Payment Scheduling in the Interval Debt Model

Tom Friedetzky[1][0000−0002−1299−5514], David C. Kutner[1][0000−0003−2979−4513], George B. Mertzios*[1][0000−0001−7182−585X], Iain A. Stewart[1][0000−0002−0752−1971], and Amitabh Trehan[1][0000−0002−2998−0933]

Department of Computer Science, Durham University, UK,
tom.friedetzky@durham.ac.uk, david.kutner@durham.ac.uk,
george.mertzios@durham.ac.uk, i.a.stewart@durham.ac.uk,
amitabh.trehan@durham.ac.uk

Abstract. The networks-based study of financial systems has received considerable attention in recent years, but seldom explicitly incorporated the dynamic aspects of such systems. We consider this problem setting from the temporal point of view, and we introduce the *Interval Debt Model (IDM)* and some scheduling problems based on it, namely: Bankruptcy Minimization / Maximization, in which the aim is to produce a schedule with at most / at least k bankruptcies; Perfect Scheduling, the special case of the minimization variant where $k=0$; and Bailout Minimization, in which a financial authority must allocate a smallest possible bailout package to enable a perfect schedule. In this paper we investigate the complexity landscape of the various variants of these problems. We show that each of them is NP-complete, in many cases even on very restrictive input instances. On the positive side, we provide for Perfect Scheduling a polynomial-time algorithm on (rooted) out-trees. In wide contrast, we prove that this problem is NP-complete on directed acyclic graphs (DAGs), as well as on instances with a constant number of nodes (and hence also constant treewidth). When the problem definition is relaxed to allow *fractional* payments, we show by a linear programming argument that Bailout Minimization can be solved in polynomial time.

Keywords: temporal graph · financial network · payment scheduling · NP-complete · polynomial-time algorithm

1 Introduction

In the study of financial systems, network-based paradigms were introduced to model behaviors associated with the connectedness and complexity exhibited in real-world financial systems. We introduce the *Interval Debt Model*, focusing on the *choices* that real-world financial entities have in times at which they pay their debts by applying temporal graphs to this setting. Previous work in the study of financial networks had seldom explicitly incorporated the temporal aspects inherent to real-world debt.

Financial Networks have been studied by applying concepts from ecology [9], statistical physics [3], and Boolean networks [6]. In 2001, Eisenberg and Noe (EN) [7] introduced a paradigm which has been the basis for much work in the network-based

* Partially supported by the EPSRC grant EP/P020372/1.

analysis of financial systems, see also e.g. the survey [11]. In this model, financial entities all operate within a single clearing system. The paradigm has been extended to include default costs [18], Credit Default Swaps (CDSs) (derivatives through which banks can bet on the default of another bank in the system) [19], and the sequential behavior of bank defaulting in real-world financial networks [16].

A core motivation of financial network analysis is to inform central banks' and regulators' policies. The concepts of *solvency* and *liquidity* are core to this task: a bank is said to be *solvent* if it has enough assets (including non-liquid assets) to meet all its obligations, and is said to be *liquid* if it has enough liquid assets (e.g. cash) to meet its obligations on time. An illiquid but solvent bank may exist even in modern interbank markets [17]. In such cases, a central bank may act as a *lender of last resort* and extend loans to such banks to prevent their defaulting on debts [2] [17]. The optimal allocation of bailouts to a system in order to minimize damage has also been studied as an extension of Eisenberg and Noe's model [15]. Here, bailouts refer to funds provided by a third party (such as the government) to entities to help them avoid bankruptcy.

Temporal Graphs are graphs whose underlying network structure changes over time. These allow us to model real-world networks which have inherent dynamic properties, such as transportation networks [20], contact networks in an epidemic [8], and communication networks; for an overview see [4, 5, 10]. Most commonly, following the formulation introduced by Kempe, Kleinberg and Kumar [13], a temporal graph has a fixed set of vertices, and edges which appear and disappear at integer times up to an (integer) lifetime T. In such cases the temporal graph can be thought of as a static graph $G=(V, E)$ in which the edges are labeled with the times at which they appear. Often, a natural extension of a problem on static graphs to the temporal setting yields a computationally harder problem; for example, finding node-disjoint paths in a temporal graph remains NP-complete even when the underlying graph is itself a path [14], and finding a temporal vertex cover remains NP-complete even on star temporal graphs [1].

Our contribution In this paper we present a new framework for considering problems of bailout allocation and payment scheduling in financial networks by taking into account the temporal aspect of debts between financial entities, the *Interval Debt Model* (IDM). We introduce several natural problems and problem variants in this model, and show that the tractability of such problems depends greatly on the network topology and on the restrictions on payments (i.e. the admission or exclusion of partial and fractional payments on debts). While previous work has mainly focused on static financial networks, we go further and introduce the time dimension in financial networks to account for the temporal nature of real-world debts. In particular, the IDM offers the capability to represent the flexibility that entities have in paying debts earlier or later, within some *interval*. In Section 2 we present our new model IDM in detail. The formal definitions of our problems, as well as a summary of our results (see Table 1) are given in Section 2.3. All our results are formally presented in Section 3. Due to space constraints, some proofs are deferred to more complete versions of this paper.

2 The Interval Debt Model

In this section, we introduce (first by example, then formally) the Interval Debt Model, a framework in which temporal graphs are used to represent the system of debts in a financial network.

As an example, consider a tiny financial network consisting of 3 banks u, v, w, with €30, €20 and €10 respectively in initial assets, and the following inter-bank financial obligations. Bank u owes bank v €20, which it must pay by time 3, and €15, which it must pay between times 4 and 5. Bank v has agreed to lend bank w €25 at time 2 exactly, which bank w must repay to bank v between times 4 and 6. A graphical representation of this system is shown in Fig. 1.

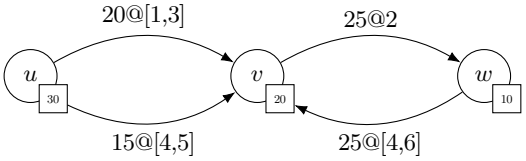


Fig. 1: A simple instance of the IDM

Several points can be made about this system: node u is insolvent as its €30 in assets are insufficient to pay all its debts; node v may be illiquid (it may default on part of its debt to w, e.g. if u pays all of its first debt at time 3) or may remain liquid (e.g. if it receives at least €5 from u by time 2); and node w is solvent and certain to remain liquid in any case.

One may ask several questions about this system: Are partial payments admitted (i.e. u paying €18 of the €20 debt at time 1, and the rest later)? If so, are non-integer payments admitted? Can money received be immediately forwarded (e.g. u paying v €20 at time 2, and v paying w €25 at time 2)?

We now specify in detail the setting we consider in the remainder of the paper.

2.1 Formal Setting

Formally, an Interval Debt Model (IDM) instance is a 3-tuple (G, D, A^0), where:

- $G=(V,E)$ is a finite digraph with n nodes (or, alternatively, banks) from $V=\{v_i: i=1, 2, ..., n\}$ and directed labelled multi-edges (but no loops) from $E\subseteq V\times V\times\mathbb{N}$, with the edge $(u,v,id)\in E$ denoting that there is a *debt*, whose label is *id*, from the *debtor* u to the *creditor* v; moreover, the labels of some pair (u,v) (appearing in at least one triple $(u,v,id)\in E$) form a non-empty contiguous integer sequence 0,1,2,... We refer to the subset of edges directed out of or in to some specific node v by $E_{\text{out}}(v)$ and $E_{\text{in}}(v)$, respectively.
- $D: E\to\{(a,t_1,t_2): a,t_1,t_2\in\mathbb{N}\setminus\{0\}, t_1\leq t_2\}$ is the *debt function* which associates *terms* to every debt (ordinarily, we abbreviate $D((u,v,n))$ as $D(u,v,n)$). Here, if e is a debt with terms $D(e)=(a,t_1,t_2)$ then a is the monetary amount to be paid and t_1 (resp. t_2) is the first (resp. last) time at which any portion of this amount can be paid; also, for any debt $e\in E$, we write $D(e)=(D_a(e),D_{t_1}(e),D_{t_2}(e))$. For simplicity of notation, we sometimes denote the terms $D(e)=(a,t_1,t_2)$ by $a@[t_1,t_2]$, and $a@t_1$ when $t_1=t_2$.
- $A^0=(e^0_{v_1},e^0_{v_2},...e^0_{v_n})\in\mathbb{N}^n$ is a tuple with $e^0_{v_i}$ denoting the *initial external assets* of bank v_i.

We refer to the greatest timestamp that appears in any debt for a given instance as *lifetime*. The instance shown in Fig. 1, which has lifetime $T=6$, is given by: $V=\{u,v,w\}$, $E=$

$\{(u,v,0),(u,v,1),(v,w,0),(w,v,0)\}$, $D(u,v,0)=(20,1,3)$, $D(u,v,1)=(15,4,5)$, $D(v,w,0)=(25,2,2)$, $D(w,v,0)=(25,4,6)$, and $A^0=(e_u^0,e_v^0,e_w^0)$, where $e_u^0=30$, $e_v^0=20$, and $e_w^0=10$.

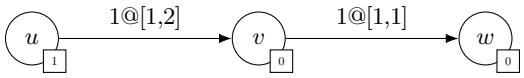


Fig. 2: An instance of the IDM with exactly two schedules.

Similarly, the instance shown in Fig. 2 has lifetime $T=2$ and is given by $V=\{u,v,w\}$, $E=\{(u,v,0),(v,w,0)\}$, $D(u,v,0)=(1,1,2)$, $D(v,w,0)=(1,1,1)$, and $A^0=(e_u^0,e_v^0,e_w^0)$, where $e_u^0=1$, $e_v^0=0$, $e_w^0=0$.

2.2 Schedules

Given an IDM instance (G,D,A^0), a *schedule* describes at what times the banks transfer *assets* to one another via *payments*. Formally, a schedule is a set of $|E|*T$ *payment* values $p_e^t \geq 0$, one for each time-edge pair. Equivalently, a schedule can be expressed as an $|E|\times T$ matrix S, and the variables p_e^t are the entries of that matrix. The value of p_e^t is the monetary amount of the debt e paid at time t. Our intention is that at any time $t\in[1,T]$, every payment $p_e^t>0$ of a schedule S is paid by the debtor of e to the creditor of e, not necessarily for the full amount $D_a(e)$ but for the amount p_e^t. A schedule for the instance of Fig. 2 consists of the four payments $p^1_{(u,v,0)}$, $p^1_{(v,w,0)}$, $p^2_{(u,v,0)}$ and $p^2_{(v,w,0)}$. Note that, using the above representations of a schedule S, we might have a large number of zero payments. Therefore, for simplicity of presentation, in the remainder of the paper we specify schedules by only specifying the non-zero payments. An example schedule for the instance in Fig. 2 is then $p^1_{(u,v,0)}=1$, $p^1_{(v,w,0)}=1$.

We now introduce some auxiliary variables which are not strictly necessary but help us to concisely express constraints on and properties of schedules (for nodes $u,v\in V(G)$ and time $t\in[T]$):

- Denote by I_v^t the total monetary amount of incoming payments of node v at time t.
- Denote by O_v^t the total monetary amount of outgoing payments (expenses) of node v at time t.
- We write $p_{u,v}^t$ to denote the total amount of all payments made from u to v at time t in reference to *all* debts from u to v. That is, $p_{u,v}^t=\sum_i p^t_{(u,v,i)}$.
- We denote by e_v^t node v's external assets at time t. Then $e_v^t=e_v^{t-1}+I_v^t-O_v^t$ for every v and t.

Note that, whenever there is only one edge from a node u to a node v, we have $p_{u,v}^t=p^t_{(u,v,0)}$; we use this in proofs for conciseness where possible. Recall the example schedule for Fig. 2, which we can then represent as $p^1_{u,v}=1$, $p^1_{v,w}=1$. As we shall see, the payments in this schedule can be legitimately discharged in order to satisfy the terms of all debts but in general this need not be the case. In fact, there might be schedules that are invalid, as well as schedules in which banks default on debts (go bankrupt). We deal with the notions of validity and bankruptcy now.

Definition 1 (Valid schedule; payable, due, overdue debts). *A schedule is* valid *if it satisfies the following properties (for any edge e and debt $D(e)=(a,t_1,t_2)$):*
- *All payment variables are nonnegative. That is, $p_e^t \geq 0$ for every e and t.*
- *All asset variables (as derived from payment variables and initial assets) are nonnegative. That is, $e_v^t \geq 0$ for every v and t.*
- *No debts are overpaid. That is, $\sum_{t\in[t_1,T]} p_e^t \leq a$.*
- *No debts are paid early. $\sum_{t\in[0,t_1-1]} p_e^t = 0$.*

Given some IDM instance and schedule, a debt $D(e)=a@[t_1,t_2]$ is said to be payable *for the interval $[t_1,t_2-1]$. At time t_2, $D(e)$ is said to be* due. *At every time $t \geq t_2$, if the full amount a has not yet been paid with reference to e, then $D(e)$ is said to be* overdue *at time t. A debt is* active *whenever it is payable, due, or overdue.*

A bank is said to be withholding *if, at some time t, it has an overdue debt and sufficient assets to pay (part of, where fractional or partial payments (see below) are permitted) the debt. If any bank is withholding in the schedule, then the schedule is not valid.*

Definition 2 (Bankrupt). *A bank is said to be* bankrupt *(at time t) if it is the debtor of an overdue debt (at time t). We say a schedule has k* bankruptcies *if k distinct banks go bankrupt at any point in the schedule. A bank may* recover *from bankruptcy if it receives sufficient income to pay off all its overdue debts.*

Definition 3 (Insolvent). *A bank v is said to be* insolvent *if all its assets (the sum of all debts due to v and of v's initial assets) are insufficient to cover all its obligations (the sum of all debts v owes). Formally, v is insolvent if*

$$e_v^0 + \sum_{e\in E_{in}(v)} D_a(e) \;<\; \sum_{e\in E_{out}(v)} D_a(e)$$

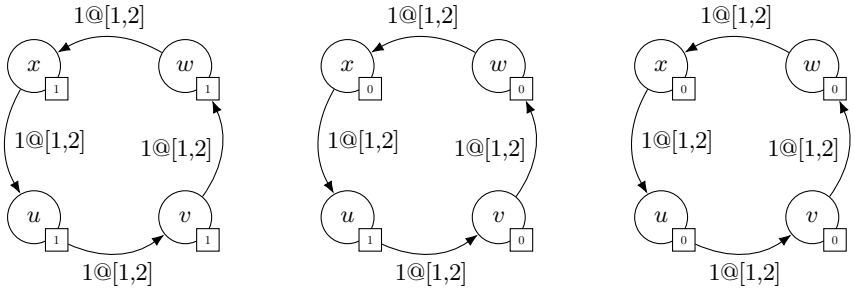
A bank which is insolvent will necessarily be bankrupt in any schedule.

We emphasize that the timing of bankruptcy and recovery or not of the banks is not considered.

We consider three natural variants of the model, in which different natural constraints are imposed on the payment variables:
- In the **Fractional Payments (FP)** variant, the payment variables may take rational values. That is, $p_e^t \in \mathbb{Q}$ for every e and t.
- In the **Partial Payments (PP)** variant, the payment variables may take only integer values. That is, $p_e^t \in \mathbb{N}$ for every e and t, and we allow payments for a smaller amount than the total debt.
- In the **All-or-Nothing (AoN)** variant, every payment must fully cover the relevant debt; every payment variable must be for the full amount, or zero. That is, $p_e^t \in \{D_a(e),0\}$ for every edge e.

For example, the instance of Fig. 2 has the following valid schedules:
- (In all variants) the one above in which $p_{u,v}^1 = p_{v,w}^1 = 1$ (all debts are paid in full at time 1).
- (In all variants) one in which $p_{u,v}^2 = p_{v,w}^2 = 1$ (all debts are paid in full at time 2). Under this schedule, node v is **bankrupt** at time 1, as €1 of the debt $D(v,w,0)$ is unpaid and that debt is **due**.
- (In the FP variant only) for every $a \in \mathbb{Q}$, where $0 < a < 1$, the schedule in which $p_{u,v}^1 = p_{v,w}^1 = a$ and $p_{u,v}^2 = p_{v,w}^2 = 1-a$. Under each of these, node v is **bankrupt** at time 1, as €$1-a$ of the debt $D(v,w,0)$ is unpaid and that debt is **due**.

Instant forwarding and cycles. We emphasize that we allow a bank to instantly spend income received. Note that in all valid schedules for the instance in Fig. 2 above, v *instantly forwards* money received from u to w; the assets of v never exceed 0 in any valid schedule. This behavior is consistent with the EN model [7] in which financial entities operate under a single clearing authority. Indeed, in such cases a payment chain of any length is admitted and the payment takes place in unit time regardless of chain length. Furthermore, still consistent with the EN model is the possibility of a payment cycle.



(a) Every node starts with €1. (b) Only u starts with €1. (c) Every node starts with €0.

Fig. 3: Examples illustrating the behavior of cycles in the IDM.

Fig. 3 shows three cyclic IDM instances, all with lifetime $T=2$. By our definition of a valid schedule, the schedule $p^1_{u,v}=p^1_{v,w}=p^1_{w,x}=p^1_{x,u}=1$ is valid in all three instances. In Fig. 3b we may imagine the €1 moves from node u along the cycle, statisfying every debt at time 1. This is a useful abstraction, but not strictly accurate - rather, we may imagine that all 4 banks simultaneously order payments forward under a single clearing system. The clearing system calculates the balances that each bank would have with those payments executed, ensures they are all nonnegative (one of our criteria for schedule validity) and then executes the transfer by updating all accounts simultaneously. This distinction is significant when we consider Fig. 3c, in which no node has assets. A clearing system ordered to simultaneously pay all debts would have no problem doing so in the EN model, and in our model this constitutes a valid schedule. We highlight that there also exist valid schedules for the instance in Fig. 3c in which all 4 banks go bankrupt, namely the schedule in which every payment variable is set to zero; then no bank is withholding (they all have zero assets), so the schedule is valid, and every bank has an overdue debt.

Lemma 1. *For any given IDM instance and a (FP/PP/*AoN*) schedule, it is possible to check in polynomial time whether the schedule is valid for that instance, and to compute the number of bankruptcies under the schedule.*

Proof sketch. It is possible to iterate over the schedule once and calculate: the assets of every node at every time; the number of debts which are overdue; the number of nodes which have overdue debts. The validity of the schedule (no withholding banks, non-negative assets at every time, no overpaid debts, and no debts paid early, FP/PP/AoN constraint) can similarly be verified in a single iteration over the schedule. □

Definition 4. *Let* (G,D,A^0) *be an instance. Then the set of timestamps* $\{t : D_{t_1}(e) = t$ *or* $D_{t_2}(e) = t$ *for some edge* $e\}$ *is the set of* extremal *timestamps.*

Remark 1. There is a simple preprocessing step such that we can assume afterwards that the lifetime T is polynomially bounded in the input size. This preprocessing step modifies the instance such that every $t \in [T]$ is an extremal timestamp. Observe that this procedure does not make any previously impossible schedule outcome (number of bankruptcies and finishing assets) possible, nor does it make any previously possible outcome impossible.

Hence we need not consider pathological cases in which the lifetime (and so the size of schedules) is exponential in the size of the input.

2.3 Problem definitions

Here, we define some problems with natural real-world applications in the IDM.

IDM Bankruptcy Minimization
Input: an IDM instance (G,D,A^0) and integer k
Question: does there exist a valid schedule S for the input such that at most k banks go bankrupt (have overdue debts) at any point in the schedule?

IDM Perfect Schedule
Input: an IDM instance (G,D,A^0)
Question: does there exist a valid schedule S for the input such that no debt is ever overdue?

This problem is equivalent to IDM Bailout Minimization where $b=0$ and to IDM Bankruptcy Minimization where $k=0$.

IDM Bailout Minimization
Input: an IDM instance (G,D,A^0) and integer b
Question: does there exist a positive bailout vector $B=(b_1,b_2,...b_{|V|})$ with $\sum_{i\in|V|} b_i \leq b$ and schedule S such that S is a perfect schedule for the instance (G,D,A^0+B)?

IDM Bankruptcy Maximization
Input: an IDM instance (G,D,A^0) and integer k
Question: does there exist a valid schedule S for the input such that **at least** k banks go bankrupt (have overdue debts) at any point in the schedule?

This problem is interesting to consider for quantifying a "worst-case" schedule, where banks' behavior is unconstrained beyond the terms of their debts.

All of the problems above exist in the **All-or-Nothing (AoN)** variant, where an AoN schedule is required; in the **Partial Payments (PP)** variant, in which a PP schedule is required; and in the **Fractional Payments (FP)** variant, in which an FP schedule is required.

All of the above problems, in all three variants, are in NP. For every yes-instance, there exists a witness schedule polynomial in the size of the input the validity of which can be verified in polynomial time by Lemma 1.

Every valid **PP** schedule is a valid **FP** schedule. Not every valid **AoN** schedule is a valid **PP** schedule. In an **AoN** schedule, a bank may go bankrupt while still having assets (insufficient to pay off any of its debts) - this is prohibited in any **PP** schedule as that bank would be **withholding**. If we restrict the input to only those in which for every edge e $D_a(e)=1$ then every valid **AoN** schedule for that instance is a valid **PP** schedule and a valid **FP** schedule.

We call a graph *multiditree* whenever the underlying undirected graph (i.e. the undirected graph that is obtained by replacing each directed multiedge with an undirected edge) is a tree. We call *rooted out-tree* (or *out-tree*) a multiditree in which every edge is directed away from the root. By an *out-path* we mean an out-tree where the underlying undirected graph is a path, and the root is either of the endpoints.

Problem \ Constraint on graph G	out-tree	multiditree	DAG	general case
FP Bailout Minimization and FP Perfect Scheduling	P (Thms. 8,9)	P (Thm. 9)	P (Thm. 9)	P (Thm. 9)
FP Bankruptcy Minimization	?	?	NP-C (Thm. 1)	NP-C (Thm. 1)
PP Bailout Minimization and PP Perfect Scheduling	P (Thm. 8)	NP-C (Thm. 3)	NP-C (Thms. 2,4)	NP-C (Thms. 2,3,4)
PP Bankruptcy Minimization	?	NP-C (Thm. 3)	NP-C (Thm. 1)	NP-C (Thms. 1-3,4)
PP Bankruptcy Maximization	?	?	NP-C (Thm. 5)	NP-C (Thm. 5)
AoN, all problems	NP-C (Thms. 6+7)	NP-C (Thms. 6+7)	NP-C (Thms. 6+7)	NP-C (Thms. 6+7)

Table 1: Summary of results. Note that Perfect Scheduling is a subproblem of both Bailout Minimization and Bankruptcy Minimization.

3 Our Results

In this section we investigate the complexity of the problems presented. We first present our hardness results in Subsection 3.1, and then show in Subsection 3.2 that under certain constraints the problem of Bailout Minimization becomes tractable.

3.1 Hardness results

Here we show that every problem introduced is NP-complete in the PP and AoN variants, even in various special cases, and that Bankruptcy Minimization is NP-complete and para-NP-Hard in all three variants for a variety of possible parameters.

Theorem 1. *For each of the variants* AoN, PP, *and* FP, Bankruptcy Minimization *is (i) NP-complete, even when the underlying graph G has $O(1)$ vertices, and (ii) NP-complete, even when $T=1$, the underlying graph G is a DAG with a longest path of length 4, out-degree at most 2, in-degree at most 3, debt at most €3 per edge, and starting assets at most €3 per bank.*

By Theorem 1(i) it follows that each of the AoN, PP, and FP variants of Bankruptcy Minimization are *para-NP-hard*, when parameterized by any parameter that is upper-bounded by the number of vertices, such as e.g. the number of bankruptcies k, or the treewidth of the underlying graph.

Theorem 2. AoN Perfect Scheduling *and* PP Perfect Scheduling *are both NP-complete even when* $T \leq 3$*, the underlying graph* G *is a DAG with out-degree at most 3, in-degree at most 3, debt at most* €*2 per edge, and starting assets at most* €*3 per bank.*

Theorem 3. PP Perfect Scheduling *is NP-complete even when the input is restricted to multiditrees with diameter 6, to* €1 *debts, and to a maximum of 6 multiedges between any two nodes.*

In all the above results, the input is allowed to have unlimited (i.e. unbounded) total assets in the system, which might be unrealistic in practically relevant financial systems. We now show that, even in the highly restricted case where just €1 in liquid assets exists in the system, PP Perfect Scheduling still remains NP-complete.

Theorem 4. AoN Perfect Scheduling *and* PP Perfect Scheduling *are both NP-complete even when* $sum(A^0) = 1$*, i.e. the total of all external assets is* €*1.*

Proof. We proceed by reduction from Directed Hamiltonian cycle (DHC), an NP-complete problem [12]:

Input: a digraph $G = (V,E)$.

Question: does there exist a DHC on G (a directed cycle which visits every vertex exactly once)?

For this reduction, we introduce the **at-least-once** gadget shown in Fig. 4. The intuition of the proof is that there is only €1 in the system, and that in any perfect schedule that €1 must pass through each gadget at least once, and therefore exactly once since there are n such gadgets and n timesteps the €1 can "rest" at a **center node** v_C.

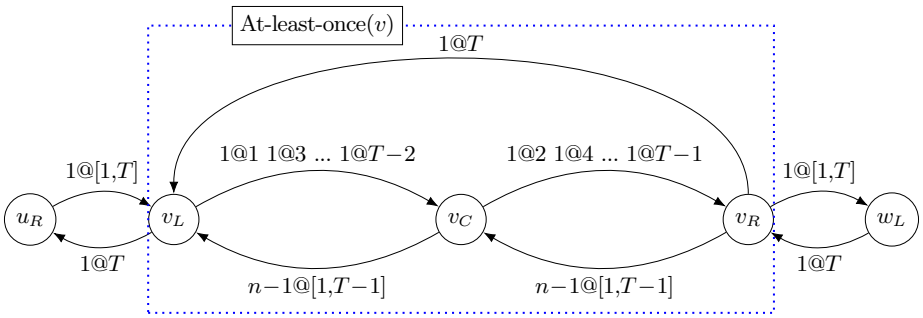


Fig. 4: The **at-least-once** gadget for node $v \in V(G_{DHC})$ where $\{(u,v),(v,w)\} \subseteq E(G_{DHC})$. Note the lifetime $T = 2|V(G_{DHC})|+1$.

Given a DHC instance G_{DHC}, construct a PP Perfect Scheduling instance (G_{PS}, D, A^0) by introducing a copy of the *at-least-once* gadget for each node $v \in G_{DHC}$ and then connecting gadgets within G_{PS} iff there exists a directed edge from one of the corresponding nodes in G_{DHC} to the other, as shown in Fig. 4. We then give €1 in assets to exactly onearbitrarily-chosen *right node* v_R in G_{PS}, and €0 in assets to every other node.

Claim 1. *If the IDM instance G_{PS}, D, A^0 admits a perfect schedule, then the DHC instance G_{DHC} admits a directed Hamiltonian cycle.*

Claim 2. *If the DHC instance G_{DHC} admits a directed Hamiltonian cycle, then G_{PS} admits a perfect schedule.*

Proof sketch. Given a DHC v_1, v_2, ..., v_n, v_1, we describe the order in which the €1 in "real" assets moves through the network in our constructed perfect schedule. All other payments in the schedule can be efficiently found by debt cancellation, i.e. some node u pays some node v some amount €1 at time t and v pays u the same amount €1 also at time t, resulting in no "real" asset movement. This is possible by construction of the instance (G_{PS}, D, A^0) - in general, there is no guarantee that v would also have an active debt to u at time t. The €1 starts in node v_{1R}, then is paid forward in the order indicated by the edge labels in Figure 5. We emphasize once more that the "payment" around the entire cycle at time $2n+1$ does not result in any "real" asset movement – all balances remain unchanged. □

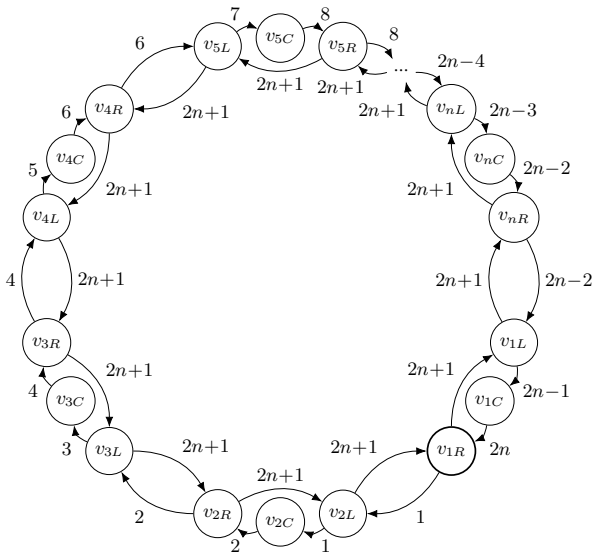


Fig. 5: The path that the "real" €1 takes in our constructed PP Perfect Scheduling instance, if the input graph G contained a Hamiltonian cycle.

Hence we have that there exists a perfect schedule in (G_{PS},D,A^0) iff there exists a Hamiltonian cycle in G_{DHC}. This completes the proof of Theorem 4. □

Theorem 5. AoN Bankruptcy Maximization *and* PP Bankruptcy Maximization *are both NP-complete even when $T=2$, the underlying graph G is a DAG with out-degree at most 2, in-degree at most 3, debt at most €2 per edge, and assets at most €3 per bank.*

The constraints imposed by AoN schedules rapidly increase the problem complexity. Indeed, every problem considered is NP-complete, even whenever the input graph is an out-path on at most 4 vertices and with lifetime $T \leq 2$.

Theorem 6. *When the underlying graph G is an out-path on 4 vertices* AoN Perfect Scheduling *is weakly NP-Hard when the lifetime $T=2$, and strongly NP-Hard when T is unbounded.*

Theorem 7. AoN Bankruptcy Maximization *is NP-Hard even when the underlying graph G is an out-path on 3 vertices and the lifetime T of the graph is at most 2.*

3.2 Polynomial-time algorithms

In this section we show that the PP variant of Bailout Minimization is solvable in polynomial time on out-trees, while its FP variant is always polynomial-time solvable. Our algorithm for the PP variant contrasts with the NP-completeness of its subproblem PP Perfect Scheduling on both DAGs (Theorem 2) and multiditrees (Theorem 3); note that out-trees are a subclass of both DAGs and multiditrees.

Theorem 8. PP Bailout Minimization *is in P when the input is restricted to out-trees.*

Proof sketch. We show that, given an instance of PP Bailout Minimization G, D, A^0, b in which G is an out-tree on at least 3 nodes, it is always possible to produce an updated instance which is equivalent (a yes-instance iff the original instance was a yes-instance) and in which G is strictly smaller (has fewer nodes). The process is illustrated in Fig. 6, and is as follows:

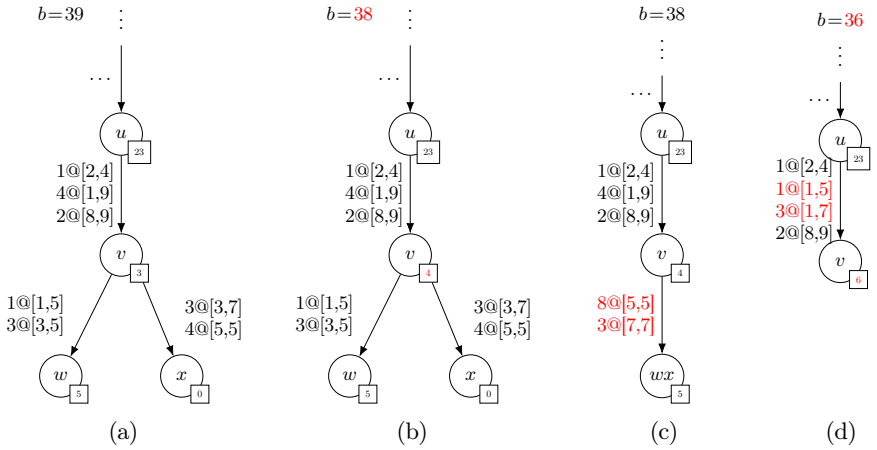


Fig. 6: Example of shrinking an out-tree while preserving (non)existence a bailout schedule.

1. While any node v is insolvent, increment v's assets and decrement the bailout amount b. In Fig. 6a, for instance, node v has €7 in income and €3 in external assets, but €11 in debt, so is insolvent. In Fig. 6b v's assets have increased by €1 and the bailout amount has decreased by €1.
2. Update every debt from a parent to its leaf children to be a 1-interval at the end timestep as, so a debt due in the interval [3,7] is updated to be due at time 6 exactly (i.e. at [7,7]).
3. While any parent has two leaf children (i.e. in Fig. 6b v has two leaf children w and x), **merge** the two sibling leaves into a single leaf (i.e. in Fig. 6c w and x are combined into wx). Sum node assets, and combine debts due at the same time into a single debt (i.e. v's debts 1@[6,6] to w and 4@[6,6] to x are combined into a single debt 5@[6,6] to wx).
4. While there exist 3 nodes j, k, l with l a leaf and the only child of k, and k a child of j **prune** l. In Fig. 6c $j=u$, $k=v$, and $l=wx$, and we prune wx. That is, update the debts from j to k to reflect exactly the constraints imposed by k's debts to l, then remove l from the graph. If payments from j cannot cover all payments to l in time (i.e. k is necessarily illiquid), increment k's assets and decrement the bailout amount b until they do, then update the debts. (Binary search may be applied where debt amounts are very large.)

   In Fig. 6d, the 8@[5,5] debt to wx will be paid using €6 of v's external assets (including €1 of "solvency" bailout and €2 of "liquidity" bailout), €1 from u's debt 1@[2,4], and €1 of u's debt at [1,9], now constrained to the interval [1,5]. The 3@[7,7] debt to wx will be paid using €3 of u's debt at [1,9], now constrained to the interval [1,7]. The money received at v in the interval [8,9] cannot be usefully directed to wx, as the earliest it could be received ($t=8$) is still later than the latest debt from v ($t=7$).
5. If the instance has more than two nodes, loop to step 2. Otherwise, if the bailout amount $b<0$, reject, else accept. We emphasize the root necessarily has enough assets to pay all its debts, because every node is solvent after step 1.

Note that the algorithm presented loops $|V(G)|$ times in the worst case (i.e. when the input is a path), and each step runs in polynomial time. □

Theorem 9. FP Bailout Minimization *is in P.*

Proof sketch. We show that an instance of FP Bailout Minimization G, D, A^0, b can be encoded in a linear program. The constraints are all expressed as linear expressions:
- $\text{sum}(B) \leq b$.
- The set of *payment variables* p_e^t is defined as in subsection 2.2.
- The definitions of I_v^t, O_v^t, and e_v^t are all linear combinations of *payment variables*. Set $e_v^0 := A^0[v] + B[v]$; that is, the starting assets of v are its *external assets* combined with the *bailouts received* at v under the vector B.
- The constraints on valid schedules in Definition 1 can all also be expressed as linear combinations.
- We additionally impose that no banks are bankrupt under the schedule, or equivalently that no debt is ever overdue. That is, for every edge e and debt $D(e) = (a, t_1, t_2)$, $\sum_{t \in [t_1, t_2]} p_e^t = a$.

Any assignment to B and to the payment variables satisfying the above is necessarily a valid perfect schedule on an instance in which starting assets of nodes were supplemented by at most €b in total.

Linear programs can be efficiently solved when fractional solutions are admitted, hence all instances of FP Bailout Minimization are tractable. We emphasize that this

is in contrast with PP Bailout Minimization and FP Bankruptcy Minimization, both of which are NP-complete. The method above solves neither of these: the former would correspond to an integer linear program (which are NP-complete in general) and it is not possible to express a constraint on the *number of bankruptcies* through a linear combination on the payment variables. □

4 Conclusion and open problems

This paper introduces the *Interval Debt Model (IDM)*, a new model seeking to capture the temporal aspects of debts in financial networks. We investigate the computational complexity of various problems involving debt scheduling, bankruptcy and bailout with different payment options (All-or-nothing (AoN), Partial (PP), Fractional (FP)) in this setting. We prove that many variants are hard even on very restricted inputs but certain special cases are tractable. For example, we present a polynomial time algorithm for PP Bailout Minimization where the IDM graph is an out-tree. However, for a number of other classes (DAGs, multitrees, total assets are €1), we show that the problem remains NP-hard. This leaves open the intriguing question of the complexity status of problems which are combinations of two or more of these constraints, most naturally on multitrees which are also DAGs, an immediate superclass of our known tractable case.

We prove that FP Bailout Minimization is polynomial-time solvable by expressing it as a Linear Program. Can a similar argument be applied to some restricted version of FP Bankruptcy Minimization (which is NP-Complete, in general)? A natural generalization is simultaneous Bailout and Bankruptcy minimization i.e. can we allocate €b in bailouts such that a schedule with at most k bankruptcies becomes possible. Variations of this would be of practical interest. For example, if regulatory authorities can allocate bailouts as they see fit, but not impose specific payment times, it would be useful to consider the problem of allocation of €b in bailouts such that the maximum number of bankruptcies in any valid schedule is at most k. Conversely, where financial authorities can impose specific payment times, the combination of the problems Bankruptcy Minimization and Bailout Minimization would be more applicable.

Finally, can we make our models even more realistic and practical? How well do our approaches perform on real-world financial networks? Can we identify topological and other properties of financial networks that may be leveraged in designing improved algorithms?

Acknowledgements The authors are thankful to Roger Wattenhofer for insightful discussions and suggestions and to Nina Klobas and Tamio-Vesa Nakajima for technical discussions early in the project.

References

1. Akrida, E.C., Mertzios, G.B., Spirakis, P.G., Zamaraev, V.: Temporal vertex cover with a sliding time window. Journal of Computer and System Sciences **107**, 108–123 (2020)
2. Bagehot, W.: Lombard street: a description of the money market. King (1873)
3. Bardoscia, M., Barucca, P., Battiston, S., Caccioli, F., Cimini, G., Garlaschelli, D., Saracco, F., Squartini, T., Caldarelli, G.: The physics of financial networks. Nature Reviews Physics pp. 1–18 (2021)

4. Casteigts, A., Flocchini, P.: Deterministic Algorithms in Dynamic Networks: Formal Models and Metrics. Tech. rep., Defence R&D Canada CR 2013-020 (April 2013), https://hal.archives-ouvertes.fr/hal-00865762
5. Casteigts, A., Flocchini, P.: Deterministic Algorithms in Dynamic Networks: Problems, Analysis, and Algorithmic Tools. Tech. rep., Defence R&D Canada CR 2013-021 (April 2013), https://hal.archives-ouvertes.fr/hal-00865764
6. Eisenberg, L.: A summary: Boolean networks applied to systemic risk. Neural Networks in Financial Engineering pp. 436–449 (1996)
7. Eisenberg, L., Noe, T.H.: Systemic risk in financial systems. Management Science **47**(2), 236–249 (2001)
8. Enright, J., Meeks, K., Mertzios, G.B., Zamaraev, V.: Deleting edges to restrict the size of an epidemic in temporal networks. Journal of Computer and System Sciences **119**, 60–77 (2021)
9. Haldane, A.G., May, R.M.: Systemic risk in banking ecosystems. Nature **469**(7330), 351–355 (2011)
10. Holme, P., Saramäki, J. (eds.): Temporal Networks. Springer (2013)
11. Jackson, M.O., Pernoud, A.: Systemic risk in financial networks: A survey. Annual Review of Economics **13**(1), 171–202 (2021)
12. Karp, R.M.: Reducibility among Combinatorial Problems, pp. 85–103. Springer US, Boston, MA (1972)
13. Kempe, D., Kleinberg, J.M., Kumar, A.: Connectivity and inference problems for temporal networks. In: Proceedings of the 32nd annual ACM symposium on Theory of computing (STOC). pp. 504–513 (2000)
14. Klobas, N., Mertzios, G.B., Molter, H., Niedermeier, R., Zschoche, P.: Interference-free walks in time: Temporally disjoint paths. In: Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI). pp. 4090–4096 (2021)
15. Papachristou, M., Kleinberg, J.: Allocating stimulus checks in times of crisis. In: Proceedings of the ACM Web Conference 2022. pp. 16–26 (2022)
16. Papp, P.A., Wattenhofer, R.: Sequential defaulting in financial networks. In: 12th Innovations in Theoretical Computer Science Conference ITCS. pp. 52:1–52:20 (2021)
17. Rochet, J.C., Vives, X.: Coordination failures and the lender of last resort: was Bagehot right after all? Journal of the European Economic Association **2**(6), 1116–1147 (2004)
18. Rogers, L.C., Veraart, L.A.: Failure and rescue in an interbank network. Management Science **59**(4), 882–898 (2013)
19. Schuldenzucker, S., Seuken, S., Battiston, S.: 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. vol. 67, pp. 32:1–32:20 (2017)
20. Tesfaye, B., Augsten, N., Pawlik, M., Böhlen, M., Jensen, C.: Speeding up reachability queries in public transport networks using graph partitioning. Information Systems Frontiers **24**, 11–29 (2022)