[\frac{1}{2\varepsilon}a + \frac{1}{2} - \frac{\zeta}{2\varepsilon}\right] \end{aligned}$$

$f_{C_{>\zeta}}$ is the continuous function connecting the points $(0,0)$, $(0,\zeta-\varepsilon)$, $(1,\zeta+\varepsilon)$, and $(1,1)$ via straight lines. All gate functions are Lipschitz with constant $K := \max(2, \zeta_{\max}, \frac{1}{2\varepsilon})$ where $\zeta_{\max}$ is the maximum ζ such that C has a $C_{\times\zeta}$ gate.

Let N be the set of nodes in the circuit. We define a function $F : [0,1]^N \to [0,1]^N$. For $x \in [0,1]^N$ and $i \in N$ let $F_i(x)$ be defined as follows:

- If i is an output of a gate g and the inputs of g are nodes $a_1, \dots, a_k$, then $F_i(x) := f_g(x_{a_1}, \dots, x_{a_k})$.
- If i is output of no gate, then $F_i(x) := x_i$.

Any ε-approximate fixed point of F is an ε-solution of C, though the converse does not hold. Since all gate functions are Lipschitz with constant K, so is F.

The first part of the lemma now follows just like in the proof of Lemma 4.5: if x is an exact fixed point of F and x' is x rounded to multiples of $\delta := \frac{\varepsilon}{K+1}$, then x' is an ε-approximate fixed point of F and thus an ε-solution of C and has polynomial length. It is not a problem that K depends on ε.

The second part of the lemma follows by reduction to the generic BROUWER problem just like in the proof of Theorem 4.6. This in fact proves that the weakly harder problem of computing an ε-solution where ε is not a parameter, but part of the input, is still in PPAD. It is again not a problem that K depends on ε because the generic BROUWER problem takes the Lipschitz constant as an input, just like ε. □