# Financial Networks with Singleton Liability Priorities

Stavros D. Ioannidis, Bart de Keijzer, and Carmine Ventre

King's College London

**Abstract.** Financial networks model debt obligations between economic firms. Computational and game-theoretic analyses of these networks have been recent focus of the literature. The main computational challenge in this context is the clearing problem, a fixed point search problem that essentially determines insolvent firms and their exposure to systemic risk, technically known as recovery rates. When Credit Default Swaps, a derivative connected to the 2008 financial crisis, are part of the obligations *and* insolvent firms pay the same proportion of all their debts, computing a weakly approximate solution is PPAD-complete [29], whereas computing a strongly approximate solution is FIXP-complete [17].
This paper addresses the computational complexity of the clearing problem in financial networks with derivatives, whenever priorities amongst creditors are adopted. This practically relevant model has been only studied from a game-theoretic standpoint. We explicitly study the clearing problem whenever the firms pay according to a singleton liability priority list and prove that it is FIXP-complete. Finally, we provide a host of NP-hardness results for the computation of priority lists that optimise specific objectives of importance in the domain.

## 1 Introduction

The financial services industry has been very creative, with the constant introduction of new products designed as investment and/or risk management tools. This makes the web of liabilities between the different institutions in the market hard to track and oversee. It is, in fact, this inherent complex structure of the evolving modern financial system that has led to several somewhat unforeseen and deeply damaging crises, such as the Great Financial Crisis (GFC) of 2008. There is, therefore, the need to mathematically model and study this network of obligations among interconnected financial agents in order to understand the impact of new products, regulations or even single contracts.

The main computational challenge in this context is the *clearing problem* introduced in [6]: Given the banks' funds and the face values of all the liabilities in the network, compute for each bank its exposure to systemic risk, in the form of what is known as its *clearing recovery rate*. It turns out that the complexity of this problem is closely related to the class of financial products populating the network. In fact, in financial networks where firms only subscribe simple debt contracts, clearing recovery rates can be computed in polynomial time [6,25]. Whilst this setup makes up for an easy modeling of networks as directed graphs with nodes standing for financial institutions and edges representing debt obligations, this representation is too simplistic in that it does not capture more advanced products, based on other existing contracts like mortgages, loans, interest rates etc. These complex contracts are called *derivatives*. The addition

of one such derivative, namely, *Credit Default Swaps (CDSes)*, introduced to financial networks in [28], makes the model more intriguing from a computational perspective. A CDS involves three parties $i, j, k$, where $k$ must pay an amount of money to $j$ on the condition that $i$ cannot pay off all of its obligations. Since adding CDSes to a financial network may generate irrational clearing recovery rates [28,17], our interest turns to finding approximate clearing recovery rates; it is proved that weakly (or "almost") approximate recovery rates are PPAD-complete to compute [29] and strong (or "near") approximate solutions are FIXP-complete to compute [17].

In this paper, we look at this problem from the perspective of the financial regulator. We ask whether rules that determine how insolvent banks pay off their debts can fundamentally change the computational hardness landscape above. The most-studied payment scheme is the *proportional payment scheme* where each bank pays off its debts proportionally to its recovery rate. However, defining priority classes amongst creditors and pay proportionally in each class (with funds available at the current priority level) is another widely adopted measure used in practice in the industry. For example, some regulatory regimes require employees to be prioritised over other creditors whereas some advanced derivatives (such as, the renown Collateralized Debt Obligations leading to the GFC) define their payoff via *tranching* (effectively a priority list) of the underlying securities. Whilst priority payments have been studied under a game-theoretic framework, see, e.g., [18,3,21], nothing is known about the computational complexity of the clearing problem with this payment scheme in presence of financial derivatives.

**Our Contribution.** We study the clearing problem in financial networks with derivatives under the *priority list* payment scheme. Specifically, we examine financial networks consisting of both debt and CDS contracts and address the complexity of computing a *clearing recovery rate vector* whenever each bank has to pay its debts in the following way. For each bank, we define a partition of its liabilities in priority classes. With the funds available at a certain priority (i.e., after having paid all the liabilities with higher priority), the liabilities at the current priority are paid proportionally – in particular, this means that these are paid in full if the funds are sufficient. This notion generalises the proportional payment scheme studied in related literature (i.e., consider the case in which the partition contains one part) and the class of *singleton priorities*, where each part is a singleton. We call this problem CDS-PRIORITY-CLEARING. Note that without CDSes, CDS-PRIORITY-CLEARING is known to be in P, both for proportional [6] and priority [18] payment schemes.

We observe here that whenever the partition defined by the priority list contains at least of part of size 2 or more then CDS-PRIORITY-CLEARING is FIXP-complete. FIXP [8] is a complexity class that captures the fixed point computations of total search problems. In our context, it is important to observe that FIXP is defined in terms of fixed-point functions defined upon the algebraic basis $A = \{+, -, *, /, \max, \min, \sqrt[k]{}\}$. It is not hard to see that the FIXP-completeness follows from the recent reduction given in [17] where all gadgets adopted have maximum out-degree 2. [1] Therefore, the only case left open is when all the parts of the priority list are singletons. We focus our attention on this setup in our work. We then call CDS-PRIORITY-CLEARING the problem

[1] We defer a more formal treatment of this case to the full version of the paper.

of finding recovery rates for singleton priorities, i.e., each bank has an ordering of its liabilities according to which its debts are paid off.

Our first contribution gives technical evidence that the financial regulator cannot change the complexity of computing clearing recovery rates, by enforcing priority payments. Specifically, we prove that CDS-PRIORITY-CLEARING is at least as hard as the *square root sum* (SQRT-SUM) problem [8,11,20,30] and that it is complete for FIXP. We then give the full picture of the complexity for the clearing problem, under all payment schemes proposed in the literature. Whilst the proof of FIXP-completeness adopts known approaches [17], our reduction introduces new financial network gadgets with priority payments for the operations in $A$. In particular, our new multiplication gadget highlights the flexibility of financial networks in handling arithmetic operations.

Whilst the regulator cannot ease the computational complexity of the problem, we wonder whether one can efficiently compute the banks' priority lists to optimise certain objective functions of financial interest. These include maximising the equity of a specific bank, maximising the liquidity in the system, and minimising the number of activated CDSes. As our second main contribution, we present a set of NP-hardness results showing an interesting parallel with the known hardness of computing similarly "optimal" solutions with proportional payments [22].

**Related work.** Clearing problems and mechanisms have been studied a lot in the literature [25,6,7,1,5,16,15,14,12]. Analysis of financial networks with CDSes as well as their properties is a popular topic in the area [23,21,24,27,29,28,17]. A game-theoretic approach to financial networks, the edge ranking game as well as other financial network games are listed in [18,19,3,22]. The FIXP-complexity class was defined first in [31,8,17], which established the FIXP-completeness of various fundamental fixed point computation problems. There are various further recent papers that show FIXP-completeness of a range of problems, including [17,13,10,9].

## 2 Model and Preliminaries

**Financial Networks and Payment schemes.** A financial network consists of a set of financial entities (which we refer to as *banks* for convenience), interconnected through a set of financial contracts. Let $N = \{1, \ldots, n\}$ be the set of $n$ banks. Each bank $i \in N$ has *external assets*, denoted by $e_i \in \mathbb{Q}_{\geq 0}$. We let $e = (e_1, \ldots, e_n)$ be the vector of all external assets. We consider two types of liabilities among banks: *debt contracts* and *credit default swaps (CDSes)*. A debt contract requires one bank $i$ (debtor) to pay another bank $j$ (creditor) a certain amount $c_{i,j} \in \mathbb{Q}_{\geq 0}$. A CDS requires a debtor $i$ to pay a creditor $j$ on condition that a third bank called the *reference bank* $R$ is in default, meaning that $R$ cannot fully pay its liabilities. Formally, we associate each bank $i$ a variable $r_i \in [0, 1]$, called the *recovery rate*, that indicates the proportion of liabilities it is able to pay. Having $r_i = 1$ means that bank $i$ can fully pay its liabilities, while $r_i < 1$ indicates that $i$ is in default. In case a reference bank $R$ of a CDS is in default, the debtor $i$ of that CDS pays the creditor $j$ an amount of $(1 - r_R)c^R_{i,j}$, where $c^R_{i,j} \in \mathbb{Q}_{\geq 0}$ is the face value of the CDS. The value $c_{i,j}$ ($c^R_{i,j}$, resp.) of a debt contract (CDS, resp.) is also called the *notional* of the contract. Finally, we let $c$ be the collection of all contracts'

notionals. We do not allow any bank to have a debt contract with itself, and assume that all three banks in any CDS are distinct.

The financial system $\mathcal{F}$ can therefore be represented as the triplet $(N, e, c)$. Given $\mathcal{F}$, we let $\mathcal{DC}_{\mathcal{F}}$ denote the set of all pairs of banks participating in a debt contract in $\mathcal{F}$. Similarly, $\mathcal{CDS}_{\mathcal{F}}$ denotes the set of all triplets participating in a CDS in $\mathcal{F}$. (We drop $\mathcal{F}$ from the notation when it is clear from the context.)

The *contract graph* of $\mathcal{F} = (N, e, c)$ is defined as a coloured directed multigraph $G_{\mathcal{F}} = (V, A)$, where $V = N$ and $A = (\cup_{k \in N} A_k) \cup A_0$ where $A_0 = \{(i, j) \mid i, j \in N \wedge c_{i,j} \neq 0\}$ and $A_k = \{(i, j) \mid i, j \in N \wedge c^k_{i,j} \neq 0\}$. Each arc $(i, j) \in A_0$ is coloured blue and each $(i, j) \in A_k$ orange. For all $(i, j, R) \in \mathcal{CDS}$ we draw a dotted orange line from node $R$ to arc $(i, j) \in A_R$, denoting that $R$ is the reference bank of the CDS between $i$ and $j$. [2] Finally, we label each arc with the notional of the corresponding contract, and each node with the external assets of the corresponding bank.

All banks are obliged to pay off their liabilities according to a set of prespecified rules, which we refer to as a payment scheme. If a bank has sufficient assets, then the payment scheme is trivial and prescribes to simply make payments that correspond exactly to each of the bank's liabilities. If there are insufficient assets, on the other hand, the payment scheme will determine for each of the outgoing contracts how much of it is paid off. The most studied payment scheme is the *proportional payment scheme*, where each bank $i$ submits an $r_i$ proportion of each liability, leaving a $(1 - r_i)$ fraction of each liability unpaid. In this paper we study payments resulting from another rule, called the *singleton liability priority lists* payment scheme. More specifically, given a financial system $\mathcal{F} = (N, e, c)$, for each bank $i$ we define a total order over the arcs going out of $i$ in $G_{\mathcal{F}}$. We denote the singleton liability priority list of $i$ as $P_i = (i_1 \mid i_2 \mid ... \mid i_{\text{outdeg}(i)})$, where $i_k$ stands for the $k$-th element in the order, or $k$th *priority* of node $i$, and $\text{outdeg}(i)$ denotes the out degree of node $i$ in $G_{\mathcal{F}}$. The payments under this scheme are now formed through an iterative process where each bank pays off its liabilities, one after the other, preserving the ordering in its priority list. We denote by $c_{i_k}$ the contract notional of the $i_k$th priority and denote by $\mathcal{P} = (P_1, \ldots, P_n)$ the profile of singleton liability priority lists. We denote a financial system $\mathcal{F}$ endowed with a singleton liability priority profile $\mathcal{P}$ as $(\mathcal{F}, \mathcal{P})$. The next example illustrates the model.

*Example 1.* The financial system of Fig. 1 consists of six banks, $N = \{1, 2, 3, 4, 5, 6\}$. Banks 2 and 5 have external assets $e_2 = e_5 = 1 - c$, for some constant $c \in (0, 1)$, while all other banks have zero external assets. The set of debt contracts is $\mathcal{DC} = \{(2, 3), (5, 4)\}$ and the set of CDS contracts is $\mathcal{CDS} = \{(2, 1, 5), (5, 6, 2)\}$. All contract notionals are set to 1. For example, $c_{2,3} = c^5_{2,1} = 1$. Node 2 has two candidate singleton liability priority lists, one is $P^1_2 = ((2, 3) \mid (2, 1, 5))$, where $2_1 = (2, 3)$ with contract notional $c_{2_1} = c_{2,1} = 1$ and $2_2 = (2, 1, 5)$ with contract notional $c_{2_2} = c^5_{2,1} = 1$. The other one is $P^2_2 = ((2, 1, 5) \mid (2, 3))$ where $2_1 = (2, 1, 5)$ with $c_{2_1} = c^5_{2,1} = 1$ and $2_2 = (2, 3)$ with $c_{2_2} = c_{2,3} = 1$. Symmetrically one can derive the lists for node 5.

We are interested in computing for a pair $(\mathcal{F}, \mathcal{P})$, for each bank $i$, the proportion of liabilities that it is able to pay. This proportion is captured by the *recovery rate*, mentioned earlier: For each bank $i$ we associate a variable $r_i \in [0, 1]$, that indicates the

[2] Strictly speaking, this means that $G_{\mathcal{F}}$ is a directed hypergraph with arcs of size 2 and 3.

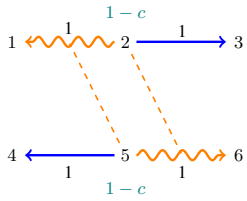



Fig. 1: Example of a financial network

proportion of liabilities that bank $i$ can pay. Recall that, to define the liability generated from a CDS contract, we need the recovery rate of the reference banks. Consequently in order to define all liabilities of banks in a financial system, we need to be presented with an a-priori recovery rate vector $r = (r_1, \cdots, r_n)$. So given a $(\mathcal{F}, \mathcal{P})$ and assuming a vector $r \in [0,1]^n$, we define the liabilities, the payments that each bank submits and the assets for each bank as follows.

We denote by $l_{i_k}(r)$ the $k$-th *liability priority* of node $i$. If $i_k = (i,j) \in \mathcal{DC}$ for some $j \in N$, then $l_{i_k}(r) = c_{i,j}$ and if $i_k = (i,j,R) \in \mathcal{CDS}$ for some $j, R \in N$, then $l_{i_j}(r) = (1 - r_R)c^R_{i,j}$. The liability of bank $i \in N$ to a bank $j \in N$ is denoted by $l_{i,j}(r)$ and it holds that

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

We denote by $l_i(r)$ the total liabilities of node $i$, and it holds that

$$l_i(r) = \sum_{j=1}^{\text{outdeg}(i)} l_{i_j}(r) = \sum_{j \neq i} l_{i,j}(r) = \sum_{j \in N \setminus \{i\}} \left( c_{i,j} + \sum_{k \in N \setminus \{i,j\}} (1 - r_k) c^k_{i,j} \right).$$

Node $i$ can fully pay its $k$th priority only if he has sufficient assets left after paying off the liabilities corresponding to priorities $i_1, \ldots, i_{k-1}$. We denote by $p_{i_k}(r)$ the payment of node $i$ to its $k$-th priority, and by $a_i(r)$ its assets, which are defined as the external assets it possesses plus all incoming payments submitted from its debtors (see below for a more formal definition). Under our singleton priority lists payment scheme:

$$p_{i_j}(r) = \max \left\{ 0, \min \left\{ l_{i_j}(r), a_i(r) - \sum_{j' < j} l_{i_{j'}}(r) \right\} \right\}. \tag{2}$$

Moreover, we denote by $p_{i,j}(r)$ the payment of node $i$ to node $j$ under recovery rate vector $r$: Let $\mathcal{C}_j = \{i_j \mid i_j \text{ is a contract with node } j \text{ as the creditor}\}$, then $p_{i,j}(r) = \sum_{i_j \in \mathcal{C}_j} p_{i_j}(r)$. The total payment made by a node is the sum of its individual payments to its priorities which is equal to the total sum of its payments to its creditors. Also, the payment of node $i$ needs to be equal to the proportion of its total liabilities it can pay off. Therefore, the following equations hold.

$$p_i(r) = \sum_{j=1}^{\text{outdeg}(i)} p_{i_j}(r) = \sum_{j \in N} p_{i,j}(r) = r_i l_i(r). \tag{3}$$

The assets of a bank $i$ are denoted as $a_i(r)$ and are the total amount of money it possesses summing its external assets and all incoming payments made by its debtors. It holds that

$$a_i(r) = e_i + \sum_{j \neq i} p_{j,i}(r). \tag{4}$$

We are interested in computing a specific recovery rate vector in $[0,1]^n$, such that (3) holds (i.e., $p_i(r) = r_i l_i(r)$ for all $i \in N$), under the singleton liability priority list payment rule (just defined formally by (2) and (4)). Formally:

**Definition 1 (Clearing recovery rate vector).** *Given a financial system and a singleton liability priority profile $(\mathcal{F}, \mathcal{P})$, a recovery rate vector $r$ is called clearing if and only if for all banks $i \in N$,*

$$r_i = \min\left\{1, \frac{a_i(r)}{l_i(r)}\right\} \text{ if } l_i(r) > 0, \text{ and } r_i = 1 \text{ if } l_i(r) = 0. \tag{5}$$

We illustrate the notions of the dynamics and the clearing recovery rate vectors (CRRVs) by reconsidering Example 1 and computing them for some priority profile $\mathcal{P}$.

*Example 1 (continued).* Let $c = 1/4$ in Fig. 1. Let $\mathcal{P} = (P_2 = ((2,3) \mid (2,1,5)), P_5 = ((5,4) \mid (5,6,2)))$. Both nodes 2 and 5 receive no payment from any other node thus their assets are defined as $a_2 = e_2 = 1 - c$ and $a_5 = e_5 = 1 - c$. For node 2, given $P_2$, we get that $l_{2_1} = l_{2,1} = c_{2_1} = c_{2,3} = 1$ and $l_{2_2} = l_{2,1} = (1 - r_5)c^5_{2,1} = (1 - r_5)$, thus the total liabilities for node 2 are $l_2 = l_{2_1} + l_{2_2} = 2 - r_5$. For node 5 we get that $l_{5_1} = l_{5,4} = c_{5_1} = c_{5,4} = 1$ and $l_{5_2} = l_{5,6} = c_{5_2} = (1 - r_2)c^2_{5,6} = 1 - r_2$, thus the total liabilities for node 5 are $l_5 = l_{5_1} + l_{5_2} = 2 - r_2$. Let us compute the CRRV. By (5) it must be $r_2 = \min\{1, a_2(r)/l_2(r)\} = \min\{1, 1 - c/2 - r_5\}$ and $r_5 = \min\{1, a_5(r)/l_5(r)\} = \min\{1, (1-c)/(2 - r_2)\}$. After solving this system we get that $r_2 = r_5 = 1 - \sqrt{c}$ and since we assumed $c = 1/4$ we finally get that $r_2 = r_5 = 1/2$. For the payments of node 2 , we know that $a_2 = 3/4$ and it first prioritises node 3 for which it has a liability of 1, thus it cannot fully pay off that liability and submits all of its assets to node 3, namely $p_{2_1} = p_{2,3} = 3/4$ and $p_{2_2} = p_{2,1} = 0$. The payments of node 5 are symmetrical.

**Our search problem.** We define CDS-PRIORITY-CLEARING to be the search problem that asks for a clearing recovery rate vector $r$ given a pair $(\mathcal{F}, \mathcal{P})$. The term CDS refers to the fact that $\mathcal{F}$ may contain CDS contracts (the problem becomes polynomial time computable without CDSes [18]) and the term PRIORITY indicates that banks pay according to singleton liability priority list $\mathcal{P}$. Similarly to [29,17], we assume that $\mathcal{F}$ is non-degenerate. We will discuss this assumption in more details below.

**Definition 2.** *A financial system is* non-degenerate *if and only if the following two conditions hold. Every debtor in a CDS either has positive external assets or is the debtor in at least one debt contract with a positive notional. Every bank that acts as a reference bank in some CDS is the debtor of at least one debt contract with a positive notional.*

Given an instance $I \in$ CDS-PRIORITY-CLEARING we transform (1) into a function defined on arbitrary recovery rate vectors $r = (r_1, \cdots, r_n)$ as:

$$f_I(r)_i = \frac{a_i(r)}{\max\{a_i(r), l_i(r)\}}.^{3} \tag{6}$$

From (6) we ascertain that $r$ is a clearing recovery rate vector for $I$ if and only if $r$ is a fixed point of $f_I$, namely $r = f_I(r)$. Thus, solving CDS-PRIORITY-CLEARING comes down to computing the fixed points of $f_I$. We define CDS-PRIORITY-CLEARING to contain only non-degenerate financial networks, for the analytical convenience that non-degeneracy provides (note that a division by 0 never occurs in $f_I(r)_i$ for these instances). It is not hard to see that $f_I$ has fixed points, see, e.g., [21]. Moreover, there exist instances of $(\mathcal{F}, \mathcal{P})$ that admit multiple CRRVs, see Appendix A

**Irrationality.** As is the case for the proportional payments, the singleton liability priority list model contains instances that admit irrational CRRVs. Observation 1 below provides such an example while Observations 2 and 3 present examples of how the priority profile affects the payments in the network. These examples also provide insights on an important difference between the two payment scheme: In the proportional model, whenever the CRRV is irrational then the clearing payment vector must be irrational as well. That is not the case in the singleton liability priority payment scheme, where irrationality of a CRRV need not cause any irrationality in the payments.

**Observation 1** *There exist instances of $(\mathcal{F}, \mathcal{P})$ that have irrational CRRVs. For instance, we know that $r_2 = r_5 = 1 - \sqrt{c}$ in Example 1. Thus, it is clear that for many choices of $c \in (0, 1)$ (e.g., $c = 1/2$) the CRRV is irrational.*

**Observation 2** *There exist a pair $(\mathcal{F}, \mathcal{P})$ with an irrational CRRV and irrational payments. Take again Example 1 and fix $c = 1/3$. We have $e_2 = e_5 = 2/3$, $l_{2,3} = l_{5,6} = 2/3$ and $r_2 = r_5 = 1 - \sqrt{1/3}$. Now consider the singleton liability priority lists $P_2 = ((2, 1, 5) \mid (2, 3))$ and $P_5 = ((5, 6, 2) \mid (5, 4))$. Since node 2 prioritises the $(2, 1, 5)$ contract, it has to pay an amount of $1 - \sqrt{1/3}$ to node 1. Given that its total assets are $2/3$, it can fully pay this liability and so $p_{2,1} = 1 - \sqrt{1/3}$ and what is left is being paid to node 3. Symmetrically, one can compute that $p_{5,6} = 1 - \sqrt{1/3}$.*

**Observation 3** *There exist a pair $(\mathcal{F}, \mathcal{P})$ with an irrational CRRV and rational payments. Consider Example 1 once more and fix $c = 1/2$. This yields $e_2 = e_5 = 1/2$, $l_{2,3} = l_{5,6} = 1/2$ and $r_2 = r_5 = 1 - \sqrt{1/2}$. Consider the singleton liability priority lists $P_2 = ((2, 1, 5) \mid (2, 3))$ and $P_5 = ((5, 6, 2) \mid (5, 4))$. Since node 2 prioritises the $(2, 1, 5)$ contract, it has to pay a amount of $1 - \sqrt{1/2}$ to node 1 but only possesses total assets of $1/2$. Thus it cannot fully pay this liability, meaning that $p_{2,1} = 1/2$. Symmetrically, we can compute that $p_{5,6} = 1/2$.*

**A primer on FIXP.** A useful framework for studying the complexity of fixed point computation problems is defined in [8]. Both exact and approximate computation of the solutions to such problems are considered. We begin by defining the notion of approximation we are interested in. Let $F$ be a continuous function that maps a compact convex set

[3] Strictly speaking, $f_I(r)_i$ is defined only for nodes $i$ that are not sinks in the contract graph. Sink nodes have recovery rate 1, cf. (5). Their exclusion simply allows to bypass potential divisions by 0 in $f_I$ (e.g., take node 1 in Fig. 1 when $c = 1$) while preserving its continuity. For notational simplicity, we will implicitly assume that we compute $f_I(r)_i$ iff $i$ is not a sink.

to itself and let $\epsilon > 0$ be a small constant. An *$\epsilon$-approximate fixed point* of $F$ is a point $x$ is within a distance $\epsilon$ near a fixed point of $F$, i.e., $\exists x' : F(x') = x' \wedge \|x' - x\|_\infty < \epsilon$. This notion is also known as *strong approximation*. [4] We now introduce the problems we are focusing on. A *fixed point problem* $\Pi$ is defined as a search problem such that for every instance $I \in \Pi$ there is an associated continuous function $F_I : D_I \to D_I$ —where $D_I \subseteq \mathbb{R}^n$ (for some $n \in \mathbb{N}$) is a convex polytope described by a set of linear inequalities with rational coefficients that can be computed from $I$ in polynomial time— such that the solutions of $I$ are the fixed points of $F_I$.

**Definition 3.** *The class* FIXP *consists of all fixed point problems $\Pi$ for which for all $I \in \Pi$ the function $F_I : D_I \to D_I$ can be represented by an algebraic circuit $C_I$ over the basis $\{+, -, *, /, \max, \min, \sqrt[k]{}\}$, using rational constants, such that $C_I$ computes $F_I$, and $C_I$ can be constructed from $I$ in time polynomial in $|I|$.*

*The class* FIXP$_a$ *is defined as the class of search problems that are the strong approximation version of some fixed point problem that belongs to* FIXP*.*

*The class* Linear-FIXP *is defined analogously to* FIXP*, but under the smaller arithmetic basis where only the gates $\{+, -, \max, \min\}$ and multiplication by rational constants are used.*

These classes admit complete problems. The completeness results in [8] in fact show that it is without loss of generality to consider a restricted basis $\{+, *, \max\}$ ($\{+, \max\}$) for FIXP (Linear-FIXP), and to assume that $D_I = [0, 1]^n$.[5] Hardness of a search problem $\Pi$ for FIXP is defined through the existence of a polynomial time computable function $\rho : \Pi' \to \Pi$, for all $\Pi \in$ FIXP, such that the solutions of $I$ can be obtained from the solutions of $\rho(I)$ by applying a (polynomial-time computable) linear transformation on a subset of $\rho(I)$'s coordinates. This type of reduction is known as a *polynomial time SL-reduction*.

It is known that FIXP$_a \subseteq$ PSPACE and Linear-FIXP $=$ PPAD [8]. An informal understanding of the hardness of FIXP vis-a-vis PPAD is as follows. PPAD captures a type of computational hardness stemming from an essentially combinatorial source. FIXP introduces on top of that a type of numerical hardness that emerges from the introduction of multiplication and division operations: These operations give rise to irrationality in the exact solutions to these problems, and may independently also require the computation of rational numbers of very high precision or very high magnitude.

## 3 Hardness of CDS-PRIORITY-CLEARING

We are interested in identifying the complexity of CDS-PRIORITY-CLEARING. Recall that in Example 1, we presented an instance which under proper coefficients admits only irrational clearing recovery rate vectors, which means that either one should study this problem with respect to complexity classes compatible real-valued solutions, or

[4] A *weak $\epsilon$-approximate fixed point* of a continuous function $F$ is a point $x$ such that its image is within distance $\epsilon$ of $x$, i.e., $\|x - F(x)\|_\infty < \epsilon$. Under *polynomial continuity*, a mild condition on the fixed point problem under consideration, a strong approximation is also weak [8].

[5] We will make use of these facts in the proof of Theorem 1, below.

one should redefine the goal of the problem along the lines of finding a rational-valued approximation to a potentially irrational solution.

As an initial step, we first show that determining whether $r_i < 1$ for a specific bank $i$ is at least as hard as solving the *square root sum* (SQRT-SUM) problem. An instance of SQRT-SUM consists of $n+1$ integers $d_1, d_2, ..., d_n, k$ and asks whether $\sum_{i=1}^{n} \sqrt{d_i} \leq k$. It is known that SQRT-SUM is solvable in PSPACE but it is unknown whether it is in P, or even in NP. In [30] it is shown that it can be solved in polynomial time in the unit-cost RAM model [30,26,2].

**Lemma 1.** *For a given pair $(\mathcal{F}, \mathcal{P})$, deciding whether a specific bank is in default is* SQRT-SUM*-hard.*

*Proof.* We prove the lemma by reducing CDS-PRIORITY-CLEARING to SQRT-SUM. Let $(d_1, ..d_n, k)$ be an instance of SQRT-SUM. Firstly, we note that in [4] it is shown that checking whether $\sum_{i=1}^{n} \sqrt{d_i} = k$ can be done in polynomial time. We check whether equality holds for our input first and proceed without loss of generality to the proof without minding equality.

We construct a financial system $\mathcal{F}$ as follows, first we construct $n$ financial subnetworks which we refer to as *square root gadgets* and denote by $g_{i,\sqrt{\cdot}}$ the $i$th square root gadget. Whenever referring to a node $k$, belonging in a square root gadget $g_{i,\sqrt{\cdot}}$, we use the notation $k_i$. The square root gadget $g_{i,\sqrt{\cdot}}$ consists of the financial network we presented in Fig. 1, augmented with two additional nodes $x_i, y_i$, and the CDS contract $(x_i, y_i, 2_i)$. We let the external assets of nodes $2_i$ and $5_i$ be $e_{2_i} = e_{5_i} = 1 - d_i$. We let $e_{x_i} = 1$ and $e_{y_i} = 0$ and the CDS contract $(x_i, y_i, 2_i)$ has a notional of 1: $c^{2_i}_{x_i,y_i} = 1$. The $n$ square root gadgets are all connected to a single node $\tau$ with $e_\tau = 0$, by $n$ debt contracts $\{(y_i, \tau) : i \in [n]\}$. There is one further node $\tau'$ to which $\tau$ is connected through debt contract $(\tau, \tau')$ with notional $c_{\tau,\tau'} = k$. The construction is illustrated in Fig. 2.

We claim that this resulting financial system has a clearing recovery rate vector $r$ with $r_\tau = 1$ if and only if $\sum_{i=1}^{n} \sqrt{d_i} \geq k$. From the analysis of Example 1, it follows that under any clearing recovery rate vector $r$, the recovery rate of node $2_i$ is $r_{2_i} = 1 - \sqrt{d_i}$ for all $i \in [n]$. Since node $2_i$ is always in default(assuming all $d_i \neq 0$) the CDS $(x_i, y_i, 2_i)$ is activated and since node $x_i$ can fully pay its liabilities, node $y_i$ receives a payment of $1 - r_{2_i} = \sqrt{d_i}$. This implies that $\tau$ receives a total payment of $\sum_{i\in[n]} \sqrt{d_i}$. Since $\tau$ has only one liability of $k$, it holds that $r_\tau = 1$ if and only if the total payment that $\tau$ receives exceeds $k$, i.e., if and only if $\sum_{i\in[n]} \sqrt{d_i} \geq k$, and this proves the claim. □

Next we show that CDS-PRIORITY-CLEARING and its strong approximation variant are FIXP and FIXP$_a$ complete, respectively. Our hardness reduction does not start from a particular FIXP-hard problem. Rather, we show that we can take an arbitrary algebraic circuit and encode it in a direct way in the form of a financial system accompanied by a specific singleton liability profile. Hence, our polynomial time hardness reduction works from any arbitrary fixed point problem in FIXP. The reduction is constructed by devising various financial network gadgets which enforce that certain banks in the system have recovery rates that are the result of applying one of the operators in

(a) The $i$ square root gadget $g_{i,\sqrt{\cdot}}$ (b) The constructed financial system

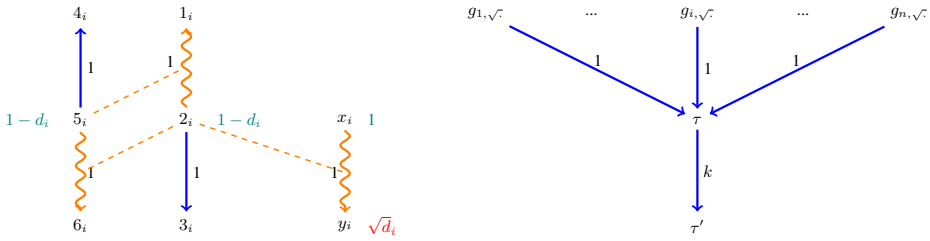



Fig. 2: The financial system constructed from a given Square Root Sum instance

FIXP's arithmetic base to the recovery rates of two other banks in the system: In other words, we can design our financial systems and singleton liability priorities such that the interrelation between the recovery rates mimics a computation through an arbitrary algebraic circuit.

**Theorem 1.** CDS-PRIORITY-CLEARING *is FIXP-complete, and its strong approximation version is FIXP$_a$-complete.*

*Proof (sketch).* Containment of CDS-PRIORITY-CLEARING in FIXP is immediate: The clearing vectors for an instance $I = (N, e, c) \in$ CDS-PRIORITY-CLEARING are the fixed points of the function $f_I$ defined coordinate-wise by

$$f_I(r)_i = \frac{a_i(r)}{\max\{l_i(r), a_i(r)\}}$$

as in (6). The functions $l_i(r)$ and $a_i(r)$ are defined in (1) and (4), from which it is clear that $f_I$ can be computed using a polynomial size algebraic circuit with only $\{\max, +, *\}$, and rational constants. Note that non-degeneracy of $I$ prevents division by 0, so that the output of the circuit is well-defined for every $x \in [0,1]^n$.This shows that CDS-PRIORITY-CLEARING is in FIXP and that its strong approximation version is in FIXP$_a$.

For the FIXP-hardness of the problem, let $\Pi$ be an arbitrary problem in FIXP. We describe a polynomial-time reduction from $\Pi$ to CDS-PRIORITY-CLEARING. Let $I \in \Pi$ be an instance, let $F_I : [0,1]^n \to [0,1]^n$ be $I$'s associated fixed point function, and let $C_I$ be the algebraic circuit corresponding to $F_I$. Our reduction is analogous to the proof in [17], where the variant of the problem with proportional payments is shown to be FIXP-hard: The proof consists of a "pre-processing" step (in which the algebraic circuit is transformed into a specific form) followed by a transformation into a financial network, where a set of financial subnetwork gadgets are interconnected, and each such gadget corresponds to an arithmetic gate in the algebraic circuit. The pre-processing step of this reduction is entirely equal to that of [17]: This step transforms $C_I$ into a circuit $C'_I$ that satisfies that all the signals propagated by all gates in $C'_I$ and all the used rational constants in $C'_I$ are contained in the interval $[0,1]$. The transformed circuit $C'_I$ may contain two additional type of gates: Division gates and gates that compute the

absolute value of the difference of two operands. We will refer to the latter type of gate as an *absolute difference gate*. The circuit $C'_I$ does not contain any subtraction gates, and will not contain $\max$ and $\min$ gates either. The reader interested in the details of this pre-processing step is referred to [17].

The second part of the proof (the transformation step) differs from [17] in that a different set of gadgets needs to be used. For notational convenience, we may treat $C'_I$ as the function $F_I$, hence we may write $C'_I(x) = y$ to denote $F_I(x) = y$.

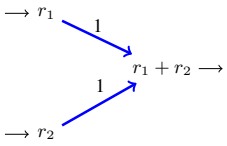
Each of our gadgets has one or two *input banks* that correspond to the input signals of one of the types of arithmetic gate, and there is an *output bank* that corresponds to the output signal of the gate. For each of the gadgets, it holds that the output bank must have a recovery rate that equals the result of applying the respective arithmetic operation on the recovery rates of the input banks. Each gadget represents an arithmetic gate of the circuit, and the output banks of a gadget are connected to input banks of other gadgets such that there is a direct correspondence with the infrastructure of the arithmetic circuit. For precise details on this correspondence, we refer the reader to the proof in [17].

For defining our gadgets, we use our graphical notation for convenience. As stated, our gadgets each have one or two input banks, and one output bank, where the output bank is forced to have a clearing recovery rate that is obtained by applying a arithmetic operation on the recovery rate of the input banks, so as to simulate the arithmetic basis on which the algebraic circuit $C'_I$ is built.

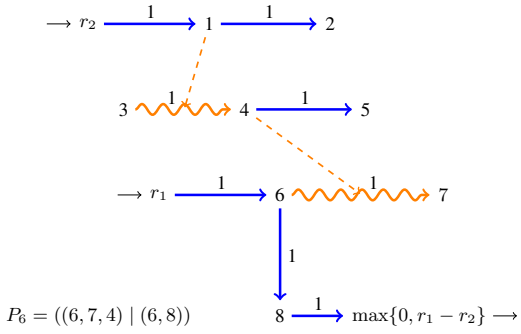
As a simple example of one of these gadgets we define a straightforward addition gadget, named $g_+$, depicted in Fig. 3a. This gadget directly accounts for the addition operation in the arithmetic basis. In our figures, input banks are denoted by black arrows incoming from the left, and output banks correspond to black arrows pointing out of the bank. The black arrows represent connections to other gadgets, and these connections are always realised by a debt contract of unit cost, and are always from the output node of a gadget to an input node of another gadget. Another slightly more complex gadget example is the *positive subtraction* gadget, displayed in Fig. 3b, taking two inputs $r_1, r_2$ and producing as output the value $\max\{0, r_1 - r_2\}$. This gadget is in turn used to form the absolute difference gadget, which can be constructed by combining two positive subtraction gadgets with an addition gadget (as $|r_1 - r_2| = \max\{0, r_1 - r_2\} + \max\{0, r_2 - r_1\}$). It is also used in the construction of our multiplication gadget.

In the figures representing our gadgets, some of the nodes have been annotated with a formula in terms of the recovery rates of the input banks of the gadget, subject to the resulting values being in the interval $[0, 1]$. This can be seen for example in the output node of our addition gadget in Fig. 3a. Such a formula represents the value that a clearing recovery rate must satisfy for the respective node. It is straightforward to verify that the given formulas are correct for each of our gadgets.

Since all signals inside $C'_I$ are guaranteed to be in $[0, 1]$ for all input vectors, our financial system gadgets can readily be used and connected to each other to construct a financial system that corresponds to $C'_I$, i.e., such that the clearing recovery rates of the input and output banks of each of the gadgets must correspond to each of the signals inside the circuit $C'_I$.




(a) Addition gadget $g_+$.




(b) Positive subtraction gadget $g_{\text{pos}-}$. Generates the value $\max\{0, r_1 - r_2\}$ from the input recovery rates $r_1$ and $r_2$. In this gadget node 6 pays according to the priority list $P_6 = ((6,7,4) \mid (6,8))$ thus it submits to node 8 the amount of $r_1 - r_2$ in case $r_1 > r_2$ and 0 otherwise

Fig. 3: Example gadgets from our reduction.

The remaining gadgets are displayed in Fig. 11 to 17, in Appendix C. Some of our gadgets are composed of auxiliary gadgets, where in particular, the construction of the division gadget is rather involved, and a separate discussion of how this gadget is constructed is provided in Appendix C.1. Another notably non-trivial gadget is the multiplication gadget, as can be seen in Fig. 16, where the main challenge in its construction is to ensure that non-degeneracy holds and that the expressions for the intermediate recovery rates inside the gadget are all contained in $[0, 1]$. □

## 4 Hardness of Deciding the Best Priority Profile

In a financial network, each node $i$ can be assigned one of outdeg$(i)$! singleton liability priority lists. Consequently the number of candidate priority profiles for a system is exponentially large in terms of its input size. Next, we consider the setting where a financial authority is able to determine which priority list each bank should get assigned, and is interested in assigning these in such a way that a specific objective is optimised for. We show that this problem is NP-hard for a set of natural choices of objective functions:

1. Minimising the number of defaulting nodes.
2. Minimising the number of not fully paid liabilities.
3. Minimising the number of activated CDSes in the financial system.
4. Maximising the equity of a specific node.
5. Maximising the liquidity in the financial system.

We defer the proofs of Statements 3 and 5 to Appendix B.

**Lemma 2.** *Finding a priority list profile that minimises the number of defaulting banks and finding a priority list profile that minimises the number of not fully paid liabilities are both NP-hard problems.*

*Proof.* We prove the lemma via a reduction from the satisfiability problem (SAT), where we are given a boolean formula in conjunctive normal form, and have to determine whether there is a truth assignment to the variables that renders the formula true. Let $F = \bigwedge_{i=1}^{n} C_i$ be a SAT instance, where $C_1, \ldots, C_n$ are the clauses, and let $V_F$ be the set of all variables that in $F$. We create a financial network from $F$ as follows. For each variable $x \in V_F$ we construct a gadget that is refereed to as the $x$-subnetwork. Each $x$-subnetwork consists of four nodes, labeled as $i_x, x, \neg x, j_x$, where $e_{i_x} = 1$ and $e_x = e_{\neg x} = j_x = 0$, and of four debt contracts, $\mathcal{DC} = \{(i_x, x), (i_x, \neg x), (x, j_x), (\neg x, j_x)\}$ all with contract notionals equal to zero. Moreover, the constructed financial network has $n$ further nodes, labeled as $C_1, \ldots, C_n$, that correspond to each clause in $F$ with $e_{C_1} = \cdots = e_{C_n} = 0$, and one terminal node labeled as $\tau$, towards which each $C_i$ holds a debt contract of notional one, i.e $c_{C_i,\tau} = 1$. Finally for each variable $x$ of $F$, we construct two nodes labeled as $k_x$ and $k_{\neg x}$ respectively, where $e_{k_x}$ and $e_{k_{\neg x}}$ are equal to the number of occurrences of literals $x$ and $\neg x$ in $F$, respectively. Whenever a literal $l$ belongs to a clause $C_i$ we construct the CDS $(k_l, C_i, \neg l)$ with $c^{\neg l}_{k_l,C_i} = 1$. An example of a network induced from $F = (x \vee y) \wedge (y \vee \neg y)$ is given in Fig. 4. We now map a truth assignment $\mathrm{T} : V_F \mapsto \{\text{true}, \text{false}\}$ to a priority list profile $\mathcal{P}_\mathrm{T}$ as follows: If $\mathrm{T}(x) = \text{true}$, node $i_x$ prioritises node $x$, and otherwise it prioritises node $\neg x$. All $k_l$ nodes posses enough external assets to fully pay their debts under any priority list, so we can take an arbitrary list for those nodes, and all remaining nodes have at most one liability. Conversely, from a priority list profile $\mathcal{P}$ we induce the truth assignment $\mathrm{T}_\mathcal{P}$ as follows: if $i_x$ prioritises $x$ then $\mathrm{T}(x) = \text{true}$ otherwise $\mathrm{T}(x) = \text{false}$.

Let $\mathrm{T} : V_F \mapsto \{\text{true}, \text{false}\}$ be any truth assignment and consider the priority list profile $\mathcal{P}_\mathrm{T}$. In each $x$-subnetwork, node $i_x$ can fully pay only its first priority, thus in each $x$-subnetwork there exist two defaulting nodes regardless of the choice of priority list of $i_x$, meaning that the minimum number of defaulting nodes in the induced financial system is $2|V_F|$. Similarly, each $x$-subnetwork has two not fully paid liabilities regardless of the choice of priority lost of $i_x$. The only additional defaulting nodes might be the nodes $C_1, \ldots, C_n$, and the only additional not fully paid liabilities might be on the $n$ debt contracts in which one of $C_1, \ldots, C_n$ is the debtor. Let us inspect which of the latter set of nodes are defaulting and which of the latter liabilities are not fully paid under $\mathcal{P}_\mathrm{T}$. If clause $C_i$ is a clause that is not satisfied under T, then none of the CDSes involving node $C_i$ are activated, and node $C_i$ does not have any assets $\mathcal{P}_\mathrm{T}$. Since $C_i$ has to pay 1 to $\tau$, node $C_i$ will be in default, and $C_i$'s liability of 1 will not be paid. If clause $C_i$ is a clause that is satisfied under T, then at least one CDS involving node $C_i$ is activated, and since the reference bank in this CDS has recovery rate 0, node $C_i$ will receive the CDS's full notional of 1, with which it can fully pay its liability of 1. Hence, in the latter case, node $C_i$ is not in default.

Hence, for a truth assignment T, under $\mathcal{P}_\mathrm{T}$, the number of banks in default and the number of not fully paid liabilities is are both equal to $2|V_F|$ plus the number of

unsatisfied clauses. Since we argued above that restricting to the profiles

$$\{\mathcal{P}_{\mathrm{T}} \mid \mathrm{T} \text{ is a truth-assignment for } F\}$$

is without loss of generality, from finding the profile of priority lists minimising the number of defaulting banks or minimising the number of not fully paid liabilities in the constructed financial network, one can infer whether the formula $F$ is satisfiable, which proves our claim. □

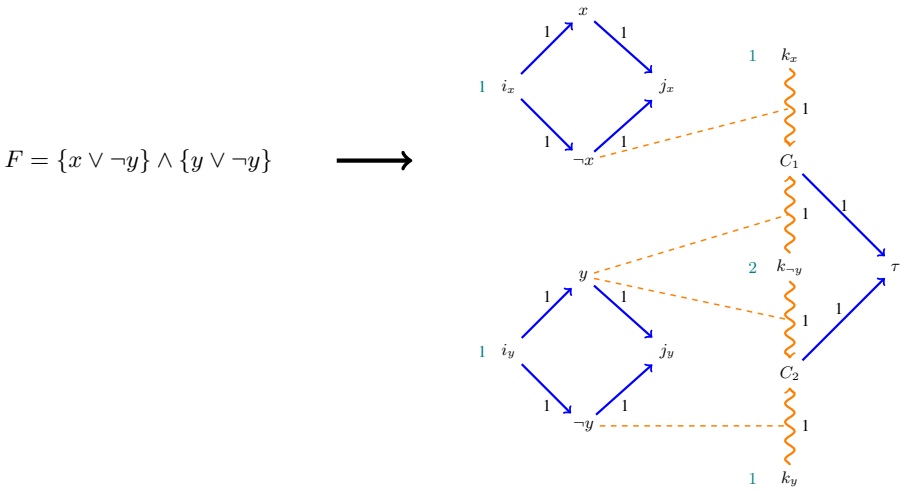



Fig. 4: The financial system corresponding to the formula $F = \{x \vee \neg y\} \wedge \{y \vee \neg y\}$.

**Lemma 3.** *Finding a priority list profile that maximises the equity of a specific node is NP-hard.*

*Proof.* We prove the lemma via a reduction from KNAPSACK. Let us be given a knapsack instance with a knapsack of capacity $B$, a set $S = \{a_1, \ldots, a_n\}$ of objects having profit $\text{profit}(a_i)$ and size $\text{size}(a_i)$. Without loss of generality, we assume that $\text{size}(a_i)$, for all $a_i \in S$ as well as $B$ are integer numbers. We construct a financial network $\mathcal{F} = (N, e, c)$ as follows: We introduce a node $0$ with external assets $e_i = B$ and for each object $a_j \in S$ we introduce a corresponding node $j$ and let node $0$ hold a debt contract towards each node $j$ with notional $c_{0,j} = \text{size}(a_j)$. Moreover we introduce a node $\tau$ with $e_\tau = 0$, and a node $T$ with $e_T = 0$, which we refer to as the *terminal node*. We add a debt contract of notional $\text{size}(a_j)$ from each node $j$ to node $\tau$. For each node $j$ we moreover introduce a $j$*-subnetwork*, consisting of:

- two nodes $y_j$ and $x_j$ with $e_{y_j} = 1, e_{x_j} = 0$,
- a CDS contract $(y_j, x_j, j)$ with notional $c^j_{y_j,x_j} = \max_{a_j \in S}\{size(a_j)\}$,
- a node $z_j$ towards which $x_j$ holds a debt contract of notional $c_{x_j,z_j} = 1$.

– a node $k_j$ with $e_{k_j} = \text{profit}(a_j)$ and an outgoing CDS $(k_j, T, x_j)$ with notional profit$(a_j)$.

The construction of the $j$-subnetwork is illustrated in Fig. 5.

Assume an optimal solution to the original Knapsack instance and let OPT be the set of the objects $a_j$ contained in it. We know that $\sum_{a_i \in \text{OPT}} \text{size}(a_i) \leq B$ and that $\sum_{a_i \in \text{OPT}} \text{profit}(a_i)$, is the maximum profit that can fit in the knapsack. We define the set $N = \{1, \ldots, n\}$, and $N_{\text{OPT}} = \{j \mid a_j \in \text{OPT}\}$ to be the set containing all nodes in $\mathcal{F}$ that correspond to objects contained in the optimal solution. Fix the singleton liability priority list for node $i$ to be $\mathcal{P}_i = (\{N_{\text{OPT}}\} \mid \{N \setminus N_{\text{OPT}}\})$, meaning that $i$ first prioritises all creditors in $N_{\text{OPT}}$ in an arbitrary order and afterwards all other creditors in $N \setminus N_{\text{OPT}}$ again in an arbitrary order. Next we prove that under this profile, node $T$ receives its maximum total assets. Observe that $\forall j \in N_{\text{OPT}}$, $p_{0j} = \text{size}(a_j)$ since $\sum_{j \in N_{\text{OPT}}} c_{0j} = \sum_{a_j \in \text{OPT}} \text{size}(a_j) \leq B = e_0$. Since every $j$ node that corresponds to an object $a_j \in \text{OPT}$ receives size$(a_j)$, it can fully pay node $\tau$, so $r_j = 1$. For all creditors $m \in N \setminus N_{\text{OPT}}$ it holds that $p_{0m} < \text{size}(a_m)$ So, $\forall j \in N_{\text{OPT}} : r_j = 1$ while $\forall m \in N \setminus N_{\text{OPT}} : r_m < 1$. Next we prove that for each $j \in N_{\text{OPT}}$, $p_{k_j,T} = \text{profit}(a_j)$ and for each $m \in N \setminus N_{\text{OPT}}$, $p_{k_m,T} = 0$. Take a $j \in N_{\text{OPT}}$, we know that $r_j = 1$, which implies that the CDS $(y_j, x_j, j)$ is not activated thus $r_{x_j} = 0$ which in turn activates the CDS $(k_j, \text{T}, x_j)$ where node $k_j$ pays profit$(a_j)$ to node $T$. On the other side, for a $m \in N \setminus N_{\text{OPT}}$, it holds that $r_m < 1$, which means that the CDS $(y_m, x_m, m)$ is activated and generates a liability of $\max_{a_i \in S}\{\text{size}(a_i)\} \cdot (1 - r_m)$ for node $y_m$. We prove that this liability is at least 1: For an object $a_m \notin \text{OPT}$, $r_m$ indicates the proportion of size$(a_m)$ that fits in the available knapsack area unoccupied by the objects in OPT. Obviously for $a_m \notin \text{OPT}$, size$(a_m) > B - \text{size(OPT)}$, otherwise $a_m \in \text{OPT}$ and $r_m \cdot \text{size}(a_m) + \text{size(OPT)} = B$. Since by assumption $B$ and size$(a_j)$ for all $a_j \in S$ are integers, it holds that $\forall a_m \notin \text{OPT}$, $r_m \leq (\max_{a_k \in S}\{\text{size}(a_k)\} - 1)/(\max_{a_k \in S}\{\text{size}(a_k)\}$, so the generated liability for $y_m$ is:

$$
\begin{aligned}
l^m_{y_m,x_m} &= \max_{a_k \in S}\{\text{size}(a_k)\}(1 - r_m) \\
&\geq \max_{a_k \in S}\{\text{size}(a_k)\} \left(1 - \frac{\max_{a_k \in S}\{\text{size}(a_k)\} - 1}{\max_{a_k \in S}\{\text{size}(a_k)\}}\right) = 1.
\end{aligned}
$$

So eventually, $\forall m \in N \setminus N_{\text{OPT}}$, $p_{y_m,x_m} = 1$. Now $r_{x_m} = 1$ thus the CDS $(k_m, \text{T}, x_m)$ is not activated meaning that $p_{k_m,T} = 0$. From the above observations we conclude that the equity of $T$ is $\sum_{j \in N_{\text{OPT}}} \text{profit}(a_j)$: node $T$ receives money from all nodes that correspond to objects contained in OPT. We claim that this is the maximum equity node $T$ can receive. If there exist a higher equity for $T$, then this must be generated from another profile $\mathcal{P}'_i$ that corresponds to a solution to the original Knapsack instance with higher profit than the optimal one which is a contradiction.

For the opposite direction assume $\mathcal{P}_0$ to be the profile of 0 that maximises $T$'s equity. Let $A = \{a_j \mid p_{0j} = \text{size}(a_j)\}$ be the set of objects that corresponds to creditor nodes that 0 can fully pay. Obviously $A$ can be computed in polynomial time from $\mathcal{P}_0$. We claim that $A$ is the optimal solution to the original Knapsack instance. Assume that there exists another set $A'$ such that $\sum_{a_j \in A'} \text{size}(a_j) \leq B$ and $\sum_{a_j \in A'} \text{profit}(a_j) > \sum_{a_j \in A} \text{profit}(a_j)$. Now node 0 could rearrange its priorities by prioritising all cred-

itors $j$ for which $a_j \in A'$. Doing so, 0 can fully pay all nodes $j$ for which $a_j \in A'$ since $\sum_{a_j \in A'} \text{size}(a_j) \leq B = e_0$ and node $T$ receives $\sum_{a_j \in A'} \text{profit}(a_j) > \sum_{a_j \in A} \text{profit}(a_j)$, a contradiction to the original assumption that $\sum_{a_j \in A} \text{profit}(a_j)$ is the maximum equity for node $T$. □

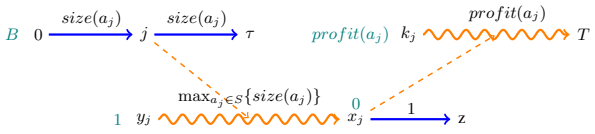



Fig. 5: The $j$ subnetwork of the constructed financial system

## 5 Conclusions and Future work

Financial networks have emerged as a fertile research area in both computational complexity and algorithmic game theory. It is paramount to understand systemic risk in finance from both these perspectives. In this paper, we join both streams of work by settling questions around the computational complexity of systemic risk for priority payments, a scheme so far only studied from the game-theoretic point of view. In an interesting parallel with the state of the art for proportional payments, we prove that computing clearing recovery rates is FIXP-complete whereas it is NP-hard to compute the priority lists optimising several measures of financial health of the system.

Our work paves the way for studying payment schemes in financial networks in more detail. Is there a payment scheme for financial networks with derivatives that makes the computation of systemic risk easy and/or that induces "nice" equilibria? We also wonder the extent to which the flexibility of working with financial networks can lead to a deeper understanding of FIXP; e.g., are there payment schemes that can be proved complete for variants of FIXP defined upon a different basis?

## References

1. Daron Acemoglu, Asuman Ozdaglar, and Alireza Tahbaz-Salehi. Systemic risk and stability in financial networks. *American Economic Review*, 105(2):564–608, 2015.
2. Alberto Bertoni, Giancarlo Mauri, and Nicoletta Sabadini. A characterization of the class of functions computable in polynomial time on random access machines. In *STOC*, pages 168–176. ACM, 1981.
3. Nils Bertschinger, Martin Hoefer, and Daniel Schmand. Strategic payments in financial networks. In Thomas Vidick, editor, *ITCS*, volume 151 of *LIPIcs*, pages 46:1–46:16, 2020.
4. Allan Borodin, Ronald Fagin, John E. Hopcroft, and Martin Tompa. Decreasing the nesting depth of expressions involving square roots. *J. Symb. Comput.*, 1(2):169–188, 1985.
5. Rodrigo Cifuentes, Gianluigi Ferrucci, and Hyun Song Shin. Liquidity risk and contagion. *Journal of the European Economic association*, 3(2-3):556–566, 2005.

6. Larry Eisenberg and Thomas H. Noe. Systemic risk in financial systems. *Manag. Sci.*, 47(2):236–249, 2001.
7. Matthew Elliott, Benjamin Golub, and Matthew O Jackson. Financial networks and contagion. *American Economic Review*, 104(10):3115–53, 2014.
8. Kousha Etessami and Mihalis Yannakakis. On the complexity of nash equilibria and other fixed points. *SIAM J. Comput.*, 39(6):2531–2597, 2010.
9. Aris Filos-Ratsikas, Yiannis Giannakopoulos, Alexandros Hollender, Philip Lazos, and Diogo Poças. On the complexity of equilibrium computation in first-price auctions. In *EC*, pages 454–476, 2021.
10. Aris Filos-Ratsikas, Kristoffer Arnsfelt Hansen, Kasper Høgh, and Alexandros Hollender. Fixp-membership via convex optimization: Games, cakes, and markets. In *FOCS*, pages 827–838, 2021.
11. Michael R Garey, Ronald L Graham, and David S Johnson. Some np-complete geometric problems. In *STOC*, pages 10–22, 1976.
12. Paul Glasserman and H Peyton Young. How likely is contagion in financial networks? *Journal of Banking & Finance*, 50:383–399, 2015.
13. Paul W. Goldberg and Alexandros Hollender. The hairy ball problem is ppad-complete. *J. Comput. Syst. Sci.*, 122:34–62, 2021.
14. Sebastian Heise and Reimer Kühn. Derivatives and credit contagion in interconnected networks. *The European Physical Journal B*, 85(4):1–19, 2012.
15. Brett Hemenway and Sanjeev Khanna. Sensitivity and computational complexity in financial networks. *Algorithmic Finance*, 5(3-4):95–110, 2016.
16. Daning Hu, J Leon Zhao, Zhimin Hua, and Michael CS Wong. Network-based modeling and analysis of systemic risk in banking systems. *MIS quarterly*, pages 1269–1291, 2012.
17. Stavros D. Ioannidis, Bart de Keijzer, and Carmine Ventre. Strong approximations and irrationality in financial networks with derivatives. In *ICALP*, 2022.
18. Panagiotis Kanellopoulos, Maria Kyropoulou, and Hao Zhou. Financial network games. *CoRR*, abs/2107.06623, 2021.
19. Panagiotis Kanellopoulos, Maria Kyropoulou, and Hao Zhou. Forgiving debt in financial network games. *CoRR*, abs/2202.10986, 2022.
20. Christos H. Papadimitriou. The euclidean traveling salesman problem is np-complete. *Theor. Comput. Sci.*, 4(3):237–244, 1977.
21. Pál András Papp and Roger Wattenhofer. Network-aware strategies in financial systems. In *ICALP*, volume 168, pages 91:1–91:17, 2020.
22. Pál András Papp and Roger Wattenhofer. Debt swapping for risk mitigation in financial networks. In *EC*, pages 765–784, 2021.
23. Pál András Papp and Roger Wattenhofer. Default ambiguity: Finding the best solution to the clearing problem. In *WINE*, volume 13112, pages 391–409, 2021.
24. Pál András Papp and Roger Wattenhofer. Sequential defaulting in financial networks. In *ITCS*, volume 185, pages 52:1–52:20, 2021.
25. L. C. G. Rogers and Luitgard A. M. Veraart. Failure and rescue in an interbank network. *Manag. Sci.*, 59(4):882–898, 2013.
26. Arnold Schönhage. On the power of random access machines. In *ICALP*, volume 71, pages 520–529, 1979.
27. Steffen Schuldenzucker and Sven Seuken. Portfolio compression in financial networks: Incentives and systemic risk. In *EC*, page 79, 2020.
28. Steffen Schuldenzucker, Sven Seuken, and Stefano Battiston. Clearing payments in financial networks with credit default swaps [extended abstract]. In *EC*, page 759, 2016.
29. Steffen Schuldenzucker, Sven Seuken, and Stefano Battiston. Finding clearing payments in financial networks with credit default swaps is PPAD-complete. In *ITCS*, volume 67, pages 32:1–32:20, 2017.

30. Prasoon Tiwari. A problem that is easier to solve on the unit-cost algebraic RAM. *J. Complex.*, 8(4):393–397, 1992.
31. Mihalis Yannakakis. Equilibria, fixed points, and complexity classes. In *STACS*, volume 1, pages 19–38, 2008.

## A Instance with multiple clearing recovery rate vectors

The financial network of Fig. 6 consists of three banks $1, 2, 3$, two debt contracts, $(1, 2)$ and $(2, 3)$, and one CDS contract, $(1, 3, 2)$. All contract notionals are set to one. Let node 1 pay according to the priority list $P_1 = ((1, 3, 2) \mid (1, 2))$. For node 1 it holds that $e_1 = 1$ and $l_1(r) = l_{1,2}(r) + l_{1,3}(r) = c_{1,2} + (1 - r_2)c^2_{1,3} = 2 - r_2$, thus $r_1 = 1/(2 - r_2)$. Moreover 1 can fully pay any liability generated from the contract $(1, 3, 2)$ because $e_1 = 1 > 1 - r_2 = l_{1,3}$. This means that node 2 receives an incoming payment of $1 - p_{(1,3,2)} = 1 - (1 - r_2) = r_2$. It is not hard to verify that the set of all clearing recovery rate vectors are of the form $r = (1/(2 - r_2), r_2, 1), r_2 \in [0, 1]$.

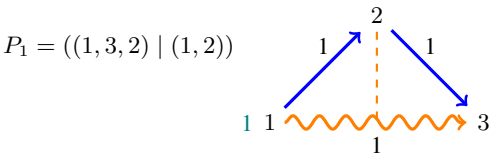



Fig. 6: A pair of a financial network and a singleton liability priority list that admits multiple clearing recovery rate vectors. Whenever 1 pays according to $P_1 = ((1, 3, 2) \mid (1, 2)))$ the clearing recovery rate vectors are of the form $r = (1/(2 - r_2), r_2, 1), r_2 \in [0, 1]$.

## B Proofs of Section 4

**Lemma 4.** *Finding a prioritly list profile that minimises the number of activated CDSes is NP-hard.*

*Proof.* We reduce from the problem of minimising the number of defaulting nodes, proved NP-hard in Lemma 2. Let $\mathcal{F}$ be a financial system and let $m$ be the maximum number of CDS contracts issued on the same reference bank, meaning that there exists a node that is the reference bank in $m$ CDSes and no other node is a reference bank of more CDSes than that. We construct a financial network $\mathcal{F}'$, in which we maintain all nodes and contracts, retain the same contract notionals, and add for each node $i$ of $\mathcal{F}$ a number $n_i$ of CDS contracts $(\alpha^1_i, \beta^1_i, i), \ldots, (\alpha^{n_i}_i, \beta^{n_i}_i, i)$ with $c^i_{\alpha_i,\beta_i} = 1$ and $e_{\alpha_i} = 1$ which we call dummy CDSes (where $\alpha^j_i, \beta^j_i, j \in [n_i]$, are newly introduced nodes) such that eventually all nodes in $\mathcal{F}'$ are reference banks to exactly $m$ CDSes. Fig. 7 shows an example of this reduction.

By this construction, it is straightforward to see that if under any priority list profile a set of banks default, then in $\mathcal{F}'$, for each of these banks $i$, exactly $m$ distinct CDSes activate in which $i$ is the reference bank. Since the priority lists of $\mathcal{F}$ and $\mathcal{F}'$ are in one-to-one correspondence with each other, we conclude that finding the priority list profile minimising the number of activated CDSes in $\mathcal{F}'$ is equivalent to finding the priority list profile minimising the number of defaulting banks in $\mathcal{F}$. □

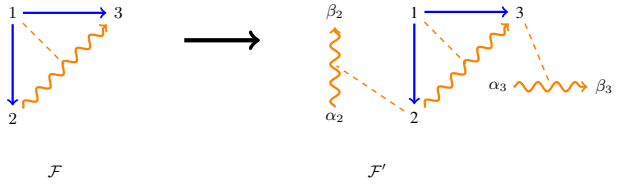



Fig. 7: $\mathcal{F}$ and $\mathcal{F}'$

In [19], the authors define the term systemic liquidity as the total amount of payments that are being transacted among the economic firms in the financial system under some clearing recovery rate vector. Given a financial system $\mathcal{F}$ we use the notation $\mathcal{L}^{\mathcal{P}}_{\mathcal{F}}(r)$ to denote the systemic liquidity of the system under the priority profile $\mathcal{P}$ and assuming a clearing recovery rate vector $r$ and is defined as $\mathcal{L}^{\mathcal{P}}_{\mathcal{F}}(r) = \sum_{i\in N}\sum_{j\in N} p_{i,j}(r)$. Like the previous objectives it turns out that choosing the profile that maximises the systemic liquidity in a financial network is NP-hard.

**Lemma 5.** *Finding a priority list profile that maximises the systemic liquidity is NP-hard.*

*Proof.* We prove the lemma by a reduction from the problem in Lemma 2. Given a financial system $\mathcal{F}$ we construct a modified financial system $\mathcal{F}'$ according to the following procedure:

- For each debt contract $(i,j) \in \mathcal{DC}_{\mathcal{F}}$ with contract notional $c_{i,j}$: Erase the debt contract $(i,j)$. Add a new node denoted as $\tau_{ij}$ with $e_{\tau_{ij}} = 0$ and two new debt contracts $(i,\tau_{ij})$ and $(\tau_{ij},j)$ each with contract notional $c_{i,\tau_{ij}} = c_{\tau_{ij},j} = c_{i,j}$. Finally add two new nodes $\alpha_{ij}$ and $\beta_{ij}$ with $e_{\alpha_{ij}} = 2c_{i,j}$ and construct the CDS contract $(\alpha_{ij},\beta_{ij},\tau_{ij})$ with contract notional $c^{\tau_{ij}}_{\alpha_{ij},\beta_{ij}} = c_{i,j}$. We refer to this construction as the $\tau_{ij}$-gadget. It is illustrated in Fig. 8.
- For each CDS contract $(i,j,k) \in \mathcal{CDS}_{\mathcal{F}}$ with contract notional $c^k_{i,j}$: Erase the CDS $(i,j,k)$. Add a new node denoted as $\tau^k_{ij}$ with $e_{\tau^k_{ij}} = 0$ and construct the CDS contract $(i,\tau^k_{ij},k)$ with contract notional $c^k_{i,\tau^k_{ij}} = c^k_{ij}$ and the debt contract $(\tau^k_{ij},j)$ with contract notional $c_{\tau_{ij},j} = c^k_{i,j}$. Finally we add two new nodes $\alpha^k_{ij}$ and $\beta^k_{ij}$ with $e_{\alpha^k_{ij}} = 2c^k_{i,j}$ and construct the CDS $(\alpha^k_{ij},\beta^k_{ij},\tau^k_{ij})$ with contract notional $c^{\tau^k_{ij}}_{\alpha^k_{ij},\beta^k_{ij}} = 2c^k_{ij}$. We refer to this construction as the $\tau^k_{ij}$-gadget. It is illustrated in Fig. 9.
- For each node $k \in N_{\mathcal{F}}$: We add five new nodes $\chi_k,\psi_k,\zeta_k,\alpha_k,\beta_k$ with $e_{\chi_k} = 1/(2 \mid N_{\mathcal{F}} \mid)$, $e_{\alpha_k} = 2$ and construct the CDS $(\chi_k,\psi_k,k)$ with $c^k_{\chi_k,\psi_k} = \infty$[6], the debt contract $(\psi_k,\zeta_k)$ with $c_{\psi_k,\zeta_k} = 1/(2 \mid N_{\mathcal{F}} \mid)$ and the CDS $(\alpha_k,\beta_k,\psi_k)$ with $c^{\psi_k}_{\alpha_k,\beta_k} = 2$. We refer to this construction as $k$-gadget. It is illustrated in Fig. 10.

[6] Here we assume that a contract may have $\infty$ notional in the sense that whenever the debtor defaults it must submit all of its remaining assets through this contract.

For any priority profile $\mathcal{P}$ of $\mathcal{F}$ we define its 'natural extension' priority profile denoted as $\mathcal{P}^{\text{ext}}$ of $\mathcal{F}'$ as follows: For a node $i \in N_{\mathcal{F}}$ having a singleton liability priority list $\mathcal{P}_i$ we substitute each priority that corresponds to some debt contract $(i,j) \in \mathcal{DC}_{\mathcal{F}}$ with the debt contract $(i,\tau_{ij}) \in \mathcal{DC}_{\mathcal{F}'}$ and each priority that corresponds to some CDS contract $(i,j,k) \in \mathcal{CDS}_{\mathcal{F}}$ with the CDS contract $(i,\tau_{ij}^k,k) \in \mathcal{CDS}_{\mathcal{F}'}$. For all other nodes appearing in the constructed gadgets, the priority lists are uniquely defined since they have only one outgoing edge. For example if some node $i$ pays in $\mathcal{F}$ according to $\mathcal{P}_i = ((i,j) \mid (i,j,k))$ then it pays in $\mathcal{F}'$ according to $\mathcal{P}_i^{\text{ext}} = ((i,\tau_{ij}) \mid (i,\tau_{ij}^k,k))$. Next we prove three useful claims that we use in our reduction.

**Claim 1** *The 'natural extension' priority profile does not affect the recovery rate of nodes belonging in $N_{\mathcal{F}} \cap N_{\mathcal{F}'}$. Namely if node $i$ has recovery rate $r_i$ under $\mathcal{P}$ in $\mathcal{F}$ then it has recovery rate $r_i$ under $\mathcal{P}^{ext}$ in $\mathcal{F}'$.*

*Proof.* The amount of money transferred via a liability $(i,j)$ in $\mathcal{F}$ under $\mathcal{P}$ are transferred from $\tau_{ij}$ to node $j$ under $\mathcal{P}^{\text{ext}}$ in $\mathcal{F}'$, since node $\tau_{ij}$ has an incoming payment of at most $c_{i,j}$, a liability of $c_{i,j}$ and zero external assets. Similarly the amount of money that are being transferred via a CDS contract $(i,j,k)$ in $\mathcal{F}$ under $\mathcal{P}$ are transferred from $\tau_{ij}^k$ to node $j$ in $\mathcal{F}'$ under $\mathcal{P}^{\text{ext}}$ because $\tau_{ij}^k$ receives a payment of at most $(1-r_k)c_{i,j}^k$, has a liability of $c_{i,j}^k$ and zero external assets. Also by the way we constructed $\mathcal{F}'$ the liabilities of all nodes appearing in $\mathcal{F}$ are unchanged. That means that the 'natural extension' profile $\mathcal{P}^{\text{ext}}$ of a profile $\mathcal{P}$ does not affect the assets and liabilities of nodes in $N_{\mathcal{F}} \cap N_{\mathcal{F}'}$, thus their recovery rate is the same. □

Next we denote by $\mathcal{F}'_\tau$ the financial system that is constructed from $\mathcal{F}$ only by adding the $\tau_{ij}$-gadget and $\tau_{ij}^k$-gadget. We denote by $\mathcal{P}_\tau$ the 'natural extension' of any priority profile $\mathcal{P}$ of $\mathcal{F}$ in $\mathcal{F}_\tau$. In the following claim we prove that the liquidity of $\mathcal{F}'_\tau$ is the same under any priority profile and any clearing recovery rate vector.

**Claim 2** *For each priority profile $\mathcal{P}$ and each clearing recovery rate vector, $\mathcal{L}^{\mathcal{P}}_{\mathcal{F}'_\tau}(r) = 2\left(\sum_{i,j\in N_{\mathcal{F}}} c_{i,j} + \sum_{i,j,k\in N_{\mathcal{F}}} c_{i,j}^k\right)$.*

*Proof.* Assume a priority profile $\mathcal{P}$ of $\mathcal{F}$ and let $\mathcal{P}_\tau$ be its 'natural extension' profile in $\mathcal{F}_\tau$. First we will prove that each $(i,j) \in \mathcal{DC}_{\mathcal{F}}$ generates liquidity of $2c_{i,j}$ in $(\mathcal{F}_\tau, \mathcal{P}_\tau)$. Assume $l \leq c_{i,j}$ to be the amount of money transferred via some debt contract $(i,j)$ in $(\mathcal{F},\mathcal{P})$. We discriminate two cases.

1. $l < c_{i,j}$: The recovery rate for node $\tau_{ij}$ is $r_{\tau_{ij}} = l/c_{i,j} < 1$ thus the CDS $(\alpha_{ij},\beta_{ij},\tau_{ij})$ is activated and node $\alpha_{ij}$ owes $2(1-r_{\tau_{ij}})c_{i,j} = 2c_{i,j} - 2l$ to node $\beta_{ij}$ which can fully pay off. Edges $(i,\tau_{ij})$ and $(\tau_{ij},j)$ generate a liquidity of $2l$ thus the generated liquidity in that case is $2c_{i,j} - 2l + 2l = 2c_{i,j}$.
2. $l = c_{i,j}$: The recovery rate of node $\tau_{ij}$ is $r_{\tau_{ij}} = 1$ thus the $(\alpha_{ij},\beta_{ij},\tau_{ij})$ is not activated. The generated liquidity in that case arises only from edges $(i,\tau_{ij})$ and $(\tau_{ij},j)$ and is equal to $2c_{i,j}$.

Next we will prove that any CDS contract $(i,j,k) \in \mathcal{CDS}_{\mathcal{F}}$ generates a liquidity of $2c_{i,j}^k$ in $(\mathcal{F}_\tau,\mathcal{P}_\tau)$. Assume a clearing recovery rate vector $r$ and let $l \leq (1-r_k)c_{i,j}^k$ to be the amount of money transferred via some CDS $(i,j,k)$ in $(\mathcal{F},\mathcal{P})$. We discriminate three cases.

1. $l = (1 - r_k)c^k_{i,j}$: The recovery rate for node $\tau_{ij}$ is $r_{\tau_{ij}} = l/c^k_{i,j} = 1 - r_k < 1$ thus the CDS $(\alpha^k_{ij}, \beta^k_{ij}, \tau^k_{ij})$ is activated and node $\alpha^k_{ij}$ owes $2(1 - r_{\tau^k_{ij}})c^k_{i,j} = 2r_k c^k_{i,j}$, which can fully pay off. Since edges $(i, \tau_{i,j})$ and $(\tau_{ij}, j)$ generate a liquidity of $2(1 - r_k)c^k_{i,j}$ the total generated liquidity is $2r_k c^k_{i,j} + 2c^k_{i,j} - 2r_k c^k_{i,j} = 2c^k_{i,j}$.
2. $l < (1 - r_k)c^k_{i,j}$: Again $r_{\tau_{i,j}} = l/c^k_{i,j} < 1$ thus $(\alpha^k_{ij}, \beta^k_{ij}.\tau^k_{ij})$ is activated and $\alpha^k_{ij}$ owes $2c^k_{i,j}(1-(l/c^k_{i,j})) = 2c^k_{i,j} - 2l$ to $\beta^k_{ij}$, which can fully pay. The total generated liquidity is $2l + 2c^k_{i,j} - 2l = 2c^k_{i,j}$.
3. $r_k = 1$: From Claim 1, we know that the recovery rate of any node does not change under $\mathcal{P}^{\text{ext}}$. It is not hard to see that the same holds under $\mathcal{P}_\tau$. If $r_k = 1$ then $r_{\tau^k_{ij}} = 0$ and the $(i, \tau^k_{ij}, k)$ is inactive. So node $\alpha^k_{ij}$ owes $2c^k_{i,j}$ to $\beta^k_{ij}$, which can fully pay and the generated liquidity is $2c^k_{i,j}$. □

Notice that $\mathcal{F}'$ is actually $\mathcal{F}_\tau$ with the addition of the $k$-gadgets, and since we proved that the systemic liquidity of $\mathcal{F}'$ is always the same under any priority profile it must be the case that the liquidity if $\mathcal{F}'$ depends on the generated liquidity in the $k$-gadgets.

**Claim 3** *The generated liquidity from a $k$-gadget under any clraring recovery rate vector is $1/(\mid N_{\mathcal{F}} \mid)$ whenever $r_k < 1$ and 2 whenever $r_k = 1$.*

*Proof.* Assume some node $k$ and let a clearing recovery rate vector with $r_k = 1$ in $\mathcal{F}$ under some profile $\mathcal{P}$. Again from claim 1 we get that $r_k = 1$ in $\mathcal{F}'$ under $\mathcal{P}^{\text{ext}}$, thus the CDS $(\chi_k, \psi_k, k)$ is inactive which means that $r_{\psi_k} = 0$ and node $\alpha_k$ owes 2 to node $\beta_k$ which it can fully pay off and the generated liquidity in that case is 2. If $r_k < 1$ then the CDS $(\chi_k, \psi_k, k)$ is activated and since $c^k_{\chi_k,\psi_k} = \infty$ node $\chi_k$ must submit all of its assets to $\psi_k$. Now $\psi_k$ can fully pay off $\zeta_k$ thus the $(\alpha_k, \beta_k, \psi_k)$ is inactive and the generated liquidity is $1/ \mid N_{\mathcal{F}} \mid$. □

Next we provide the reduction based on the above three claims. Assume $\mathcal{P}$ is the priority profile of $\mathcal{F}$ under which the number of defaulting nodes is minimised and let $\lambda$ be that number. We construct the financial system $\mathcal{F}'$ as described above and consider the priority profile $\mathcal{P}^{\text{ext}}$. We will prove that under $\mathcal{P}^{\text{ext}}$ the liquidity in $\mathcal{F}'$ is maximised. From Claim 1 we know that the recovery rate of each node in $\mathcal{F}$ under $\mathcal{P}^{\text{ext}}$ is unchanged thus if a node is in default in $\mathcal{F}$ under $\mathcal{P}$ for some clearing recovery rate vector then it is in default in $\mathcal{F}'$ under $\mathcal{P}^{\text{ext}}$. That means that there exist exactly $\lambda$ $k$-gadgets, which by Claim 3 generate liquidity $\lambda/ \mid N_{\mathcal{F}} \mid$, which (taking into consideration Claim 2) means that

$$\mathcal{L}^{\mathcal{P}^{\text{ext}}}_{\mathcal{F}'}(r) = 2\left(\sum_{i,j \in N_{\mathcal{F}}} c_{i,j} + \sum_{i,j,k \in N_{\mathcal{F}}} c^k_{i,j}\right) + \frac{\lambda}{\mid N_{\mathcal{F}} \mid} + 2(\mid N_{\mathcal{F}} \mid -\lambda).$$

Assume there exists another profile $\mathcal{Q}^{\text{ext}}$ such that $\mathcal{L}_{\mathcal{F}'}^{\mathcal{Q}^{\text{ext}}}(r) > \mathcal{L}_{\mathcal{F}'}^{\mathcal{P}^{\text{ext}}}(r)$. This means that there must exist another $\lambda'$ such that

$$2\left(\sum_{i,j\in N_{\mathcal{F}}} c_{i,j} + \sum_{i,j,k\in N_{\mathcal{F}}} c_{i,j}^{k}\right) + \frac{\lambda'}{\mid N_{\mathcal{F}} \mid} + 2(\mid N_{\mathcal{F}} \mid -\lambda') >$$
$$2\left(\sum_{i,j\in N_{\mathcal{F}}} c_{i,j} + \sum_{i,j,k\in N_{\mathcal{F}}} c_{i,j}^{k}\right) + \frac{\lambda}{\mid N_{\mathcal{F}} \mid} + 2(\mid N_{\mathcal{F}} \mid -\lambda),$$

which is true if and only if $\lambda' < \lambda$. This means that the natural extension of the priority profile that minimises the number of defaulting nodes in $\mathcal{F}$ maximises the systemic liquidity in $\mathcal{F}'$ and vice versa. □

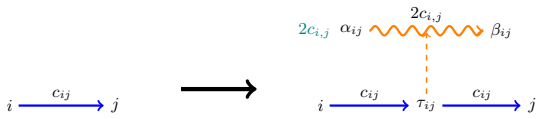



Fig. 8: Transformation of an $(i, j)$ contract that appears in $\mathcal{F}$ to a $\tau_{ij}$-gadget appearing in ${\mathcal{F}'}_{\tau}$ and $\mathcal{F}'$

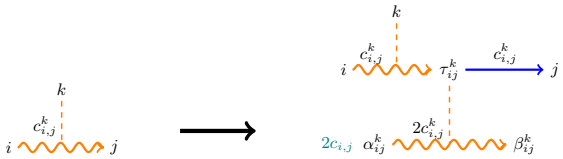



Fig. 9: Transformation of an $(i, j, k)$ contract that appears in $\mathcal{F}$ to a $\tau_{ij}^{k}$-gadget appearing in ${\mathcal{F}'}_{\tau}$ and $\mathcal{F}'$

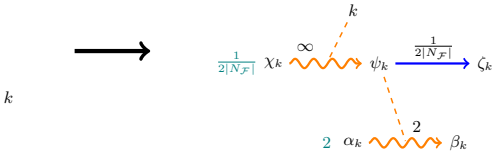



Fig. 10: Addition of the $k$-gadget in $\mathcal{F}'$ for every node $k$ of $\mathcal{F}$

## C Financial System Gadgets

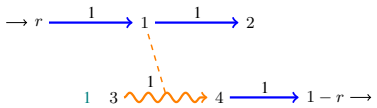



Fig. 11: Inversion gadget $g_{\text{inv}}$: Computing $1 - r$ from $r$.

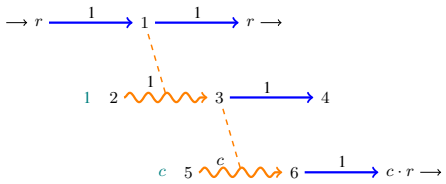



Fig. 12: Duplication (and multiplication-by-constant) gadget $g_{\text{dup}}$: This gadget outputs two values: $r$ and $cr$, where $c \in [0, 1]$ is a rational constant. Choosing $c = 1$ yields a duplication gadget that takes the input recovery rate $r$ and outputs $r$ as the two recovery rates of the output nodes. We may also denote this gadget by $g_{c \cdot r}$, for $c \in [0, 1]$ in case this gadget is used to multiply the input by a constant $c$.

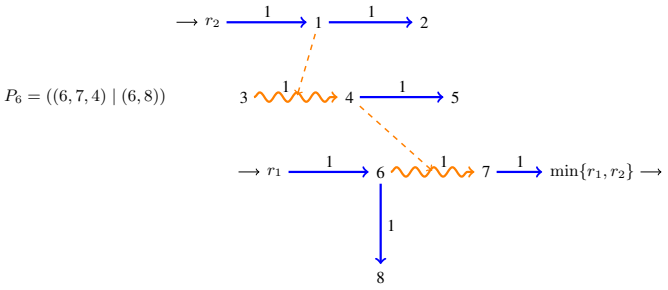



Fig. 13: Minimum gadget $g_{\min}$, computing $\min\{r_1, r_2\}$. In this gadget we assume node 6 pays according to the priority list $P_6 = ((6, 7, 4) \mid (6, 8))$. Node 6 has a liability of $r_2$ and an incoming payment of $r_1$ thus since it prioritises the contract $(6, 7, 4)$ it submits to node 7 the $\min\{r_1, r_2\}$.

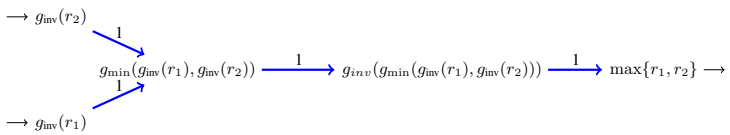



Fig. 14: Maximum gadget $g_{\max}$, computing $\max\{r_1, r_2\}$. This is a compact representation where the nodes labeled with a subscripted $g$ have to be replaced by copies of the respective gadgets, in order to obtain the full financial system defining the gadget. This gadget exploits the fact that $\max\{a, b\} = 1 - \min\{1 - a, 1 - b\}$. for $a, b \in [0, 1]$

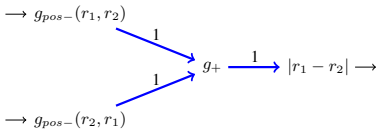



Fig. 15: Absolute difference gadget $g_{\text{abs}}$: This is a compact representation where the nodes labeled with a subscripted $g$ have to be replaced by copies of the respective gadgets, in order to obtain the full financial system defining the gadget. The recovery rate of the output node is $|r_1 - r_2|$, where $r_1$ and $r_2$ are the recovery rates of the input nodes. The gadget is formed by first applying two $g_{\text{pos}-}$ gates, the first on input $(r_1, r_2)$ and the second on input $(r_2, r_1)$. The two positive subtraction gadgets $g_{\text{pos}-}$ compute $\max\{r_1 - r_2, 0\}$ and $\max\{r_2 - r_1, 0\}$, after which these two maxima are added together using $g_+$, resulting in the desired output $|r_1 - r_2|$.

### C.1 Construction of the Division Gadget

To obtain our division gadget, we will essentially get rid of division gates altogether in $C'_I$, and replace each of them by a set of alternative operations that achieve the same result. We thus make a few modifications to $C'_I$. The first modification is that we alter the sub-circuit $T$, which is used in the pre-processing step to generate the value $t = 1/2^{2^d}$, which scales all the signals in the circuit so that each signal is in $[0, 1]$ irrespective of the input vector. Here, $d$ is a polynomial time computable value that satisfies that every signal in the original circuit $C_I$ is at most $2^{2^d}$. We adapt $T$ such that the scaling factor it outputs is $1/(2^{1+2^d})$ instead of $1/2^{2^d}$. This can be done by adding one additional multiplication gate at the end of $T$. Let the output of the modified $T$ be $t' = t/2$. After this modification, it holds that all signals inside the circuit are in $[0, 1/2]$.

We now make use of the fact that division gates are used in a very limited way in $C'_I$ (with $T$ modified as above).

- First, division is used for dividing by $c$, where $c$ is a given explicit constant in the original circuit $C_I$. For such divisions, we can simply use our constant multiplication gadget $g_{\text{dup}}$ to multiply by $1/c$, which is equivalent to dividing by $c$.
- Secondly, division is used to divide certain outputs of gates by $t'$ (previously $t$), where $t' = 1/2^{1+2^d}$. This type of division happens in two cases.
  1. At every point in the circuit where two scaled signals $t'a$ and $t'b$ with $a, b \leq 2^{2^d}$ are multiplied with each other, resulting in the value $t't'ab$. This value is

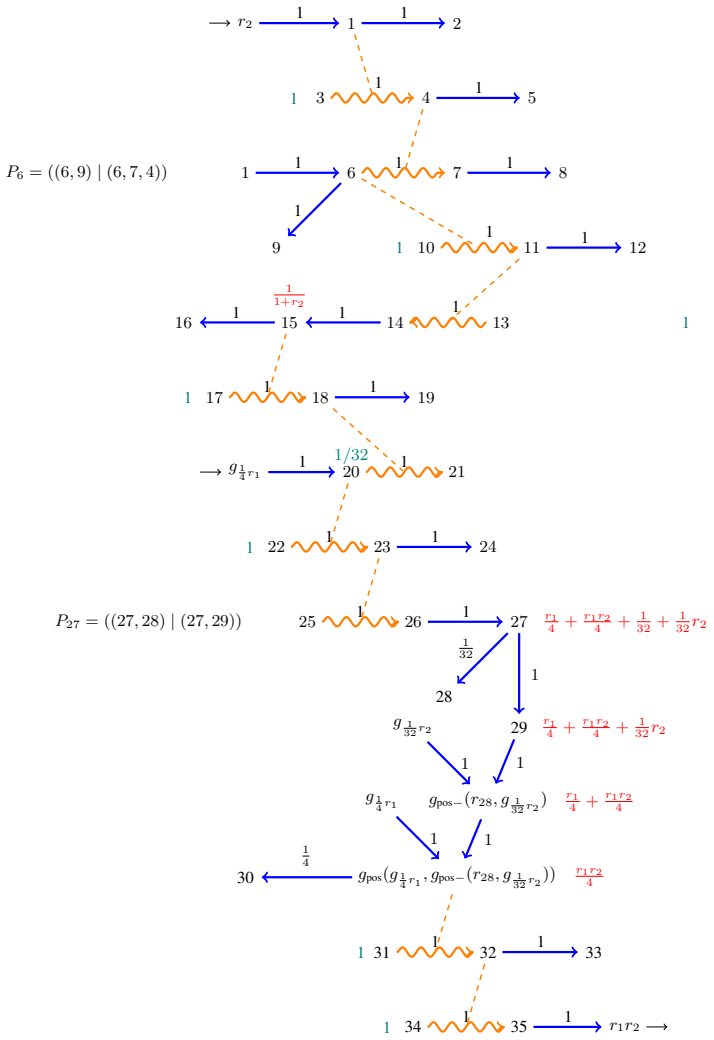



Fig. 16: Multiplication gadget $g_*$: This gadget's construction is rather involved and makes use of various instances of multiplication-by-constant gadgets and positive subtraction gadgets. The function of nodes 1 to 15 is to compute $1/(1 + r_2)$, which is subsequently transformed into the expression $r_1/4 + r_1 r_2/4 + 1/32 + 1/32 r_2$ in node 27. After node 27, it is then straightforward to extract the term $r_1 r_2$ from the latter expression by making use of multiplication-by-constant gadgets and positive subtraction gadgets. Some of the constants appearing in this gadget (particularly the factor $1/4$ and the external assets of $1/32$ in node 20) have the function to keep the expressions (marked in red) small enough so that they remain in the interval $[0, 1]$ and can thus take the form of a valid recovery rate. The presence of positive external assets at node 20 is also needed to ensure non-degeneracy, and this is also the reason for the presence of some of the debt contracts in this gadget.

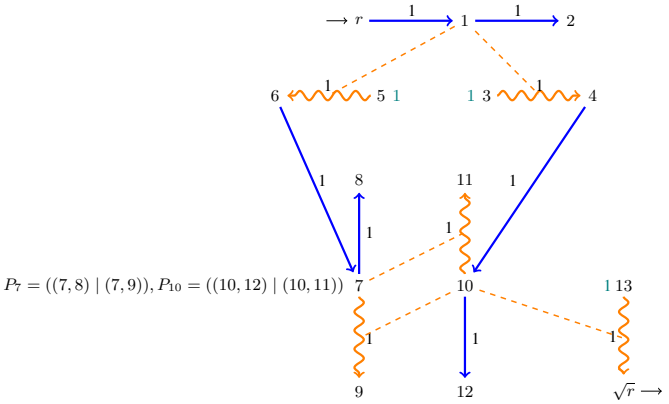



Fig. 17: Square Root gadget $g_{\sqrt{\cdot}}$: On input $r$ the recovery rate of the output node is $\sqrt{r}$. The gadget works as follows, nodes 4 and 6 receive a payment of $1 - r$ from nodes 3 and 5 respectively and pass them on to nodes 7 and 10 respectively. From figure 1, we know that $r_7 = r_{10} = 1 - \sqrt{r}$, thus generating a liability of $\sqrt{r}$ to node 13 which it can fully pay off, so that the terminal node receives a payment of $\sqrt{r}$.

divided by $t'$ in order to generate the signal $t'ab$, i.e., a scaled version of the signal $ab$ in the original circuit.

2. At the end of the circuit, where a scaled signal $t'a$ is divided by $t'$ to produce an output signal of the original circuit, which is in $[0, 1]$.

In the first case, let $x = t'ab$ and in the second case, let $x = a$, so that in both cases a number $t'x$ is divided by $t'$ to result in $x$, and in both cases it holds that $x \in [0, 1]$. We replace the division $t'x/t'$ by a sequence of $d$ gates that compute the square root of its input, followed by a multiplication by 2, followed by a sequence of $d$ successive squaring multiplication gates, followed by a final multiplication by 2. This results in the correct value

$$((t'x)^{1/2^d} \cdot 2)^{2^d} \cdot 2 = (t'^{1/2^d} x^{1/2^d} \cdot 2)^{2^d} \cdot 2 = t'x \cdot 2^{2^d} \cdot 2 = \frac{1}{2} x \cdot 2 = x.$$

It is furthermore straightforward to verify that, due to the order in which we apply our arithmetic operations, all of the values throughout this computation stay in the interval $[0, 1]$. We note that the adapted scaling factor $t' = t/2$ is needed because of the multiplication by 2 that is executed after the successive square roots and before the successive squaring.

The resulting circuit has no division gates anymore, so we do not need a division gadget in our reduction. Instead, now we need a square root gadget, which is presented in Fig. 17.