

# The Computational Complexity of Clearing Financial Networks with Credit Default Swaps\*

Steffen Schuldenzucker  
University of Zurich

Sven Seuken  
University of Zurich

Stefano Battiston  
University of Zurich

First version: October 5, 2017

This version: November 1, 2017

## Abstract

We consider the problem of clearing a system of interconnected banks. Prior work has shown 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 work, we show that the computational complexity of the clearing problem drastically increases when banks can also enter into *credit default swaps (CDSs)*, i.e., financial derivative contracts that depend on the default of another bank. We first show that many important decision problems are NP-hard once CDSs are allowed. This includes deciding if a specific bank is at risk of default and deciding if a clearing vector exists in the first place. Second, we show that computing an approximate solution to the clearing problem with sufficiently small constant error is PPAD-complete. To prove our results, we demonstrate how financial networks with debt and CDSs can encode Boolean and arithmetic operations. Our results have practical importance for network stress tests and they reveal computational complexity as a new concern regarding the stability of the financial system.

## 1 Introduction

We consider systems of banks (or other financial institutions) that are interconnected by financial contracts. Some of the banks may not be able to meet their obligations towards other banks (e.g., because they experienced a shock on their assets), thus forcing them into bankruptcy (or *default*). In this setting, we study the *clearing problem*: for each bank, we are looking for its *recovery rate*, i.e, the percentage of its liabilities it can pay to its creditors.

---

\*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, Constantinos Daskalakis, Robert Kleinberg, Timo Mennle, Thomas Noe, Christos Papadimitriou, Aviad Rubinstein, and Joseph Stiglitz for helpful comments on this work. Furthermore, we are thankful for the feedback we received from various participants at EC 2016 and ITCS 2017. Some of the ideas presented in Section 5 of this paper were also described in a one-page abstract that was published in the conference proceedings of EC'16 (Schuldenzucker, Seuken and Battiston, 2016). Section 6 of this paper contains results that were previously described in a paper that was published in the conference proceedings of ITCS'17 (Schuldenzucker, Seuken and Battiston, 2017b). This project has received funding from the European Union's Horizon 2020 research and innovation programme under the DOLFINS project, grant agreement No 640772.

Recovery rates must be in accordance with the standard bankruptcy regulations. Banks may lose a percentage of their assets upon defaulting (i.e., they incur *default costs*). The clearing problem can be challenging because banks typically cover their obligations based on the payments they receive from other banks and the structure of contractual relationships can be complex and is often cyclic.

In their seminal paper, Eisenberg and Noe (2001) showed that the clearing problem always has a solution and that this solution can be computed in polynomial time.<sup>1</sup> Their result relies on the assumption that banks can only enter into simple *debt contracts*, i.e., loans from one bank to another. We argue, however, that the growing importance of financial derivatives 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 financial contracts that depend on the default of a *reference entity*, 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 debt-only networks.<sup>2</sup>

In prior work (Schuldenzucker, Seuken and Battiston, 2017a), we have shown that existence of a solution to the clearing problem is no longer guaranteed in financial networks consisting of both debt contracts and CDSs. From this insight, two research questions arise naturally regarding the computational aspects of the clearing problem with CDSs: first, can we determine efficiently whether a solution exists in a given network? And second, given a network where a solution is known to exist, can we efficiently compute it?

In this paper, we answer both questions in the negative. Towards the first question, we show that it is computationally infeasible to make virtually any statement about the solution structure of a financial network with CDSs: determining if a solution exists as well as deciding if a specific bank will be in default in some solution are all NP-hard.<sup>3</sup> Towards the second question of computing a solution to the clearing problem, we restrict our attention to the case without default costs, where a solution is known to always exist (Schuldenzucker, Seuken and Battiston, 2017a). We show that approximately computing such a solution is PPAD-hard for a certain constant approximation quality. Thus, no polynomial-time approximation scheme (PTAS) exists, unless P=PPAD. We further show that the complexity of the problem originates not exclusively from the task of computing the precise values of the recovery rates, but already determining the set of defaulting banks is PPAD-hard.

Our hardness results have practical relevance for *stress tests*, in which regulators such as the European Central Bank (ECB) evaluate the stability of the financial system under an array of adverse economic scenarios. These stress tests are increasingly designed with a macroprudential mindset, where the financial system is viewed as a whole. Consequently, it becomes ever more important that stress tests consider network effects. For example, the ECB’s *STAMP€* framework (Dees, Henry and Martin, 2017) includes a model for the assessment of interbank contagion that is very similar to (Eisenberg and Noe, 2001). The clearing problem is solved many thousands of times to obtain a probability distribution

<sup>1</sup> The solution computed by their algorithm further has the special property that it maximizes the recovery rate of each individual bank among all solutions.

<sup>2</sup>The market for these CDSs has grown to a significant size. A report by the Bank for International Settlements (2017, section single-name CDSs, financial firms) states that the total notional of the CDSs written by financial institutions on each other was USD 1.4 trillion in the first half of 2016. D’Errico et al. (2016) have shown that these CDSs form a densely connected network.

<sup>3</sup>To establish that these problems are also in NP, and thus NP-*complete*, one would need a polynomial-time witness for, e.g., existence of a solution. We leave it open to future work to explore whether such a witness exists. The solutions themselves cannot be used here because with CDSs, solutions can be irrational (cf. below).

of losses and to compensate for uncertainty about the contract structure. Therefore, it is crucial that the clearing problem can be solved quickly. Our results imply that stress testing in networks of debt and CDSs using a similar approach would have regulators face significant computational barriers.

## 2 Related Work

The main focus of prior work on financial networks has been so far on *financial contagion* (i.e., the mechanism by which small shocks to individual market participants can lead to larger losses for the system as a whole), and on the question of which network topologies are particularly susceptible to such effects (Allen and Gale, 2000; Stiglitz, 2010; Roukny et al., 2013; Acemoglu, Ozdaglar and Tahbaz-Salehi, 2015). In particular, several works have shown how a trade-off between the stabilizing effect of diversification and the destabilizing effect of contagion emerges (Battiston et al., 2012a; Elliott, Golub and Jackson, 2014; Battiston et al., 2016a; Bardoscia et al., 2017) when obligations are valued before their maturity and there are bankruptcy costs (Battiston et al., 2012b; Barucca et al., 2016; Visentin, Battiston and D’Errico, 2016).

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. In prior work (Schuldenzucker, Seuken and Battiston, 2017a), we further extended this model to CDSs. We 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.

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, Golub and Jackson’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.

We capture the complexity of the search problem (Section 6) by means of the PPAD complexity class (Papadimitriou, 1994). This class is best known for the problem of computing a Nash equilibrium, the hardness of which was shown by reduction from generalized circuits (Daskalakis, Goldberg and Papadimitriou, 2005; Chen, Deng and Teng, 2006; Daskalakis, 2013; Rubinstein, 2015). Our work builds on this technique, and in particular on Rubinstein’s (Rubinstein, 2015) PPAD-hardness result for constant accuracy. To the best of our knowledge, we are the first to implement generalized circuits using financial networks and we are the first to present a case where the clearing problem has high computational complexity.

## 3 Formal Model and Visual Representation<sup>4</sup>

We use our formal model from (Schuldenzucker, Seuken and Battiston, 2017a). The model is based on Eisenberg and Noe’s (2001) model and its extension to default costs by Rogers

---

<sup>4</sup>This section has previously appeared in our prior work (Schuldenzucker, Seuken and Battiston, 2017a). We repeat it here for convenience.

and Veraart (2013). Both of these prior models were restricted to debt contracts. We define an extension to credit default swaps. Following said prior work, we assume a static model where a financial system is given exogenously and all contracts are evaluated simultaneously. We adjust the notation where necessary.

### 3.1 The Model

We define the elements of the financial system.

**Banks and external assets.** We denote by  $N$  a finite set of *banks*. Each bank  $i \in N$  holds a certain amount of *external assets*, denoted by  $e_i \geq 0$ . 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 swaps (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 cannot fulfill this obligation 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  $j$ . The amount is called the *notional* of the contract and is denoted by  $c_{i,j}^0$ . A *credit default swap* obliges the writer  $i$  to make a conditional payment to the holder  $j$ . The amount of this payment depends on the recovery rate of a third bank  $k$ , called the *reference entity*. Specifically, the payment amount of the CDS from  $i$  to  $j$  with reference entity  $k$  and *notional*  $c_{i,j}^k$  is  $c_{i,j}^k \cdot (1 - r_k)$ .

Note that when banks enter contracts, there typically is 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, we assume that any initial payments have been made at an earlier time and are implicitly reflected by the external assets.

The contractual relationships between all banks are represented by a 3-dimensional matrix  $c = (c_{i,j}^k)_{i \in N, j \in N, k \in N \cup \{\emptyset\}}$ . Zero entries indicate the absence of the respective contract.

We make two *sanity assumptions* to rule out pathological cases. First, we require that no bank enters into a contract with itself or on itself (i.e.,  $c_{i,i}^0 = c_{i,i}^j = c_{i,j}^j = c_{i,j}^i = 0$  for all  $i, j \in N$ ). Second, as CDSs are defined as insurance on debt, we require that any bank that is a reference entity in a CDS must also be writer of a debt contract (i.e., if  $\sum_{k,l \in N} c_{k,l}^i > 0$ , then  $\sum_{j \in N} c_{i,j}^0 > 0$  for all  $i \in N$ ).

For any bank  $i$ , the *creditors of  $i$*  are those 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 which owe money to  $i$ . 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.

**Default Costs.** We model default costs following (Rogers and Veraart, 2013): there are two *default cost parameters*  $\alpha, \beta \in [0, 1]$ . Defaulting banks are only able to pay to their creditors a share of  $\alpha$  of their external assets and a share of  $\beta$  of their incoming payments. Thus,  $\alpha = \beta = 1$  means that there are no default costs and  $\alpha = \beta = 0$  means that assets held by defaulting banks are worthless. The values  $1 - \alpha$  and  $1 - \beta$  are the *default costs*.<sup>5</sup>

<sup>5</sup>Default costs could result from legal and administrative costs, a delay in payments, or from fire sales when defaulting banks need to sell off their assets quickly. Details can be found in (Rogers and Veraart, 2013).

To simplify the exposition, we assume default costs to be the same across all banks. However, our model as well as our results easily generalize to individual default cost parameters  $\alpha_i$  and  $\beta_i$  for  $i \in N$  with minor adjustments.

**Financial System.** A *financial system* is a tuple  $(N, e, c, \alpha, \beta)$  where  $N$  is a set of banks,  $e$  is a vector of external assets,  $c$  is a 3-dimensional matrix of contracts, and  $\alpha$  and  $\beta$  are default cost parameters.

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

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

The *total liabilities of  $i$  at  $r$*  are the aggregate 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* (discussed below), a bank that is in default makes payments for its contracts in proportion to the respective liability;

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

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

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

In case bank  $i$  is in default, its *assets after default costs  $a'_i(r)$*  are the assets reduced according to the factors  $\alpha$  and  $\beta$ . This is the amount that will be paid out to creditors;

$$a'_i(r) := \alpha e_i + \beta \sum_{j \in N} p_{j,i}(r).$$

**Clearing Recovery Rate Vector.** Following Eisenberg and Noe (2001), we call a recovery rate vector  $r$  *clearing* if the payments  $p_{i,j}(r)$  are in accordance with the following three principles of standard bankruptcy law:

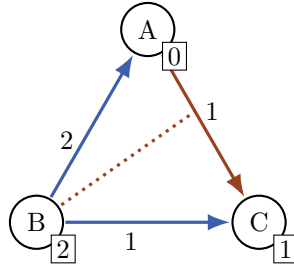
1. *Absolute Priority:* Banks with sufficient assets pay their liabilities in full. Thus, these banks have recovery rate 1.
2. *Limited Liability:* Banks with insufficient assets to pay their liabilities are in default and pay all of their assets to creditors after default costs have been subtracted. Thus, these banks have recovery rate  $\frac{a'_i(r)}{l_i(r)} < 1$ .
3. *Proportionality:* In case of default, payments to creditors are made in proportion to the respective liability.

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

---

**Figure 1** Example financial system with  $\alpha = \beta = 0.5$

---



**Definition 1** (Clearing Recovery Rate Vector). Let  $X = (N, e, c, \alpha, \beta)$  be a financial system. A *recovery rate vector* is a vector of values  $r_i \in [0, 1]$  for each  $i \in N$ . We denote by  $[0, 1]^N$  the space of all possible recovery rate vectors. Define the *update function*

$$F : [0, 1]^N \rightarrow [0, 1]^N$$

$$F_i(r) := \begin{cases} 1 & \text{if } a_i(r) \geq l_i(r) \\ \frac{a'_i(r)}{l_i(r)} & \text{if } a_i(r) < l_i(r). \end{cases} \quad (1)$$

A recovery rate vector  $r$  is called *clearing* for  $X$  if it is a fixed point of the update function, i.e., if  $F_i(r) = r_i$  for all  $i$ . We also call a clearing recovery rate vector a *solution* to the clearing problem.

*Remark 1* (Clearing Recovery Rates and Clearing Payments). Instead of clearing *recovery rates*, one may equivalently consider clearing *payments* (as Eisenberg and Noe (2001) did) and we will sometimes use this formulation in our proofs. If  $r$  is clearing, then the total payments of any bank  $i$  are either equal to its liabilities (if  $i$  is not in default) or they are equal to its assets after default costs (if  $i$  is in default). That is, we have

$$\sum_{j \in N} p_{i,j}(r) = \begin{cases} l_i(r) & \text{if } a_i(r) \geq l_i(r) \\ a'_i(r) & \text{if } a_i(r) < l_i(r). \end{cases} \quad (2)$$

Vice versa, if (2) holds, then  $r$  is clearing.<sup>6</sup>

### 3.2 Example and Visual Representation

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 top of the banks. Debt contracts are drawn as blue arrows from the writer to the holder and they are annotated with the notionals  $c_{B,A}^\emptyset = 2$  and  $c_{B,C}^\emptyset = 1$ . CDSs are drawn as orange arrows, where a dashed line connects to the reference entity, and are also annotated with the notionals:  $c_{A,C}^B = 1$ . Default cost parameters  $\alpha = \beta = 0.5$  are given in addition to the picture.

A clearing recovery rate vector for this example is given by  $r_A = 1$ ,  $r_B = \frac{1}{3}$ , and  $r_C = 1$ . The liabilities arising from this recovery rate vector are  $l_{B,A}(r) = 2$ ,  $l_{B,C}(r) = 1$ , and

---

<sup>6</sup>One special case must be considered separately: banks that have zero liabilities. The recovery rates of these banks are left unconstrained by (2), but are required to be equal to 1 by Definition 1. However, due to our assumptions, no other bank depends on these banks, so this difference does not matter. Thus,  $r$  becomes clearing (according to Definition 1) by simply setting the recovery rates of these banks to 1.

$l_{A,C}(r) = \frac{2}{3}$ . Payments are  $p_{B,A}(r) = \frac{2}{3}$ ,  $p_{B,C} = \frac{1}{3}$ , and  $p_{A,C}(r) = \frac{2}{3}$ . This is the only solution for this system.

## 4 The Effect of CDSs on the Solution Set

Rogers and Veraart (2013) have shown that in debt networks, the function  $F$  is always piecewise linear and monotonic (i.e., if  $r \leq r'$  point-wise, then also  $F(r) \leq F(r')$  point-wise). This implies that the clearing problem always has a solution and it in fact always has a solution that point-wise dominates all other solutions. Eisenberg and Noe’s (2001) *fictitious default algorithm* computes this solution, relying on linearity and monotonicity.

In our prior work, we have shown that the behavior of a financial system changes radically once CDSs are introduced into the model. Looking at the update function  $F$ , the assets  $a_i(r)$  now not only contain linear terms of form  $c_{j,i}^0 \cdot r_j$ , but also terms of form  $c_{j,i}^k \cdot r_j \cdot (1 - r_k)$ . That is why  $F_i$  is *non-linear* and *non-monotonic* in  $r$ . For example, in the above term,  $F_i$  would be increasing in  $r_j$  and decreasing in  $r_i$ . In more complex situations, whether  $F_i$  is increasing or decreasing in  $r_j$  can depend on the recovery rates of the other banks. Non-linearity and in particular non-monotonicity are the reason why prior algorithmic approaches cannot be used any more with financial systems that contain CDSs.

We have shown in (Schuldenzucker, Seuken and Battiston, 2017a) that the different properties of the update function  $F$  imply different possible shapes of the solution set compared to debt-only systems:

**Proposition 1** (Schuldenzucker, Seuken and Battiston (2017a)).

1. For any pair  $(\alpha, \beta)$  with  $\alpha < 1$  or  $\beta < 1$  there exists a financial system  $(N, e, c, \alpha, \beta)$  that has no clearing recovery rate vector.
2. Any financial system  $(N, e, c, \alpha = 1, \beta = 1)$  has a clearing recovery rate vector.
3. For any  $\alpha$  and  $\beta$  there exists a financial system  $(N, e, c, \alpha, \beta)$  with four banks that has exactly two clearing recovery rate vectors, namely  $(1, 0, 1, 1)$  and  $(0, 1, 1, 1)$ .<sup>7</sup>

Part 1 of the proposition shows that as soon as any default costs are present and CDSs are allowed, the clearing problem may have no solution at all. Our counterexample is illustrated in Figure 2. Intuitively, the reason why this system has no solution is that if  $A$  is in default, then  $B$ , and therefore  $A$  itself receive the CDS payment, and therefore  $A$  “should not” be in default. Vice versa, if  $A$  is not in default, then  $B$  and thus  $A$  receive nothing, so  $A$  “should be” in default. Default costs create a discontinuity in the update function  $F$  that separates the states “in default” and “not in default” from each other so that this case distinction is justified.

Part 2 shows that default costs are necessary for non-existence: if default costs are not present, then a solution always exists. The proof, however, is by a non-constructive fixed-point argument so that it is not immediately clear how a solution would be found in this case. We discuss the complexity of the associated search problem in Section 6.

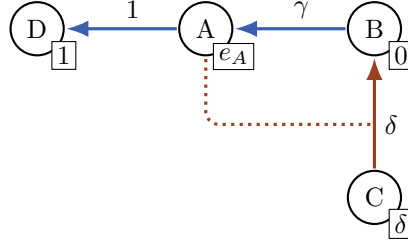
Part 3 of the proposition illustrates a particularly undesirable case of multiple solutions: while in debt-only systems, it would typically be enough to only consider the unique maximal solution, CDSs can lead to a situation where one has to deal with true multiplicity. The corresponding construction is depicted in Figure 3. The intuition is that  $A$  relies on the

<sup>7</sup> The original result in (Schuldenzucker, Seuken and Battiston, 2017a) had three solutions in case  $\beta = 1$ . We slightly adjusted our example, shown in Figure 3, such that there are always exactly two solutions.

---

**Figure 2** (Schuldenzucker, Seuken and Battiston, 2017a) Financial system with no solution for  $\alpha < 1$  or  $\beta < 1$ . For  $\beta < 1$ , let  $e_A = 0$ ,  $\gamma = 1$ , and  $\delta = 3 \cdot \frac{1}{1-\beta}$ . For  $\alpha < \beta = 1$ , let  $e_A = \frac{1}{2}$ ,  $\gamma = \frac{3-\alpha}{4}$ , and  $\delta = \frac{3-\alpha}{1-\alpha}$ .

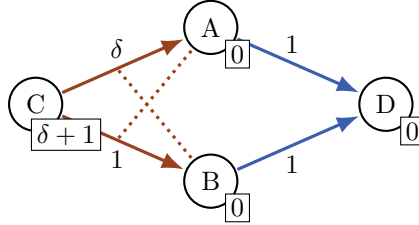
---




---

**Figure 3** (adapted from (Schuldenzucker, Seuken and Battiston, 2017a)) Financial system with solutions  $(1, 0, 1, 1)$  and  $(0, 1, 1, 1)$ . Let  $\delta = \frac{1}{1-\beta}$  if  $\beta < 1$  and  $\delta > 1$  arbitrary if  $\beta = 1$ .

---



default of  $B$  and vice versa, so exactly one of the two banks can be made well off. Note that this counterexample can be constructed both with or without default costs.

The two counterexamples in Figure 2 and 3 will serve as important building blocks for our reduction proofs in Section 5 and 6. Note that our counterexamples have size polynomial in the lengths of  $\alpha$  and  $\beta$ .

## 5 The Complexity of Deciding Defaults

Given a financial network, we consider two fundamental decision problems: first, we want to know if a solution to the clearing problem exists at all. Second, for an individual bank, we are interested in whether this bank defaults in all solutions, no solution, or if its default depends on the solution. We will show that all of these problems are NP-hard.

We show NP-hardness via reduction from the Circuit Satisfiability problem.<sup>8</sup> We proceed in three steps:

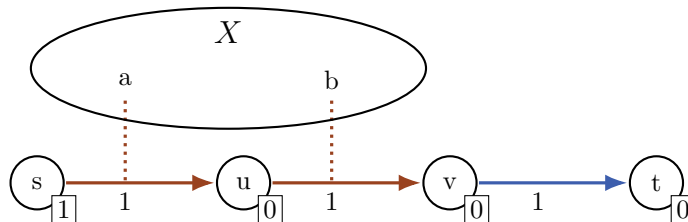
1. We show how a single logic gate can be translated into a financial system.
2. We combine copies of these financial gates to translate a whole Boolean circuit into a

---

<sup>8</sup> Recall that a Boolean circuit is an abstract description of a digital logic circuit. An acyclic graph structure connects the inputs of the circuit and a set of logic gates, encoding Boolean functions such as AND, OR, or NOT. A single node is marked as the *output*. The inputs receive an *assignment* (a tuple of 0s and 1s) and the value of the output is computed by evaluating all gates. The *size* of a Boolean circuit is the number of gates plus the number of inputs. The NP-complete Circuit Satisfiability problem asks for a given Boolean circuit whether there is an assignment that makes the output 1.



**Figure 4** Existing financial system  $X$  with input banks  $a$  and  $b$  and new banks  $s, t, u, v$  and contracts that translate a NOR gate. We have  $r_v = r_a \text{ NOR } r_b$  whenever  $r$  is clearing.



financial system. We additionally use Figure 3 (two solutions) for the inputs of the circuit and we use Figure 2 (no solution) to constrain the output of the circuit.

3. We use this translation to show how, assuming that we have an efficient algorithm to decide the above questions, we could obtain an efficient algorithm to solve Circuit Satisfiability.

## 5.1 Boolean Circuits as Financial Systems

We can assume WLOG that the involved Boolean circuits are constructed entirely of NOR gates, defined by  $a \text{ NOR } b = \neg(a \vee b)$ . This assumption is valid because NOR forms a basis of propositional logic. To translate a Boolean circuit into a financial system, we need thus only be able to translate its basic building block, a NOR gate. We then combine these gates to larger circuits in a second step.

The translation of a NOR gate cannot be described as a single financial system because financial systems lack the notion of an “input”. Instead, the following lemma shows how to *extend* an *existing* financial system with two *input banks*  $a$  and  $b$  by adding four new banks in such a way that the recovery rate of one of the new banks is always exactly  $a \text{ NOR } b$ . This extension is illustrated in Figure 4.

**Lemma 1** (Financial NOR Gate). *Let  $X = (N, e, c, \alpha, \beta)$  be a financial system and let  $a, b \in N$  be banks (not necessarily different). Assume that for all  $r$  clearing for  $X$  and all  $i \in N$  we have  $r_i \in \{0, 1\}$ . Then there exists an extension  $X'$  obtained from  $X$  by adding four banks  $s, t, u, v$  such that*

1. *If  $r$  is clearing for  $X$ , then there exists  $r'$  clearing for  $X'$  such that  $r'|_N = r$ .*
2. *If  $r'$  is clearing for  $X'$ , then  $r'|_N$  is clearing for  $X$  and  $r'_v = r'_a \text{ NOR } r'_b$ .*

Here,  $r'|_N$  is the vector  $r'$  with only the indices from  $N$ , i.e., without the indices  $s, t, u, v$ .

*Proof.* Let  $X'$  result from  $X$  by adding the banks and contracts in Figure 4. It is clear that the construction does not change the solution structure of the sub-system  $X$  of  $X'$  because  $X$  is not affected by the recovery rates of the new banks  $s, t, u, v$  and the sub-system consisting of the new banks always has a solution. Hence, solutions of  $X$  can be extended to solutions of  $X'$ , proving statement 1.

If  $r'$  is clearing for  $X'$ , then  $r'_a$  and  $r'_b$  can only be 0 or 1 by assumption. If  $r_a = 0$  and  $r_b = 0$ , then  $v$  has assets of 1 and thus a recovery rate of 1. Otherwise, the flow of money from  $s$  is “blocked” before either  $u$  or  $v$  and  $v$  has assets of 0 and thus a recovery rate of 0. This corresponds exactly to the definition of the NOR operation, proving statement 2.  $\square$

We can now to translate a complete Boolean circuit.

**Lemma 2** (Financial Boolean Circuit). *Let  $C$  be a Boolean circuit with  $m$  inputs. For  $\chi \in \{0, 1\}^m$  write  $C(\chi)$  for the value of the output of  $C$  given values  $\chi$  of the inputs.*

*For any  $\alpha \in [0, 1]$  and  $\beta \in [0, 1]$  there exists a financial system  $X = (N, e, c, \alpha, \beta)$  of size linear in the size of  $C$  with distinguished input banks  $a_1, \dots, a_m \in N$  and a distinguished output bank  $v \in N$  such that the following hold:*

1. *For any assignment  $\chi \in \{0, 1\}^m$  there exists a clearing recovery rate vector  $r$  for  $X$  such that  $r_{a_i} = \chi_i$  for  $i = 1, \dots, m$ .*
2. *If  $r$  is clearing for  $X$ , then  $r_i \in \{0, 1\}$  for any bank  $i \in N$ .*
3. *If  $r$  is clearing for  $X$ , then  $r_v = C(r_{a_1}, \dots, r_{a_m})$ .*

*Proof outline (full proof in Appendix A).* WLOG we can assume that all gates are NOR gates. Boolean circuits are acyclic, so we can proceed in topological order, starting at the inputs and ending at the output. We translate inputs into a copy of the 0-1 system from Figure 3 and we translate NOR gates by applying Lemma 1 to the part of the system that has been constructed so far.  $\square$

## 5.2 Deciding the Default of an Individual Bank

We now prove hardness of our decision problems. We begin with the second group of decision problems discussed above, i.e., deciding for a single bank whether it is in default in some or all solutions. This problem can be stated for any values of  $\alpha$  and  $\beta$ , including 1.

We use the following problems for reduction.

**Definition 2** (Circuit Problems). Define the following decision problems:

- *Circuit Satisfiability (Falsifiability):* Given a Boolean circuit, decide if there exists an assignment of inputs such that the output is 1 (0).
- *Circuit Non-Constancy:* Given a Boolean circuit, decide if it is true that there are two assignments of inputs: one that makes the output 1 and one that makes it 0.

It is well-known that Circuit Satisfiability is NP-complete. Circuit Falsifiability is NP-complete because Circuit Satisfiability is just Falsifiability in a circuit with an additional NOT gate. Circuit Non-Constancy is NP-complete because we can solve Circuit Satisfiability by solving Circuit Non-Constancy and evaluating the circuit on a single input.

**Theorem 1** (NP-hardness of Deciding the Default of an Individual Bank). *The following problems are NP-hard: Given a financial system  $X$  and a bank  $i$  in  $X$ , decide if*

- a) POSSIBLE DEFAULT: *there exists  $r$  clearing such that  $r_i < 1$ .*
- b) POSSIBLE NON-DEFAULT: *there exists  $r$  clearing such that  $r_i = 1$ .*
- c) CERTAIN DEFAULT: *for all  $r$  clearing we have  $r_i < 1$ .*
- d) CERTAIN NON-DEFAULT: *for all  $r$  clearing we have  $r_i = 1$ .*
- e) MULTIPLICITY: *there exist  $r$  and  $r'$  clearing such that  $r_i \neq r'_i$*

*Proof.* a): Reduction from Circuit Falsifiability. Given an instance  $C$  of Circuit Falsifiability, let  $X$  be the financial system resulting from Lemma 2 and let  $v$  be the output bank. By construction, if  $r$  is clearing for  $X$  and  $r_v < 1$ , then  $r_v = 0$  and  $(r_{a_1}, \dots, r_{a_m})$  is a falsifying

assignment. Vice versa, any falsifying assignment  $\chi \in \{0, 1\}^n$  gives rise to a clearing recovery rate vector  $r$  with  $r_{a_i} = \chi_i$  for  $i \in \{1, \dots, n\}$  and  $r_v = C(\chi) = 0 < 1$ .

b): Reduction from Circuit Satisfiability like for a).

c) and d): these are the complements of b) and a), respectively, and are thus NP-hard as well.<sup>9</sup>

e): Reduction from Circuit Non-Constancy like for a). For our particular construction we know that  $r_v \neq r'_v$  iff  $r_v = 0$  and  $r'_v = 1$  or vice versa.  $\square$

Theorem 1 has an important implication for network stress tests: regulators may want to simulate an array of different scenarios and test for each one if specific banks are at risk of defaulting. This should take all solutions into account since it is not clear a priori which of these would be chosen. However, Theorem 1 shows that such a stress test is computationally infeasible in a general network of debt and CDSs.

### 5.3 Deciding Existence of a Solution

We can now prove the hardness of deciding if a given financial system has a solution. Recall that financial systems with  $\alpha = \beta = 1$  always have a solution, so this decision problem only makes sense for  $\alpha < 1$  or  $\beta < 1$ .

**Theorem 2** (NP-Hardness of Determining Existence). *The following problem is NP-hard: Given a financial system  $(N, e, c, \alpha, \beta)$  with  $\alpha < 1$  or  $\beta < 1$ , decide if it has a clearing recovery rate vector.*

*Proof outline (full proof in Appendix A).* Reduction from Circuit Satisfiability. Given a Boolean circuit  $C$ , we construct the translation into a financial system as of Lemma 2. On top of that, we add a copy of the system from Figure 2 (no solution), but with one of the debt contracts replaced by a CDS on the output bank of the circuit translation.

If now a falsifying assignment is supplied to the input banks, then the output bank has recovery rate 0, so the “no solution” system is active, and thus no such recovery rate vector can be clearing. In effect, the only possible clearing recovery rate vectors correspond to satisfying assignments of the circuit  $C$ , where the output bank has recovery rate 1 and one contract of the “no solution” system drops out, in which case it *has* a solution. Thus, determining whether there is a clearing recovery rate vector in this financial system is equivalent to determining whether there is a satisfying assignment for  $C$ , which is NP-hard.  $\square$

Another important property of financial systems with CDSs is whether a solution exists that maximizes the equity, defined as  $E_i(r) = \max(0, a_i(r) - l_i(r))$ , of each individual bank  $i$ . Such a *maximal* solution would be preferred by each bank and it would also minimize the amount of money lost due to default costs. In debt-only systems, the unique solution that maximizes banks’ recovery rates also maximizes the equities. With CDSs in contrast, a point-wise equity-maximizing solution may not exist and it is NP-hard to decide if it does.

**Corollary 1** (NP-Hardness of Determining Maximality). *The following problem is NP-hard: Given a financial system  $X = (N, e, c, \alpha, \beta)$  that has a solution, decide if there is a solution that point-wise maximizes the vector of equities  $(E_i(r))_{i \in N}$  among all solutions  $r$ .*

<sup>9</sup> Recall that complements of NP-hard problems are NP-hard, where the reduction is by simply interchanging the “Yes” and “No” answers. This is not to be confused with the analogous proposition for NP-completeness, which would be equivalent to NP=coNP.

*Proof.* Proof by reduction from Circuit Satisfiability. Given is a Boolean circuit. We modify our construction from Theorem 2 by adding a single “trivial” solution as follows: add a copy of the system from Figure 3 (two solutions). We will refer to the banks  $A$  and  $C$  from this copy of the system in the following. Further add a new “source bank”  $\sigma$ . For each bank  $i$  in our original construction that had any positive external assets  $e_i$ , add a CDS  $c_{\sigma,i}^A = e_i$ , and then set  $e_i$  to 0. Set  $e_\sigma := \sum_i c_{\sigma,i}^A$  so that  $\sigma$  cannot default.

There are two types of solutions in this new system.

1. If  $r_A = 1$ , none of the CDSs  $c_{\sigma,i}^A$  pays anything. Thus, every bank other than  $C$  and  $A$  has no assets, so its recovery rate is 0 if it has positive liabilities or 1 if it has zero liabilities. Thus, there is exactly one solution with  $r_A = 1$ . In this solution, we have  $E_A(r) = \delta - 1 > 0$  and  $E_C(r) = 1$ .
2. If  $r_A = 0$ , then the CDS payments in the contracts  $c_{\sigma,i}^A$  equal the external assets banks had in our original construction in Theorem 2, so the behavior of the system is the same as there. In particular, a solution with  $r_A = 0$  exists iff the Boolean circuit has a satisfying assignment. In any such solution, we have  $E_A(r) = 0$  and  $E_C(r) = \delta > 1$ .

Overall, if the given Boolean circuit has no satisfying assignment, then there is a unique (and in particular equity-maximizing) solution, namely the one with  $r_A = 1$ . If the Boolean circuit has a satisfying assignment, then there are at least two solutions: in the one with  $r_A = 1$ ,  $A$  has higher equity and in the ones with  $r_A = 0$ ,  $C$  has higher equity. Thus, no equity-maximizing solution exists. This completes the reduction.  $\square$

Theorem 2 and Corollary 1 are again relevant for stress testing. In (Schuldenzucker, Seuken and Battiston, 2017a), we have argued that when the clearing problem has no solution or no maximal solution, this can lead to a phenomenon we call *default ambiguity*, i.e., a situation where it is impossible to tell which banks are in default. We have argued that such a situation can lead to a “paralysis” and delay banks’ resolution, which could in turn exacerbate a financial crisis. Having seen the danger that comes with default ambiguity, regulators may want to simulate an array of different scenarios and test each of them for whether there is a (maximal) solution. They could then estimate the probability of default ambiguity and they would know which scenarios are particularly problematic and should therefore be prevented at all cost. However, the theorem and corollary show that, due to computational complexity, such a simulation could not easily be carried out.

## 6 The Complexity of Approximating a Solution without Default Costs

We know from Proposition 1 that if there are no default costs, then a solution to the clearing problem always exists. Note that in this section, we consider the corresponding total search problem. Since there are financial systems where all solutions contain irrational numbers (a simple example is provided in Appendix B), the best we can hope for is an algorithm that computes a recovery rate vector that is in some sense *approximately* clearing.

In this section, we always assume that  $\alpha = \beta = 1$ . Thus, a financial system can be described by a triple  $(N, e, c) := (N, e, c, 1, 1)$ .

## 6.1 The $\varepsilon$ -FINDCLEARING Search Problem

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 section is to relax the (essentially equivalent) definition of being clearing from Remark 1. 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.

Following the alternative definition of being clearing from Remark 1 for  $\alpha = \beta = 1$ ,  $r$  is clearing if  $r_i = [\frac{a_i(r)}{l_i(r)}]$  for all  $i$  for which  $l_i(r) > 0$ .  $r_i$  is arbitrary if  $l_i(r) = 0$ . We relax this as follows: for  $\varepsilon \geq 0$  and  $i \in N$  let

$$\rho_i^\varepsilon(r) : [0, 1]^N \rightarrow 2^{[0,1]}$$

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

and let  $\rho^\varepsilon : [0, 1]^N \rightarrow 2^{[0,1]^N}$  be defined as  $\rho^\varepsilon(r) := \times_{i \in N} \rho_i^\varepsilon(r)$ .

**Definition 3** (Approximately Clearing Recovery Rate Vector). Fix a financial system without default costs and let  $\varepsilon \geq 0$ . A recovery rate vector  $r$  is called  $\varepsilon$ -*approximately clearing* or an  $\varepsilon$ -*solution* if it is a fixed point of the set-valued function  $\rho^\varepsilon$ , i.e., if  $r \in \rho^\varepsilon(r)$ . For clarity, we refer to solutions that are not approximate as *exact solutions*. Note that for  $\varepsilon = 0$  we have  $\rho^\varepsilon = \rho$ , so 0-solutions are the same thing as exact solutions.

Our definition of an approximate solution has many desirable properties from an economic and technical point of view. We provide a discussion in Appendix C. Note in particular that if  $r$  is an  $\varepsilon$ -solution and  $l_i(r) > 0$ , then  $r_i = [\frac{a_i(r)}{l_i(r)}] \pm \varepsilon$ , though the converse does not necessarily hold.

It is easy to see that for any  $\varepsilon > 0$ , there always exists an  $\varepsilon$ -solution of finite length. To guarantee that there is also an  $\varepsilon$ -solution of *polynomial* length, we make an additional assumption that we call *non-degeneracy*.<sup>10</sup> We can then state our search problem.

**Definition 4** (Non-degenerate Financial System). A financial system without default costs  $X = (N, e, c)$  is called *non-degenerate* if each bank that writes a CDS also writes a debt contract or has strictly positive external assets.

**Definition 5** ( $\varepsilon$ -FINDCLEARING Problem). For any parameter  $\varepsilon > 0$ ,  $\varepsilon$ -FINDCLEARING is the following total search problem: given a non-degenerate financial system without default costs, find an  $\varepsilon$ -solution.

The following lemma establishes that under the assumption of non-degeneracy, sufficiently “short” approximate solutions always exist in the vicinity of exact solutions, thus making  $\varepsilon$ -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 equilibrium also has this property.

**Lemma 3** ( $\varepsilon$ -FINDCLEARING is Well-posed and in PPAD).

<sup>10</sup>It is an open question whether or not  $\varepsilon$ -solutions of polynomial length are also guaranteed to exist when this assumption is not made.

1. If  $X = (N, e, c)$  is a non-degenerate financial system without default costs and  $\varepsilon > 0$ , then there exists an  $\varepsilon$ -solution of length polynomial in the length of  $X$  and the length of  $\varepsilon$ .
2. For any  $\varepsilon > 0$ , the problem  $\varepsilon$ -FINDCLEARING is in PPAD.

*Proof Outline (full proof in Appendix D).* We define a (not set-valued) function  $G$  such that any  $\varepsilon$ -approximate fixed point of  $G$  gives rise to an  $\varepsilon$ -solution of  $X$ . We prove that since  $X$  is non-degenerate,  $G$  has a polynomial Lipschitz constant. We then round the exact fixed point of  $G$  to a grid dependent on the Lipschitz constant to receive an approximate fixed point and thus an approximate solution.  $\square$

Our main contribution in this section is the proof that  $\varepsilon$ -FINDCLEARING is PPAD-hard, and thus PPAD-complete, for a sufficiently small constant  $\varepsilon$ .

**Theorem 3.** *There exists an  $\varepsilon > 0$  such that the  $\varepsilon$ -FINDCLEARING problem is PPAD-hard.*

The theorem immediately implies:

**Corollary 2.** *There is no polynomial-time approximation scheme that computes an  $\varepsilon$ -solution for a given financial system without default costs and a given  $\varepsilon$ , 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 6.2). We then reduce this problem to  $\varepsilon$ -FINDCLEARING (Section 6.3).

## 6.2 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 slightly adapt Rubinstein’s definition by assuming a reduced set of gates (cf. the proof of Lemma 5 below for a detailed comparison).

**Definition 6** (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*  $C_\zeta$  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  $\zeta$*  gate  $C_{\times\zeta}$  with one input and one output.
- For each  $\zeta \in (0, 1)$  the *compare to  $\zeta$*  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  $\zeta$  values involved.

---

**Figure 5** Conditions to hold at the different gates in an  $\varepsilon$ -solution of a generalized circuit

---

$$\begin{aligned}
g = C_\zeta &\Rightarrow x[v] = \zeta \pm \varepsilon \\
g = C_+ &\Rightarrow x[v] = [x[a_1] + x[a_2]] \pm \varepsilon \\
g = C_- &\Rightarrow x[v] = [x[a_1] - x[a_2]] \pm \varepsilon \\
g = C_{\times\zeta} &\Rightarrow x[v] = [\zeta \cdot x[a_1]] \pm (1 + \zeta) \varepsilon \\
g = C_{>\zeta} &\Rightarrow x[a_1] \leq \zeta - \varepsilon \Rightarrow x[v] = 0 \pm \varepsilon \\
&\quad x[a_1] \geq \zeta + \varepsilon \Rightarrow x[v] = 1 \pm \varepsilon \\
g = C_\neg &\Rightarrow x[a_1] = 0 \pm \varepsilon \Rightarrow x[v] = 1 \pm \varepsilon \\
&\quad x[a_1] = 1 \pm \varepsilon \Rightarrow x[v] = 0 \pm \varepsilon \\
g = C_\vee &\Rightarrow x[a_1] = 0 \pm \varepsilon \text{ and } x[a_2] = 0 \pm \varepsilon \Rightarrow x[v] = 0 \pm \varepsilon \\
&\quad x[a_1] = 1 \pm \varepsilon \text{ or } x[a_2] = 1 \pm \varepsilon \Rightarrow x[v] = 1 \pm \varepsilon
\end{aligned}$$


---

If  $\varepsilon \geq 0$  and  $C$  is a generalized circuit, then an  $\varepsilon$ -approximate solution (or  $\varepsilon$ -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  $a_1, \dots, a_l$  and output  $v$  the respective condition from Figure 5 holds.

**Definition 7** ( $\varepsilon$ -GCIRCUIT Problem). For any parameter  $\varepsilon > 0$ ,  $\varepsilon$ -GCIRCUIT is the following total search problem: given a generalized circuit, find an  $\varepsilon$ -solution.

Note how the comparison gadget  $C_{>\zeta}$  is *brittle*: its value is arbitrary if  $x[a_1]$  is close to  $\zeta$ . 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  $\varepsilon$  errors at each bank. Note how chains of Boolean gadgets do not accumulate errors, but chains of arithmetic gadgets do.

It is well known that  $\varepsilon$ -GCIRCUIT is well-posed and in PPAD. We provide a simple lemma for our variant of  $\varepsilon$ -GCIRCUIT for completeness. PPAD-hardness, and thus PPAD-completeness, of the  $\varepsilon$ -GCIRCUIT problem for constant  $\varepsilon$  follows by reduction from Rubinfeld's variant. Both proofs can be found in Appendix E.

**Lemma 4** ( $\varepsilon$ -GCIRCUIT is Well-posed and in PPAD).

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

**Lemma 5.** *There exists an  $\varepsilon > 0$  such that the  $\varepsilon$ -GCIRCUIT problem is PPAD-hard.*

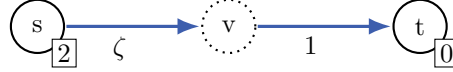
### 6.3 Reduction from Generalized Circuits to Financial Systems

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*.

---

**Figure 6** Constant Gadget: extension of an existing financial system with output bank  $v$  by new banks  $s, t$  and contracts such that  $r_v = \zeta \pm \varepsilon$ .

---



**Definition 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*  $a_1, \dots, a_l \in N$  where  $l$  depends on the gadget, and an *output bank*  $v \in N$  such that  $v$  has no assets or liabilities in  $X$ , i.e.,  $e_v = c_{v,j}^k = c_{j,v}^k = 0$  for all  $j \in N$  and  $k \in N \cup \{\emptyset\}$ .
- $X'$  consists of  $X$  together with some new banks and contracts.
- For any  $\varepsilon$  and any  $\varepsilon$ -solution  $r'$  of  $X'$ , the restriction  $r := r'|_N$  is an  $\varepsilon$ -solution for  $X$ .
- For any  $\varepsilon$  and any  $\varepsilon$ -solution  $r$  of  $X$ , there is an  $\varepsilon$ -solution  $r'$  of  $X'$  such that  $r'_i = r_i$  for all  $i \in N \setminus \{v\}$ .

In addition to these properties, gadgets typically establish some relationship between the recovery rates of the input and output banks. 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 Figures 6–9, are *fundamental* while the others are defined as combinations of the fundamental ones. We use our visual representation for financial systems where we draw the (existing) input and output banks as dotted circles and the new banks as solid circles. 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. Our gadgets will be accurate up to an error of  $3\varepsilon$ . We will later compensate for the factor 3 by choosing  $\varepsilon$  by factor 3 smaller. All gadgets lead to non-degenerate financial systems.

### 6.3.1 Addition Gadgets

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

**Lemma 6** (Constant Gadget). *Let  $\zeta \in [0, 1]$ . There is a financial system gadget with no input banks and with output bank  $v$  such that if  $r$  is an  $\varepsilon$ -solution, then  $r_v = \zeta \pm \varepsilon$ .*

*Proof.* Consider the gadget in Figure 6. We have  $\frac{a_s(r)}{l_s(r)} \geq 2 \geq 1 + \varepsilon$ . It is easy to see that this implies that  $r_s = 1$  in any  $\varepsilon$ -solution. Thus,  $s$  pays in full and  $a_v(r) = \zeta$  and  $l_v(r) = 1 \geq a_v(r)$ , so in an  $\varepsilon$ -solution  $r_v = \frac{a_v(r)}{l_v(r)} \pm \varepsilon = \zeta \pm \varepsilon$ .  $\square$

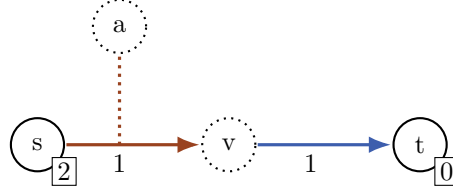
An important building block for the following constructions is a gadget that “inverts” the recovery rate of a bank.



---

**Figure 7** Inverter Gadget: extension of an existing financial system with input bank  $a$  and output bank  $v$  by new banks  $s, t$  and contracts such that  $r_v = 1 - r_a \pm \varepsilon$ .

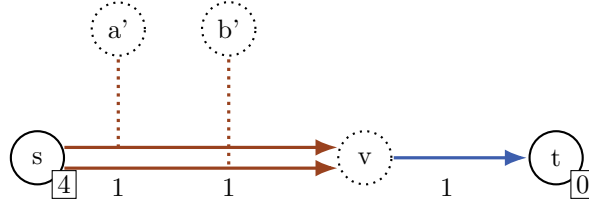
---




---

**Figure 8** 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  $r_a + r_b$ .

---



**Lemma 7** (Inverter Gadget). *There is a financial system gadget with input bank  $a$  and output bank  $v$  such that if  $r$  is an  $\varepsilon$ -solution, then  $r_v = 1 - r_a \pm \varepsilon$ .*

*Proof.* Consider the gadget in Figure 7. Since  $l_v(r) = 1$  we have in any  $\varepsilon$ -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 8** (Sum Gadget). *There is a financial system gadget with input banks  $a$  and  $b$  and output bank  $v$  such that if  $r$  is an  $\varepsilon$ -solution, then  $r_v = [r_a + r_b] \pm 3\varepsilon$ .*

*Proof.* Apply inverter gadgets (Lemma 7) to both  $a$  and  $b$  and call the output banks  $a'$  and  $b'$ , respectively. Now consider the gadget in Figure 8. We have

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

**Lemma 9** (Difference Gadget). *There is a financial system gadget with input banks  $a$  and  $b$  and output bank  $v$  such that if  $r$  is an  $\varepsilon$ -solution, then  $r_v = [r_a - r_b] \pm 3\varepsilon$ .*

*Proof.* Apply an inverter gadget (Lemma 7) to  $a$  and call the output bank  $a'$ . Apply the gadget in Figure 8 to  $a'$  and  $b' := b$  and call the output bank  $u$ . From the proof of the previous lemma we know that

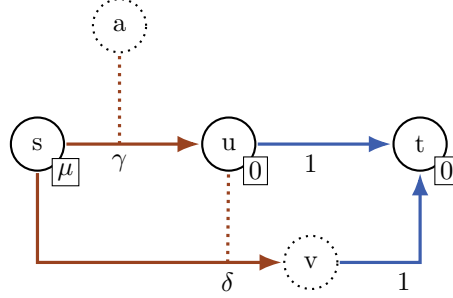
$$r_u = [1 - r_a + r_b] \pm 2\varepsilon$$

where the error is by one  $\varepsilon$  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:

---

**Figure 9** Amplifier Gadget: extension of an existing financial system with input bank  $a$  and output bank  $v$  by new banks  $s, t, u$  and contracts that translate the function  $f$  from Lemma 10. Let  $\mu = 2(\gamma + \delta)$ .

---



- If  $r_a \leq r_b$ , then  $1 - r_a + r_b \geq 1$ , so  $r_u = 1 \pm 2\varepsilon$  and thus  $r_v = 1 - r_u \pm \varepsilon = 0 \pm 3\varepsilon = [r_a - r_b] \pm 3\varepsilon$  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  $r_v = 1 - r_u \pm \varepsilon = r_a - r_b \pm 3\varepsilon = [r_a - r_b] \pm 3\varepsilon$  as required.  $\square$

### 6.3.2 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 10** (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*

$$f : [0, 1] \rightarrow [0, 1]$$

$$f(r_a) := \left[ \frac{1}{L - K} r_a - \frac{K}{L - K} \right].$$

*Note that  $f$  is monotonically increasing with  $f(K) = 0$  and  $f(L) = 1$ .*

*There is a financial system gadget with input bank  $a$  and output bank  $v$  such that if  $r$  is an  $\varepsilon$ -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 9 with

$$\gamma := \frac{1}{1 - K}$$

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

Let  $r$  be an  $\varepsilon$ -solution. 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  $z \geq 0$  and the last line is by simple algebra. Thus,  $r_v$  is as desired.  $\square$

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

**Corollary 3** (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  $\varepsilon$ -solution, then  $r_v = [\zeta r_a] \pm (1 + \zeta)\varepsilon$ . The construction can be performed in time polynomial in the length of  $\zeta$ .*

*Proof.* Use an amplifier gadget (Lemma 10) with  $K = 0$  and  $L = \frac{1}{\zeta}$ . Then  $f(r_a) = [\zeta r_a]$  and  $\delta = \zeta$ .  $\square$

We receive a gadget that acts like the brittle comparison gate  $C_{>\zeta}$  by choosing  $K$  and  $L$  closely together around a central point  $\zeta$ . The gadget is less “brittle” the closer  $K$  and  $L$  are together, but this also increases the value  $\delta$  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  $3\varepsilon$ .

**Corollary 4** (Reset Gadget). *There is a financial system gadget with input bank  $a$  and output bank  $v$  such that if  $r$  is an  $\varepsilon$ -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 10) with  $K = \frac{1}{4}$  and  $L = \frac{3}{4}$ . We have  $\delta + 1 = \frac{5}{2} < 3$ . If  $r_a \leq \frac{1}{4}$ , then  $f(r_a) = 0$ , so  $r_v = f(r_a) \pm (1 + \delta)\varepsilon = 1 \pm 3\varepsilon$ . Likewise for  $r_a \geq \frac{3}{4}$ .  $\square$

**Corollary 5** (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  $\varepsilon$ -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  $\zeta$ .*

*Proof.* We apply two constructions involving the amplifier gadget (Lemma 10): 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 4) to  $u$  as the input bank to receive the desired lower output error of  $3\varepsilon$ .  $\square$

### 6.3.3 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 11** (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  $\varepsilon$ -solution, then*

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

*Proof.* 1. Apply a sum gadget (Lemma 8) to  $a$  and  $b$  and call the output bank  $u$ . Now apply a reset gadget (Corollary 4) 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  $r_v = 1 \pm 3\varepsilon$ . If  $r_a, r_b \leq 3\varepsilon$ , then  $r_u \leq 9\varepsilon \leq \frac{1}{4}$ , so  $r_v = 0 \pm 3\varepsilon$ .

2. Apply similarly an inverter gadget (Lemma 7) and then a reset gadget. It is easy to show that the construction behaves as desired.  $\square$

### 6.3.4 Completing the PPAD-hardness Proof

We combine our gadgets to model generalized circuits, thus reducing  $\varepsilon$ -GCIRCUIT to  $\varepsilon'$ -FINDCLEARING (with  $0 < \varepsilon' < \varepsilon$ ) and proving PPAD-hardness of  $\varepsilon$ -FINDCLEARING:

*Proof of Theorem 3.* Let  $\varepsilon > 0$  be arbitrary. We reduce  $\varepsilon$ -GCIRCUIT to  $\varepsilon'$ -FINDCLEARING where  $\varepsilon' := \frac{\varepsilon}{3}$ . Assume that we are given a generalized circuit  $C$  with  $n$  nodes and  $m$  gates. Construct a financial system via the following algorithm.

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

For  $t = 0, \dots, m$  let  $C^t$  be  $C$  restricted to the first  $t$  gates. We show by induction on  $t$  that the  $\varepsilon'$ -solutions of  $X^t$  correspond to  $\varepsilon$ -solutions of  $C^t$ . For  $t = 0$ , the statement is clear. For  $t > 0$ , and assuming the statement for  $t - 1$ , it follows from the fact that the bank corresponding to the output of the  $t$ -th gate has no assets or liabilities in  $X^{t-1}$  and then from the definition of a financial system gadget and our above lemmas. By definition of the gadgets, each  $X^t$ , and thus  $X$ , is non-degenerate.  $\square$

*Remark 2.* The intermediate systems  $X^t$  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. However, since our proof does not rely on this assumption, not having it here does not lead to a problem. Alternatively, one could temporarily replace the output banks of gadgets that have not yet been executed 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. Such a financial system arises, for example, if we choose  $\delta = 1$  in Figure 3.

## 6.4 Discussion: Origin of the Computational Complexity

Given the results from this 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 terms of form  $r_j(1 - r_k)$  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. Note 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  $\varepsilon$ ) higher than the total notional of their contracts written and thus, they have recovery rate 1 in any  $\varepsilon$ -solution.
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.<sup>11</sup>

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 nonlinear 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 9** (Default Set). If  $r$  is a recovery rate vector, define the *default set* of  $r$  as

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

<sup>11</sup> 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.

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 4.** *Given  $\varepsilon \geq 0$ , a financial system without default costs with outside insurers, and the default set of any  $\varepsilon$ -solution, one can compute an  $\varepsilon$ -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  $\varepsilon$ ) and let  $M \subseteq N$  be the set of banks  $i$  for which  $e_i < (1 + \varepsilon)(\sum_j c_{i,j}^\emptyset + \sum_{j,k} c_{i,j}^k)$ . 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  $\varepsilon$ -solution  $r$  that

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

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  $j$ .

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

$$\begin{aligned} & \min \tilde{\varepsilon} \text{ s.t.} & (3) \\ & \tilde{\varepsilon} \geq 0 \\ \text{For all } i \in M : & 0 \leq r_i \leq 1 \\ \text{For all } i \in M \setminus D : & \\ & \frac{a_i(r_M)}{l_i} \geq 1 - \tilde{\varepsilon} \\ & r_i = 1 \\ \text{For all } i \in M \cap D : & \\ & \frac{a_i(r_M)}{l_i} \leq 1 + \tilde{\varepsilon} \\ & r_i = \frac{a_i(r_M)}{l_i} \pm \tilde{\varepsilon} \end{aligned}$$

One checks that any  $\varepsilon$ -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  $\varepsilon$ -solution of the financial system by setting the recovery rates of banks outside  $M$  to 1. The default set of this  $\varepsilon$ -solution may not be  $D$ , though.

Now, if  $D$  is the default set of some  $\varepsilon$ -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  $\varepsilon$ -solution.  $\square$

**Corollary 6.** *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  $\varepsilon$ -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 4 and the fact that our construction in the PPAD-hardness proof had the mentioned properties.  $\square$

**Corollary 7.** *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 4 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.

## 7 Conclusion

In this paper, we have studied the problem of computing clearing recovery rates in financial networks with debt and credit default swap (CDS) contracts. We have shown that the addition of CDSs makes many important decision problems NP-hard. We have further shown that the problem of computing an approximately clearing vector of recovery rates is PPAD-complete with CDSs even when the desired approximation quality is kept constant. Consequently, no polynomial-time approximation scheme exists unless  $P=PPAD$ . Regarding the origin of the additional complexity, we have shown that already computing the set of banks that default in an approximate solution is PPAD-complete.

Our results reveal two kinds of systemic risk in today’s financial networks that have gone unnoticed so far: first, in the event of a financial crisis, regulators might not even be able to tell which banks are in default. This could delay stabilizing measures, undermine trust in the market and thus exacerbate the crisis. Second, the risk of such a crisis happening is higher when regulators cannot conduct reliable stress tests to understand which banks are in danger of defaulting (and take targeted measures to protect them or isolate them from the rest of the system). Computational complexity is especially problematic in the latter case because stress tests are often conducted via Monte Carlo simulations and regulators will further want to conduct an array of them for different economic scenarios.

Our results can also be viewed from another, more positive angle: it is intuitive that derivatives make the financial system “more complex”. We have been able to make this statement precise in terms of computational complexity. Over the past ten years, researchers have sought to understand and quantify complexity in financial networks (Battiston et al., 2016b). In this paper, we have illustrated that besides the theory of dynamic systems and spectral graph theory, computational complexity is one of the ways by which this goal can be achieved. Our results tell us in a specific, precisely defined way in what sense financial systems with CDSs are complex: understanding the interactions between banks in financial systems with CDSs is at least as challenging as understanding the structure of Boolean circuits, as witnessed by the NP-complete Circuit Satisfiability and the PPAD-complete END OF THE LINE problems.

Our analysis in Section 6.4 shows that even very simple classes of financial systems can exhibit high computational complexity as long as banks are allowed to hold CDSs in a *naked* fashion, i.e., without also holding a corresponding debt contract from the reference entity. Note that all our gadgets use naked CDSs, and they also seem to require them. Given this, future work should investigate whether financial networks in which naked CDSs are banned admit a polynomial-time algorithm for the clearing problem, similar to debt-only networks.

Another important task for future research is to find algorithms for general financial networks with CDSs that may not have polynomial worst-case running time, but are fast in practice. These algorithms could work by successively updating the set of defaulting banks in a systematic fashion. All algorithms for realistic financial systems must in addition be able to deal with non-linearities in the function  $\frac{a_i}{l_i}$ .

## A Proofs from Section 5

*Proof of Lemma 2.* WLOG let  $C$  consist of NOR gates only. We model a Boolean circuit as a directed acyclic graph with two types of nodes: INPUT nodes have no predecessors. They encode inputs. There are exactly  $m$  input nodes. NOR nodes have two predecessors. They encode NOR gates. We assume some ordering on the nodes to identify different inputs.

In the following, we construct a financial system that encodes the circuit  $C$ . All nodes of the circuit have a corresponding bank in the financial system (that uses the same label). We also add other banks that do not correspond to nodes in the circuit. For technical reasons, we do not consider a special output node. Instead, we replace property 3 by the following, stronger property:

- (3a) Let  $k$  be any node in the circuit. For an assignment  $\chi \in \{0, 1\}^m$  let  $C_k(\chi)$  be the value of node  $k$  given inputs  $\chi$ . If  $r$  is clearing for  $X$ , then  $r_k = C(r_{a_1}, \dots, r_{a_m})$ .

We prove the theorem by induction on the number of nodes in the circuit. If there are no nodes, then the financial system with no banks has the required properties. So assume that there is at least one node.

Since the graph is acyclic, there is at least one node  $k$  with no successors. Let  $C'$  be the circuit without  $k$ . By induction hypothesis, there is a financial system  $X'$  that encodes  $C'$  in the sense of this theorem. We distinguish two cases based on the type of the node  $k$ :

*Case 1:*  $k$  is an INPUT node. Then  $k$  has no predecessors. Let  $X$  be the disjoint union of  $X'$  and a copy of the 0-1 system from Figure 3. Identify  $k$  with bank  $A$  in Figure 2. Let the input banks of  $X'$  be the input banks of  $X$  together with  $k$ . Since the two financial systems  $X'$  and the copy of Figure 2 do not interact, the solutions of the compound system  $X$  are the unions of the solutions of the two components, i.e.,

$$\{(r, 0, \dots) \mid r \text{ clearing for } X'\} \cup \{(r, 1, \dots) \mid r \text{ clearing for } X'\}$$

where “...” marks fixed, but unimportant values for the three new banks other than  $k$ . By induction hypothesis, all clearing recovery rates are 0 or 1, so property 2 holds and, since both 0 and 1 for the input  $k$  occur, also property 1 holds. Towards property 3a, let  $j$  be a node of  $C$  and let  $r$  be clearing. If  $j = k$ , then  $j$  is an input, so  $r_j = C_j(r_{a_1}, \dots, r_{a_m})$  trivially. If  $j \neq k$ , then  $r_j = C_j(r_{a_1}, \dots, r_{a_m})$  by induction hypothesis.

*Case 2:*  $k$  is a NOR node. Then  $k$  has two predecessors  $a$  and  $b$  in  $C'$ . Let  $X$  result by applying Lemma 1 to  $X'$ ,  $a$ , and  $b$ . This is allowed since by induction hypothesis, any clearing recovery rates of  $X'$  are 0 or 1. Identify  $k$  with bank  $v$  in the lemma. Let the input banks of  $X$  be the input banks of  $X'$ . The solutions of  $X$  are

$$\{(r, r_a \text{ NOR } r_b, \dots) \mid r \text{ clearing for } X'\}.$$

This implies that property 2 holds and property 1 holds by induction hypothesis because the input banks of  $X$  are the same as those of  $X'$ . Property 3a holds by induction hypothesis and Lemma 1.

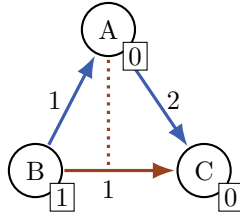
The construction adds exactly four banks in each step and there are as many steps as there are gates plus inputs, thus the size of  $X$  is linear in the size of  $C$ .  $\square$



---

**Figure 10** Financial System without default costs where the unique solution is irrational

---



*Proof of Theorem 2.* Reduction from Circuit Satisfiability. Given an instance  $C$  of Circuit Satisfiability, let  $X'$  result by application of Lemma 2 to  $C$  and let  $v$  be the output bank. Let  $X''$  be the system that has no solution, corresponding to Figure 2. There is a debt contract from  $B$  to  $A$ . Let  $X$  be the union of  $X'$  and  $X''$  where this debt contract has been replaced by a CDS with reference entity  $v$  and the same notional. We claim that  $X$  has a solution iff  $C$  has a satisfying assignment.

Like in the proof of Theorem 1, the solutions  $r$  of the  $X'$  component of  $X$  correspond to assignments  $(r_{a_1}, \dots, r_{a_m})$  and output values  $r_v = C(r_{a_1}, \dots, r_{a_m})$  of  $C$ . To see which of these extend to solutions for the whole of  $X$ , we distinguish two cases:

*Case 1:*  $r_v = 0$ . Then the CDS from  $B$  to  $A$  in the  $X''$  component of  $X$  gives rise to a liability equal to its full notional and thus solutions of this component that extend  $r$  correspond to solutions of  $X''$ , which do not exist. So no extension of  $r$  can be a solution for  $X$ .

*Case 2:*  $r_v = 1$ . Then the CDS from  $B$  to  $A$  in the  $X''$  component of  $X$  gives rise to a liability of 0. One now easily checks that  $r_A = e_A$ ,  $r_B = 1$ , and  $r_C = r_D = 1$  extend  $r$  to a solution on the whole of  $X$ .

Thus, the solutions of  $X$  correspond to the *satisfying* assignments  $(r_{a_1}, \dots, r_{a_m})$  and output values  $r_v = C(r_{a_1}, \dots, r_{a_m}) = 1$  of  $C$ .  $\square$

## B Irrational Solutions

*Example 1* (Irrational Solutions). Figure 10 shows a financial system the unique solution of which is irrational. To see this, note that by the contract structure  $r$  is clearing iff

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

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

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

## C 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  $\gamma$  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  $\varepsilon$  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  $\varepsilon$  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  $\varepsilon$  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  $\varepsilon$ -solutions where  $\varepsilon$  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 2** (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  $\varepsilon$ -solution is also an  $\varepsilon'$ -solution.
3.  $r$  is an  $\varepsilon$ -solution iff  $r$  is an  $\varepsilon'$ -solution for all  $\varepsilon' > \varepsilon$ .
4. Given  $r$  and  $\varepsilon$ , one can check in polynomial time if  $r$  is an  $\varepsilon$ -solution.

In prior work (Schuldenzucker, Seuken and Battiston, 2017a), we considered a simpler and weaker approximate solution concept. Specifically, we called  $r$  an  $\varepsilon$ -solution iff  $F(r) = r \pm \varepsilon$ . We then described an algorithm that computes such a *weak*  $\varepsilon$ -solution in a restricted case. It is easy to see that our algorithm from (Schuldenzucker, Seuken and Battiston, 2017a) in fact computes an  $\varepsilon$ -solution as defined in this paper and that our proof from this paper can be modified to show that already finding a *weak*  $\varepsilon$ -solution is PPAD-hard.

## D Proofs from Section 6.1

The following lemma lets us express  $\varepsilon$ -FINDCLEARING as the problem of finding an approximate fixed point of a certain Lipschitz continuous function. Then Lemma 3 follows using standard techniques.

**Lemma 12.** *Let  $X = (N, e, c)$  be a non-degenerate financial system without default costs and let  $\varepsilon > 0$ . Define the function*

$$G : [0, 1 + \varepsilon]^N \rightarrow [0, 1 + \varepsilon]^N$$

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

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

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

*Proof. Part 1:* It is sufficient to show that each  $G_i$  has an appropriate Lipschitz constant. So let  $i \in N$ . By non-degeneracy, 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  $G_i$  is constant  $1 + \varepsilon$ .

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

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

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

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

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  $G_i(s) = 1 + \varepsilon$  and further  $G_i(s) \rightarrow 1 + \varepsilon$  as  $l_i([s]) \rightarrow e_i$ . On  $L_i$ , one receives a Lipschitz constant for the restriction of  $\left[ \frac{a_i([s])}{l_i([s])} \right]^{1+\varepsilon}$  to  $L_i$  by applying the same reasoning as above with  $D_i := e_i^2$ . Thus,  $G_i$  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  $\max(1, \max_k \frac{N_k^i}{D_i})$ .

*Part 2:* Let  $s$  be an  $\varepsilon$ -approximate fixed point of  $G$ . Assume WLOG that  $l_i([s]) > 0$ . Let  $i \in N$  and let  $\tilde{s}_i := \frac{a_i([s])}{l_i([s])} \in [0, \infty)$ . We have  $G_i(s) = [\tilde{s}_i]^{1+\varepsilon}$  and  $s_i = G_i(s) \pm \varepsilon$  and thus

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

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  $\varepsilon$ -solution at  $i$ .  $\square$

*Proof of Lemma 3. Part 1:* Let  $X$  and  $\varepsilon$  be given and consider the function  $G$  from Lemma 12. Let  $K$  be the Lipschitz constant and recap that  $K$  is polynomial in  $X$  and  $\varepsilon$ . Since  $G$  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  $s'_i := \delta \lceil \delta^{-1} s_i \rceil$ . That is,  $s'_i$  is  $s_i$  rounded to multiples of  $\delta$ .  $s'$  has length  $n \cdot L$  where  $n = |N|$ ,  $L$  is the length of  $\delta$ , and  $L$  is polynomial in the lengths of  $X$  and  $\varepsilon$ .<sup>12</sup> Further,

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

Hence,  $s'$  is an  $\varepsilon$ -approximate fixed point of  $G$  and thus an  $\varepsilon$ -solution.

*Part 2:* Proof by reduction to the PPAD-complete generic BROUWER problem (Daskalakis, Goldberg and Papadimitriou, 2009):

Given  $n \in \mathbb{N}$ , an efficient algorithm for the evaluation of a function  $G : [0, 1]^n \rightarrow [0, 1]^n$ , a Lipschitz constant  $K$  for  $G$ , and an accuracy  $\varepsilon > 0$ , compute a point  $x$  such that  $\|G(x) - x\| \leq \varepsilon$ .

We apply the generic BROUWER problem to the function  $G$  from Lemma 12. It is easy to see 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  $G$  by a factor  $1 + \varepsilon$  and replacing  $\varepsilon$  by  $\frac{\varepsilon}{1+\varepsilon} \geq \frac{1}{2}\varepsilon$ ). Again by Lemma 12, we know that the output of the BROUWER problem gives rise to an  $\varepsilon$ -solution for  $X$ .  $\square$

## E Proofs from Section 6.2

*Proof of Lemma 4.* 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 proof of Lemma 3.

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

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

Note that  $f_{C_{>\zeta}}$  is monotonically increasing with  $f_{C_{>\zeta}}(\zeta - \varepsilon) = 0$  and  $f_{C_{>\zeta}}(\zeta + \varepsilon) = 1$ . All gate functions are Lipschitz with constant  $K := \max(2, \zeta_{\max}, \frac{1}{2\varepsilon})$  where  $\zeta_{\max}$  is the maximum  $\zeta$  such that  $C$  has a  $C_{\times\zeta}$  gate.

Let  $N = \{1, \dots, n\}$  be the set of nodes in the circuit. We define a function  $G : [0, 1]^n \rightarrow [0, 1]^n$ . For  $x \in [0, 1]^n$  and  $i \in N$  let  $G_i(x)$  be defined as follows:

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

<sup>12</sup>We assume here that numbers are encoded as fractions of binary integers. Alternatively, one could choose  $\delta$  to be the largest power of two  $\leq \frac{\varepsilon}{K+1}$ .

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

The first part of the lemma now follows just like in the proof of the first part of Lemma 3: if  $x$  is an exact fixed point of  $G$  and  $x'$  is  $x$  rounded to multiples of  $\delta := \frac{\varepsilon}{K+1}$ , then  $x'$  is an  $\varepsilon$ -approximate fixed point of  $G$  and thus an  $\varepsilon$ -solution of  $C$  and has polynomial length. It is not a problem that  $K$  depends on  $\varepsilon$ .

The second part of the lemma follows by reduction to the generic BROUWER problem just like in the proof of the second part of Lemma 3. This in fact proves that the weakly harder problem of computing an  $\varepsilon$ -solution where  $\varepsilon$  is not a parameter, but part of the input, is still in PPAD. It is again not a problem that  $K$  depends on  $\varepsilon$  because the generic BROUWER problem takes the Lipschitz constant as an input, just like  $\varepsilon$ .  $\square$

*Proof of Lemma 5.* Rubinstein (Rubinstein, 2015) proved that the following variant of the  $\varepsilon$ -GCIRCUIT problem is PPAD-hard for some  $\varepsilon$ :

1. Scaling is only allowed<sup>13</sup> by values  $\zeta \leq 1$  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  $\zeta \leq 1$ , then our  $C_{\times\zeta}$  gate has error  $(1 + \zeta)\varepsilon \leq 2\varepsilon$  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  $x[a_1] \leq x[a_2] - \varepsilon$  and  $x[v] = 1 \pm \varepsilon$  if  $x[a_1] \geq x[a_2] + \varepsilon$ .

We rewrite the expression  $x[a_1] < x[a_2]$  to use only comparison to a constant in a way that is robust against  $\varepsilon$  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  $\varepsilon'$ -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  $v$ .

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

$$\begin{aligned} x[a_1] \leq x[a_2] - \varepsilon &\Rightarrow \tilde{u} \leq \frac{1}{2} - \varepsilon = \frac{1}{2} - 4\varepsilon' - \varepsilon' \\ &\Rightarrow x[u] \leq \frac{1}{2} - \varepsilon' \\ &\Rightarrow x[v] = 0 \pm \varepsilon' = 0 \pm \varepsilon. \end{aligned}$$

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

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  $\varepsilon$ -solutions of  $C$ . This concludes the proof.  $\square$

<sup>13</sup>This assumption can be found in the full version of Rubinstein's paper (Rubinstein, 2016).

## References

- Acemoglu, Daron, Asuman Ozdaglar, and Alireza Tahbaz-Salehi.** 2015. “Systemic Risk and Stability in Financial Networks.” *American Economic Review*, 105(2): 564–608.
- Allen, Franklin, and Douglas Gale.** 2000. “Financial contagion.” *Journal of political economy*, 108(1): 1–33.
- Arora, Sanjeev, Boaz Barak, Markus Brunnermeier, and Rong Ge.** 2010. “Computational complexity and information asymmetry in financial products.” In *ICS*.
- Bank for International Settlements.** 2017. “Semianual OTC derivatives statistics, credit default swaps, by sector of reference, H1 2016.” <http://stats.bis.org/statx/srs/table/d10.4?p=20161>.
- Bardoscia, Marco, Stefano Battiston, Fabio Caccioli, and Guido Caldarelli.** 2017. “Pathways towards instability in financial networks.” *Nature Communications*, 8: 14416.
- Barucca, Paolo, Marco Bardoscia, Fabio Caccioli, Marco D’Errico, Gabriele Visentin, Stefano Battiston, and Guido Caldarelli.** 2016. “Network Valuation in Financial Systems.” Working Paper.
- Battiston, Stefano, Domenico Delli Gatti, Mauro Gallegati, Bruce Greenwald, and Joseph E Stiglitz.** 2012a. “Liaisons dangereuses: Increasing connectivity, risk sharing, and systemic risk.” *Journal of Economic Dynamics and Control*, 36(8): 1121–1141.
- Battiston, Stefano, Guido Caldarelli, Robert M May, Tarik Roukny, and Joseph E Stiglitz.** 2016a. “The price of complexity in financial networks.” *Proceedings of the National Academy of Sciences*, 113(36): 10031–10036.
- Battiston, Stefano, J. Doyne Farmer, Andreas Flache, Diego Garlaschelli, Andrew G. Haldane, Hans Heesterbeek, Cars Hommes, Carlo Jaeger, Robert May, and Marten Scheffer.** 2016b. “Complexity theory and financial regulation.” *Science*, 351(6275): 818–819.
- Battiston, Stefano, Michelangelo Puliga, Rahul Kaushik, Paolo Tasca, and Guido Caldarelli.** 2012b. “Debtrank: Too central to fail? Financial networks, the FED and systemic risk.” *Scientific reports*, 2.
- Braverman, Mark, and Kanika Pasricha.** 2014. “The computational hardness of pricing compound options.” In *Proceedings of the 5th conference on Innovations in theoretical computer science*.
- Chen, X., X. Deng, and S. h. Teng.** 2006. “Computing Nash Equilibria: Approximation and Smoothed Complexity.” In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS’06)*.
- Daskalakis, Constantinos.** 2013. “On the Complexity of Approximating a Nash Equilibrium.” *ACM Transactions on Algorithms (TALG)*. Special Issue for SODA 2011, Invited.
- Daskalakis, Constantinos, Paul W. Goldberg, and Christos H. Papadimitriou.** 2009. “The Complexity of Computing a Nash Equilibrium.” *Commun. ACM*, 52(2): 89–97.
- Daskalakis, Konstantinos, Paul W. Goldberg, and Christos H. Papadimitriou.** 2005. “The complexity of computing a Nash equilibrium.” *Electronic Colloquium on Computational Complexity (ECCC)*, , (115).
- Dees, Stéphane, Jérôme Henry, and Reiner Martin,** ed. 2017. *STAMP€: Stress-Test Analytics for Macroprudential Purposes in the euro area*. Frankfurt:European Central Bank.
- D’Errico, Marco, Stefano Battiston, Tuomas Peltonen, and Martin Scheicher.** 2016. “How does risk flow in the CDS market?” Working Paper.
- Eisenberg, Larry, and Thomas H Noe.** 2001. “Systemic risk in financial systems.” *Management Science*, 47(2): 236–249.
- Elliott, Matthew, Benjamin Golub, and Matthew O. Jackson.** 2014. “Financial Networks and Contagion.” *American Economic Review*, 104(10): 3115–53.
- Hemenway, Brett, and Sanjeev Khanna.** 2015. “Sensitivity and Computational Complexity in Financial Networks.” Working Paper.
- Papadimitriou, Christos H.** 1994. “On the complexity of the parity argument and other inefficient proofs of existence.” *Journal of Computer and System Sciences*, 48(3): 498 – 532.
- Rogers, LCG, and Luitgard AM Veraart.** 2013. “Failure and rescue in an interbank network.”

*Management Science*, 59(4): 882–898.

- Roukny, Tarik, Hugues Bersini, Hugues Pirotte, Guido Caldarelli, and Stefano Battiston.** 2013. “Default Cascades in Complex Networks: Topology and Systemic Risk.” *Sci. Rep.*, 3. Article.
- Rubinstein, Aviad.** 2015. “Inapproximability of Nash Equilibrium.” In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*. Portland, Oregon, USA:ACM.
- Rubinstein, Aviad.** 2016. “Inapproximability of Nash Equilibrium.” Working Paper.
- Schuldenzucker, Steffen, Sven Seuken, and Stefano Battiston.** 2016. “Clearing Payments in Financial Networks with Credit Default Swaps [Extended Abstract].” In *Proceedings of the 17th ACM Conference on Economics and Computation (EC)*. Maastricht, The Netherlands.
- Schuldenzucker, Steffen, Sven Seuken, and Stefano Battiston.** 2017a. “Default Ambiguity: Credit Default Swaps Create New Systemic Risks in Financial Networks.” Working Paper.
- Schuldenzucker, Steffen, Sven Seuken, and Stefano Battiston.** 2017b. “Finding Clearing Payments in Financial Networks with Credit Default Swaps is PPAD-complete.” In *Proceedings of the 8th Innovations in Theoretical Computer Science (ITCS) Conference*. Berkeley, USA.
- Stiglitz, Joseph E.** 2010. “Risk and Global Economic Architecture: Why Full Financial Integration May Be Undesirable.” National Bureau of Economic Research Working Paper 15718.
- Visentin, Gabriele, Stefano Battiston, and Marco D’Errico.** 2016. “Rethinking Financial Contagion.” Working Paper.
- Zuckerman, David.** 2011. “Pseudorandom Financial Derivatives.” In *Proceedings of the 12th ACM Conference on Electronic Commerce*. New York, NY, USA:ACM.