Theoretical Computer Science 1028 (2025) 115028

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Payment scheduling in the Interval Debt Model ☆

Tom Friedetzky, David C. Kutner,*, George B. Mertzios,[1], Iain A. Stewart, Amitabh Trehan

Department of Computer Science, Durham University, Upper Mountjoy Campus, Stockton Road, Durham DH1 3LE, UK

ARTICLE INFO

Keywords:
Temporal graph
Financial network
Payment scheduling
Computational complexity

ABSTRACT

The network-based study of financial systems has received considerable attention in recent years but has seldom explicitly incorporated the dynamic aspects of such systems. We consider this problem setting from the temporal point of view and 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 payment schedule with at most/at least a given number of bankruptcies; PERFECT SCHEDULING, the special case of the minimization variant where the aim is to produce a schedule with no bankruptcies (that is, a perfect schedule); and BAILOUT MINIMIZATION, in which a financial authority must allocate a smallest possible bailout package to enable a perfect schedule. We show that each of these problems is NP-complete, in many cases even on very restricted input instances. On the positive side, we provide for PERFECT SCHEDULING a polynomial-time algorithm on (rooted) out-trees although in contrast we prove NP-completeness on directed acyclic graphs, as well as on instances with a constant number of nodes (and hence also constant treewidth). When we allow non-integer payments, we show by a linear programming argument that the problem BAILOUT MINIMIZATION can be solved in polynomial time.

1. Introduction

A natural problem in the study of financial networks is that of whether and where a failure will occur if no preventative action is taken. We focus specifically on the flexibility that financial entities are afforded as regards the precise timing of their outgoings and for this purpose introduce the *Interval Debt Model (IDM)* in which a set of financial entities is interconnected by debts due within specific time intervals. In the IDM, a *payment schedule* specifies timings of payments to serve the debts. We examine the computational hardness of determining the existence of a schedule of payments with "good" properties, e.g., no or few bankruptcies, or minimizing the scale of remedial action. In particular, we establish how hardness depends on variations in the exact formalism of the model (to allow some small number of bankruptcies or insist on none at all) and on restrictions on the structure or lifetime of the input instance. A unique and novel feature of the IDM is its capacity to capture the temporal aspects of real-world financial systems; previous work has seldom explicitly dealt with this intrinsic facet of real-world debt.

☆ This article belongs to Section A: Algorithms, automata, complexity and games, Edited by Paul Spirakis.

* Corresponding author.
E-mail addresses: tom.friedetzky@durham.ac.uk (T. Friedetzky), david.c.kutner@durham.ac.uk (D.C. Kutner), george.mertzios@durham.ac.uk (G.B. Mertzios), i.a.stewart@durham.ac.uk (I.A. Stewart), amitabh.trehan@durham.ac.uk (A. Trehan).

[1] Partially supported by the EPSRC grant EP/P020372/1.

https://doi.org/10.1016/j.tcs.2024.115028
Received 18 March 2024; Received in revised form 17 October 2024; Accepted 6 December 2024

Available online 13 December 2024

Financial networks Graph theory provides models for many problems of practical interest for analyzing (or administering) financial systems. For example, Eisenberg and Noe's work [1] abstracts a financial system to be a weighted digraph (in which each node is additionally labelled according to the corresponding entity's assets). The authors of that work are focused on the existence and computation of a *clearing vector*, which is essentially a set of payments among nodes of the graph which can be executed synchronously without violating some validity constraints. Their model provides the basis for much subsequent work in the network-based analysis of financial systems: it has been adapted to incorporate default costs [2], Credit Default Swaps [3] (CDSs) (that is, derivatives through which banks can bet on the default of another bank in the system) and the sequential behaviour of bank defaulting in real-world financial networks [4].

An axiomatic aspect of Eisenberg and Noe's model is the so-called *principle of proportionality*: that a defaulting bank pays off each of its creditors proportionally to the amount it is owed. Some recent work has considered alternative payment schemes, which allow, for example, paying some debts in full and others not at all (so-called *non-proportional payments*). For example, Bertschinger, Hoefer and Schmand [5] study financial networks in a setting where each node is a rational agent which aims to maximize flow through itself by allocating its income to its debts. The focus of that work is on game-theoretic questions, such as the price of anarchy, or the existence, properties, and computability of equilibria. Papp and Wattenhoffer [6] also study a non-proportional setting, additionally incorporating CDSs.

Complementing the decentralized, game-theoretic approach is the question of the (centralized) computability of a *globally* "good" outcome through bailout allocation [7] (also called cash injection [8]), or timing default announcements [4], among other operations. In such works, the prototypical objective is to minimize the number of bankruptcies; related measures include total market value [7], or systemic liquidity [8]. Egressy and Wattenhoffer [7] focus solely on computational complexity of leveraging bailouts to optimize a range of objectives, in a setting which incorporates proportional payments and default costs. Kanellopoulos, Kyropoulou and Zhou [8,9] apply both a game-theoretic and a classical complexity perspective to two mechanisms: debt forgiveness (deletion of edges in the financial network) and cash injection (bailouts). Notably, in this work the central authority may remove debts in a way which may be detrimental to certain individuals, but beneficial to the total systemic liquidity.

Previous research on financial networks has also drawn from ecology [10], statistical physics [11] and Boolean networks [12].

A central 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, e.g., debts owed to it) to meet all its obligations; and it is said to be *liquid* if it has enough liquid assets (that is, cash) to meet its obligations on time. An illiquid but solvent bank may exist even in modern interbank markets [13]. In such cases, a central bank may act as a *lender of last resort* and extend loans to such banks to prevent them defaulting on debts [13,14]. 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 a government) to entities to help them avoid bankruptcy.

Temporal graphs Temporal graphs are graphs whose underlying connectivity structure changes over time. Such graphs allow us to model real-world networks which have inherent dynamic properties, such as transportation networks [16], contact networks in an epidemic [17,18] and communication networks; for an overview see [19,20]. Most commonly, following the formulation introduced by Kempe, Kleinberg and Kumar [21], a temporal graph has a fixed set of vertices together with edges that appear and disappear at integer times up to an (integer) lifetime. 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 [22], and finding a temporal vertex cover remains NP-complete even on star temporal graphs [23].

Contributions In this paper we present a novel framework, the *Interval Debt Model* (IDM), for considering problems of bailout allocation and payment scheduling in financial networks by using temporal graphs to account for the isochronal aspect of debts between financial entities (previous work has almost exclusively focused on static financial networks). In particular, the IDM offers the flexibility that entities can pay debts earlier or later, within some agreed interval. 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).

Our work explores the natural question of whether and how payments can be scheduled to avert large-scale failures in financial networks. Broadly, we establish that computing a zero-failure schedule (a *perfect schedule*) is NP-complete even when the network topology is highly restricted, unless we admit fractional payments, in which case determining the existence of a perfect schedule is tractable in general. Interestingly, if we allow a small number k of bankruptcies to occur then every problem variant is computationally hard even on inputs with $O(1)$ nodes. This can be thought of analogously to Max 2SAT being strictly harder than 2SAT (unless $\mathrm{P} = \mathrm{NP}$) despite being a "relaxation": in Max 2SAT we allow up to k clauses to not be satisfied. Furthermore, in the setting where we insist not only on payments being for integer amounts but more strongly that any payment is for the full amount of the corresponding debt, finding a perfect schedule is NP-complete even if there are only four nodes.

We begin by introducing, first by example and then formally, the Interval Debt Model in Section 2. In Section 3 we present our results: in Section 3.1–3.3 we establish some sufficient criteria for NP-hardness for each of the problems we consider; and in Section 3.4 we present two polynomial-time algorithms. Our conclusions and directions for further research are given in Section 4.

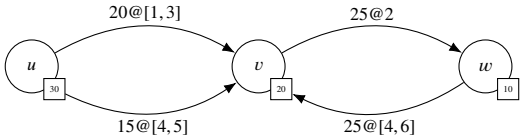


Fig. 1. A simple instance of the Interval Debt Model (IDM). Numbers in square boxes represent the initial external assets of the node (for example, €30 for node u), directed edges represent debts, and the label on an edge represents the terms of the associated debt (for example, u must pay v €20 between time 1 and time 3).

2. The Interval Debt Model

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

2.1. An illustrative example

As an example, consider a tiny financial network consisting of the 3 banks u, v and w with €30, €20 and €10, respectively, in initial external assets and where there are 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 at time 4 or time 5 (note that all payments must be made at integer times)
- bank v must pay bank w a debt of €25 at time 2 exactly
- bank w must pay €25 to bank v between times 4 and 6 (that is, at time 4, 5 or 6).

A graphical representation of this system is shown in Fig. 1 (the descriptive notation used should be obvious and is retained throughout the paper).

Several points can be made about this system: node u is *insolvent* as its €30 in initial external assets are insufficient to pay all its debts; node v may be *illiquid* for 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. The choices made by the various banks, in the form of a (payment) schedule, clearly affect the status of the overall financial system. Note that solvency is determined solely by whether sufficient funds exist whereas liquidity depends upon when debts are paid and owed.

One may ask several questions about our toy financial system such as: Are partial payments allowed (e.g., u paying €18 of the €20 debt at time 1, and the rest later)? If so, are non-integer payments allowed? Can money received be immediately forwarded (e.g., u paying v €20 at time 2 and v paying w €25 at time 2)? Does v necessarily have to pay its debt to w at time 2 if it has the liquid assets to do so? We now expand upon these questions and specify in detail the setting we consider in the remainder of the paper. Note that throughout the paper we use the euro € as our monetary unit of resource even though, as we will see, we have a variant of the Interval Debt Model within which payments can be made for any rational fraction of a euro. We often prefix monetary payments with the symbol € to make our proofs more readable.

2.2. Formal setting

Formally, an *Interval Debt Model* (*IDM*) instance is a 3-tuple (G, D, A^0) as follows.

- $G = (V, E)$ is a finite digraph with the set of n nodes (or, alternatively, *banks*) $V = \{v_i : i = 1, 2, \ldots, n\}$ and the set of m directed labelled edges $E \subseteq V \times V \times \mathbb{N}$, with the edge $(u, v, id) \in E$ denoting that there is an edge, or *debt*, whose label is id, from the *debtor* u to the *creditor* v. We can have multi-edges but the labels of the edges from some node u to some node v must be distinct and form a contiguous integer sequence $0, 1, 2, \ldots$. 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. We also refer to the undirected graph obtained from G by ignoring the orientations on directed edges as the *footprint* of G.
- $D : E \to \{(a, t_1, t_2) : a, t_1, t_2 \in \mathbb{N} \setminus \{0\}, t_1 \leq t_2\}$ is the *debt function* which associates *terms* to every debt (ordinarily, we abbreviate $D((u, v, id))$ as $D(u, v, id)$). Here, if e is a debt with terms $D(e) = (a, t_1, t_2)$ then a is the *monetary amount* (or *monetary debt*) 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. For any debt $e \in E$, we also 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 $a@[t_1, t_2]$ or by $a@t_1$ when $t_1 = t_2$ (as we did in Fig. 1); also, for simplicity, we sometimes just refer to $a@[t_1, t_2]$ as the debt.
- $A^0 = (c^0_{v_1}, c^0_{v_2}, \ldots c^0_{v_n}) \in \mathbb{N}^n$ is a tuple with $c^0_{v_i}$ denoting the *initial external assets* (i.e. starting cash) of bank v_i.

We refer to the greatest time-stamp T that appears in any debt for a given instance as the *lifetime* and assume that all network activity ceases after time T. The instance shown in Fig. 1, which has lifetime $T = 6$, is formally 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 $A^0 = (c^0_u, c^0_v, c^0_w)$, where $c^0_u = 30$, $c^0_v = 20$ and $c^0_w = 10$. 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 $A^0 = (c^0_u, c^0_v, c^0_w)$, where $c^0_u = 1$, $c^0_v = 0$ and $c^0_w = 0$.

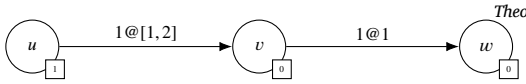


Fig. 2. An IDM instance for which every schedule is described by four payment values $p^1_{(u,v,0)}$, $p^1_{(v,w,0)}$, $p^2_{(u,v,0)}$ and $p^2_{(v,w,0)}$.

The *size* of the instance (G, D, A^0) is defined as $n + m + \log(T) + b$, where b is the maximum number of bits needed to encode any of the (integer) numeric values appearing as the monetary amounts in the debts. Note that in what follows, we usually do not mention the label id of a debt (u, v, id) but just refer to the debt as (u, v) when this causes no confusion.

2.3. Schedules

Given an IDM instance (G, D, A^0), a (*payment*) *schedule* σ describes the times at which the banks transfer *assets* to one another via *payments*. Formally, a schedule σ is a set of $|E|T$ *payment values* $p^t_e \geq 0$, one for each edge-time pair (e, t) (note that no payments are made at time 0). Equivalently, a schedule can be expressed as an $|E| \times T$ matrix S with the payment values p^t_e the entries of the matrix. The value p^t_e is the monetary amount of the debt e paid at time t. Our intention is that at any time $1 \leq t \leq T$, every payment value $p^t_e > 0$ of a schedule σ is paid by the debtor of e to the creditor of e, not necessarily for the full monetary amount $D_a(e)$ but for the amount p^t_e. A schedule for the instance of Fig. 2 consists of the four payments values $p^1_{(u,v,0)}$, $p^1_{(v,w,0)}$, $p^2_{(u,v,0)}$ and $p^2_{(v,w,0)}$. Note that, using the above representation of a schedule σ, 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 detailing the non-zero payments. An example schedule for the IDM 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 $u, v \in V$ and time $0 \leq t \leq T$, the following values are with respect to some specific schedule.

- Denote by I^t_v the total monetary amount of incoming payments of node v at time t.
- Denote by O^t_v the total monetary amount of outgoing payments (expenses) of node v at time t.
- We write $p^t_{u,v}$ to denote the total amount of all payments made from debtor u to creditor v at time t in reference to *all* debts from u to v; that is, $p^t_{u,v} = \sum_i p^t_{(u,v,i)}$.
- The vector $A^0 = (c^0_{v_1}, c^0_{v_2}, \ldots, c^0_{v_n})$ specifies the initial external assets (cash) of each node at time 0. For $t > 0$, we denote by c^t_v node v's cash assets at time t; that is, $c^t_v = c^{t-1}_v + I^t_v - O^t_v$.

For clarity, we refer to the starting cash of banks as "initial external assets" and to liquid assets in general as cash assets. By cash assets 'at time t' (resp. 'prior to time t') we mean after all (resp. before any) of the payments associated with time t have been executed. Cash assets at time 0 are then precisely the initial external assets at time 0 (possibly supplemented by some bailout, as we shall see later).

We have been a little vague so far as regards the form of the payment values in any schedule and have not specified whether these values are integral, rational or do not necessarily equal the full monetary amount of the debt. As we detail below, we have variants of the model covering different circumstances (with perhaps the standard version being when payment values are integral but do not necessarily equal the full monetary amount of the debt).

Recall the example schedule from Fig. 2, which we can represent as $p^1_{u,v} = 1$, $p^1_{v,w} = 1$. As we shall soon 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. However, there might be schedules that are not *valid*, as well as valid schedules in which banks default on debts (that is, go *bankrupt*). We deal with the key notions of validity and bankruptcy now.

Definition 1. A schedule is *valid* if it satisfies the following properties (for any debt e, terms $D(e) = (a, t_1, t_2)$ and node v):

- all payment values are non-negative; that is, $p^t_e \geq 0$, for $1 \leq t \leq T$
- all cash asset values (as derived from payment values and initial external assets) are non-negative; that is, $c^t_v \geq 0$, for $0 \leq t \leq T$
- no debts are overpaid; that is, $\sum_{t=1}^{T} p^t_e \leq a$
- no debts are paid too early; that is, $\sum_{t=1}^{t_1-1} p^t_e = 0$.

Given some IDM instance, some schedule and some debt e with terms $D(e) = (a, t_1, t_2)$, the debt e is said to be *payable* at any time in the interval $[t_1, t_2 - 1]$. At time t_2, e is said to be *due*. At time $t_2 \leq t \leq T$, if the full amount a has not yet been paid (including payments made at time t_2) then e is said to be *overdue* at time t. A debt is *active* whenever it is payable, due or overdue. However, a bank is said to be *withholding* if, at some time $1 \leq t \leq T$, it has an overdue debt and sufficient cash assets to pay (part of, where fractional or partial payments are permitted; see below) the debt. If any bank is withholding (at any time) in the schedule then the schedule is not valid.

So, for example and with reference to the IDM instance in Fig. 2, if, according to some schedule, bank u pays 1 to bank v at time 1 but v makes no payment to w at time 1 then v is withholding and the schedule is not valid.

Definition 2. With reference to some schedule, a bank is said to be *bankrupt* (at time t) if it is the debtor of an overdue debt (at time t). We say that a schedule has k *bankruptcies* if k distinct banks are bankrupt at some time in the schedule (the times at which these banks are bankrupt might vary). A bank may *recover* from bankruptcy if it subsequently receives sufficient income to pay off all its overdue debts.

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

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

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

We will not be concerned with the precise timing of bankruptcy or the recovery or not of any bank in this paper.

We now detail three variants of the model (alluded to earlier) in which different natural constraints are imposed on the payment values.

Definition 4. In what follows, e is an arbitrary debt and $1 \le t \le T$ some time.

- In the *Fractional Payments* (*FP*) variant, the payment values may take rational values; that is, $p_e^t \in \mathbb{Q}$ and we allow payments for a smaller amount than the full monetary amount of e.
- In the *Partial Payments* (*PP*) variant, the payment values may take only integer values; that is, $p_e^t \in \mathbb{N}$ and we allow payments for a smaller amount than the full monetary amount of e.
- In the *All-or-Nothing* (*AoN*) variant, every payment value must fully cover the relevant monetary amount of e; that is, every payment value must be for the full monetary amount of e or zero. So, $p_e^t \in \{D_a(e), 0\}$.

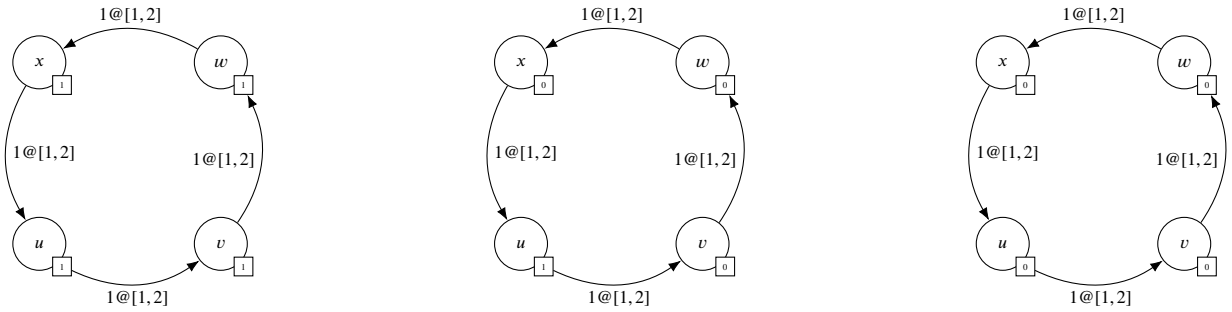
For example, the instance of Fig. 2 has the following valid schedules:

- (in all variants) the schedule above in which $p^1_{u,v} = p^1_{v,w} = €1$ (all debts are paid in full at time 1)
- (in all variants) the schedule in which $p^2_{u,v} = p^2_{v,w} = €1$ (all debts are paid in full at time 2)
  - under this schedule, node v is bankrupt at time 1 as €1 of the debt $(v, w, 0)$ is unpaid and that debt is overdue
- (in the FP variant only) for every $a \in \mathbb{Q}$, where $0 < a < 1$, the schedule in which $p^1_{u,v} = p^1_{v,w} = €a$ and $p^2_{u,v} = p^2_{v,w} = €1 - a$
  - under each of these schedules, node v is bankrupt at time 1 as $€1 - a$ of the debt $(v, w, 0)$ is unpaid and that debt is overdue.

It is worthwhile clarifying the concepts of instant forwarding and payment-cycles. We emphasize that we allow a bank to instantly spend income received. Note that in any valid schedule for the instance in Fig. 2, v *instantly forwards* money received from u to w (so as not to be withholding); so, the cash assets of v never exceed 0 in any valid schedule. This behaviour is consistent with the Eisenberg and Noe model [1] in which financial entities operate under a single clearing authority which synchronously executes payments. Indeed, in such cases a payment-chain of any length is permitted and the payment takes place instantaneously regardless of chain length.

Furthermore, and still consistent with the Eisenberg and Noe model, there is the possibility of a *payment-cycle* which is a set of banks $\{u_1, u_2, \ldots, u_c\}$, for some $c \ge 2$, with a set of debts $\{e_i = (u_i, u_{i+1}, l_i) : 1 \le i \le c-1\} \cup \{e_c = (u_c, u_0, l_c)\}$ so that at some time t, all debts are active yet none has been fully paid and where each bank makes a payment, at time t, of the same value a towards its debt. As an illustration, Fig. 3 shows three 'cyclic' IDM instances, all with lifetime $T = 2$. By our definition of a valid schedule, the schedule $p^1_{u,v} = p^1_{v,w} = p^1_{w,x} = p^1_{x,u} = €1$, forming a payment-cycle, is valid in all three instances. This is intuitive for Fig. 3a, where each node has sufficient initial external assets available to pay all its debts in full at any time, irrespective of income. In Fig. 3b, we may imagine that the €1 moves from node u along the cycle, satisfying every debt at time 1. This is a useful abstraction but not strictly accurate: rather, we should imagine that all four 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 non-negative (one of our criteria for schedule validity) and then executes the payments by updating all accounts simultaneously. This distinction is significant when we consider Fig. 3c in which no node has any initial external assets. A clearing system ordered to simultaneously pay all debts would have no problem doing so in the Eisenberg and Noe 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 four banks go bankrupt, one schedule being where all payments at any time are 0: here, no bank is withholding (they all have zero cash assets), so the schedule is valid, but every bank has an overdue debt and so is bankrupt.

We use payment-cycles throughout our constructions in a context such as that in Fig. 4. Here, a valid schedule is where all nodes pay their corresponding debts in full at time t. The effect is that the €1 of cash assets at node u is 'transferred' to €1 of cash assets at node v.



(a) All nodes start with €1. (b) Only u starts with €1. (c) All nodes start with €0.

Fig. 3. Examples illustrating the behaviour of cycles in the IDM. In all instances shown the schedule in which all nodes pay their debts in full at time 1 is valid.

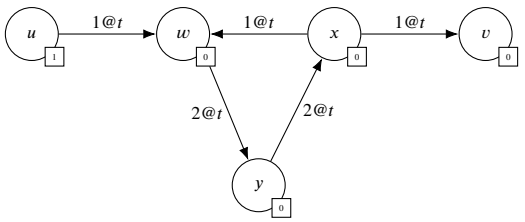


Fig. 4. Using a payment-cycle to effectively transfer €1 of assets from node u to node v.

2.4. Canonical instances

We wish to replace certain IDM instances with equivalent yet simpler ones. For example, consider the instance given in Fig. 1 but where every time-stamp in the instance is multiplied by a factor of 100 (so that, for example, the debt from w to v becomes $25@[400, 600]$). This 'inflated' instance is in essence equivalent to the original one but has a lifetime of 600.

Definition 5. Let (G, D, A^0) be an instance. Then the set of time-stamps $\{t : D_{t_1}(e) = t$ or $D_{t_2}(e) = t$, for some edge $e\}$ is the set of *extremal* time-stamps.

There is a simple preprocessing step such that we can assume that the lifetime T of any IDM instance is polynomially bounded in n and m (that is, the numbers of banks and debts, respectively). This preprocessing step modifies the instance such that every $1 \leq t \leq T$ is an extremal time-stamp with the process being to simply omit non-extremal time-stamps and then compact the remaining time-stamps. Observe that this procedure is such that any valid schedule in the original IDM instance can be transformed into a valid schedule in the compacted instance so that these schedules have the same number of bankruptcies, eventual assets and so forth, and vice versa when the compacted instance is expanded into the original instance. Hence, we need not consider pathological cases in which the lifetime is, say, exponential in the number of nodes and debts. Given this restriction, we can now revise the notion of the size of an IDM instance to say that it is $n + m + b$ where n is the number of banks, m is the number of debts and b is the maximum number of bits needed to encode any of the numeric values appearing as monetary amounts of debts.

Lemma 1. *For any given IDM instance and any schedule, in any of the FP, PP or AoN variants, it is possible in polynomial-time both to check whether the schedule is valid and to compute the number of bankruptcies under the schedule.*

Proof sketch. It is possible to iterate over the schedule once and calculate: the cash assets of every node, and which debts are overdue at each time-stamp. Computing the set $\{v | v$ has some overdue debt under $\sigma\}$ is then straightforward, and the number of bankruptcies is the cardinality of that set.

It remains to check the validity of the schedule. We can efficiently verify that there are:

No withholding banks: iterate once over the debts overdue at each time. If the debt $e = (u, v, i)$ is overdue at time t, verify that c_u^t is insufficient to make a payment toward e (i.e. $c_u^t = 0$ in the FP or PP model, or $c_u^t < D_a(e)$ in the AoN model).
No overpaid debts: iterate over all debts and ensure payments made with reference to each are no more than the debt amount.
No debts paid too early: ensure $p_e^t = 0$ for any $t < D_{t_1}(e)$, for each debt e. □

2.5. Problem definitions

We now define some decision problems with natural real-world applications.

BANKRUPTCY MINIMIZATION

Instance: an IDM instance (G, D, A^0) and an integer k
Yes-instance: an instance for which there exists a valid schedule σ such that at most k banks go bankrupt at some time in the schedule σ.

PERFECT SCHEDULING

Instance: an IDM instance (G, D, A^0)
Yes-instance: an instance for which there exists a valid schedule σ such that no debt is ever overdue in σ; that is, a *perfect schedule*.

BAILOUT MINIMIZATION

Instance: an IDM instance (G, D, A^0) (with n banks) and an integer b
Yes-instance: an instance for which there exists a positive bailout vector $B = (b_1, b_2, \ldots, b_n)$ with $\sum_{i=1}^{n} b_i \leq b$ and valid schedule σ such that σ is a perfect schedule for the instance $(G, D, A^0 + B)$.

The problem PERFECT SCHEDULING is equivalent to the BAILOUT MINIMIZATION problem where $b = 0$ and to the BANKRUPTCY MINIMIZATION problem where $k = 0$.

BANKRUPTCY MAXIMIZATION

Instance: an IDM instance (G, D, A^0) and an integer k
Yes-instance: an instance for which there exists a valid schedule σ such that at least k banks go bankrupt at some time in the schedule σ.

The problem BANKRUPTCY MAXIMIZATION is interesting to consider for quantifying a 'worst-case' schedule where banks' behaviour is unconstrained beyond the terms of their debts.

All of the problems above exist in the AoN, PP and FP variants and 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 (see Lemma 1).

Every valid PP schedule is a valid FP schedule whereas 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) whereas this is prohibited in any PP schedule as that bank would be withholding. If we restrict the instances to only those in which for every debt 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 digraph G from some IDM instance a *multiditree* whenever the footprint of G is a tree. We call a multiditree in which every edge is directed away from the root a *rooted out-tree* (or just *out-tree*). By an *out-path* we mean an out-tree where the footprint is a path and the root is either of the endpoints. We take this opportunity to note that an out-tree is both a directed acyclic graph (DAG) and a multiditree, but that not every multiditree DAG is an out-tree.

A summary of some of our upcoming results is given in Table 1 (with NP-c denoting 'NP-complete' and P denoting 'polynomial-time'). However, note that there are other, more nuanced results in what follows that do not feature in Table 1. Also, even though hardness for out-trees entails hardness for multiditrees and DAGs, we reference a separate result for the latter two settings where that is proven under different (stronger) constraints for the more general graph class. For example, our proof that AoN BANKRUPTCY MINIMIZATION is NP-complete on out-trees uses a construction requiring $T \geq 2$, but our proof of Theorem 1 has $T = 1$.

2.6. Discussion of the model

We describe here some notable differences (and similarities) of the IDM as compared with other studied models.

First and foremost, the IDM is a temporal model; the eponymous "interval debts" are its principal distinguishing feature when contrasted with other financial network models. The *timing* of payments, not their allocation to one payee or another, is the principal question. In fact, in PERFECT SCHEDULING this is the *only* question. As we shall see, under the restriction $D_{t_1} = D_{t_2}$ that problem (and its superproblem BAILOUT MINIMIZATION) becomes straightforwardly solvable in all variants. All of our hardness results arise from the expressivity of that degree of freedom (the scheduling of payments sooner or later). Indeed, in BANKRUPTCY MINIMIZATION and BANKRUPTCY MAXIMIZATION that freedom remains, and the problems remain NP-complete under the same restriction $D_{t_1} = D_{t_2}$.

Table 1
Summary of results.

problem	out-tree	multiditree	DAG	general case
FP Bankruptcy Minimization	?	?	NP-c (Theorem 1)	NP-c (Theorem 1)
PP Bankruptcy Minimization	?	NP-c (Theorem 4)	NP-c (Theorem 1)	NP-c (Theorem 1)
AoN Bankruptcy Minimization	NP-c (Theorem 6)	NP-c (Theorem 6)	NP-c (Theorem 1)	NP-c (Theorem 1)
FP Perfect Scheduling	P (Theorem 9)	P (Theorem 9)	P (Theorem 9)	P (Theorem 9)
PP Perfect Scheduling	P (Theorem 11)	NP-c (Theorem 4)	NP-c (Theorem 3)	NP-c (Theorem 3)
AoN Perfect Scheduling	NP-c (Theorem 6)	NP-c (Theorem 6)	NP-c (Theorem 3)	NP-c (Theorem 3)
FP Bailout Minimization	P (Theorem 9)	P (Theorem 9)	P (Theorem 9)	P (Theorem 9)
PP Bailout Minimization	P (Theorem 11)	NP-c (Theorem 4)	NP-c (Theorem 3)	NP-c (Theorem 3)
AoN Bailout Minimization	NP-c (Theorem 6)	NP-c (Theorem 6)	NP-c (Theorem 3)	NP-c (Theorem 3)
FP Bankruptcy Maximization	?	?	NP-c (Theorem 7)	NP-c (Theorem 7)
PP Bankruptcy Maximization	?	?	NP-c (Theorem 7)	NP-c (Theorem 7)
AoN Bankruptcy Maximization	NP-c (Theorem 8)	NP-c (Theorem 8)	NP-c (Theorem 8)	NP-c (Theorem 8)

Consequently, the results of other works which do not have a temporal component [6,7,9] do not straightforwardly carry over to the IDM.

We take this opportunity to emphasize that interval debts are practically motivated; in particular, some real-world debts may be paid neither early nor late (see, e.g., Fig. 5).

Non-proportional payments on debts are likewise nothing new to financial networks. Recent work has considered frameworks wherein *priorities* are associated with each debt, with higher-priority debts paid off before lower-priority debts [6,9]. In such a setting, the priority of some debt may be either chosen by a regulatory authority or left to the individual agents. In the former case, the hardness of computing a solution which maximizes utility is of particular interest, whereas game-theoretic approaches are more relevant in the latter. Our focus in the present work is solely on questions of computational complexity from a centralized perspective.

We note that Bailout Minimization and its subproblem Perfect Scheduling remain unchanged as decision problems if bankruptcy in the IDM is redefined to require proportional payments, or immediate deletion of the bankrupt node. Both problems fundamentally ask whether a perfect schedule exists; consequently, the manner in which bankruptcy and defaulting are modelled in the IDM are irrelevant. If there is a perfect schedule σ, then under σ all debts are by definition paid on time, in full (and hence proportionally). Conversely, if no such σ exists, then a bankruptcy (however it is modelled) must occur, and we have a no-instance of the respective problems.

Lastly, we would like to comment briefly on the respective practical value of the AoN, PP and FP variants. The FP variant is quite intuitive for theoreticians, and yields our main tractable case. On the other hand, the PP variant realizes the practical constraint that arbitrarily small transfers are impractical, and may be of interest where a fungible but indivisible resource needs to be exchanged. Personal communication [24] suggests that, perhaps unexpectedly, the AoN variant may well be the one of most interest to the finance community. Unlike most other models, the AoN model has the unintuitive property that a bankrupt bank may retain some assets. We note that this modelling of bankruptcy is not required for any of our hardness of tractability proofs in the AoN model.

3. Our results

In this section we investigate the complexity of the problems presented above. We present our hardness results for Bankruptcy Minimization, Perfect Scheduling and Bankruptcy Maximization in Sections 3.1, 3.2 and 3.3, respectively, and then in Section 3.4 show that under certain constraints the problem Bailout Minimization and its subproblem Perfect Scheduling become tractable.

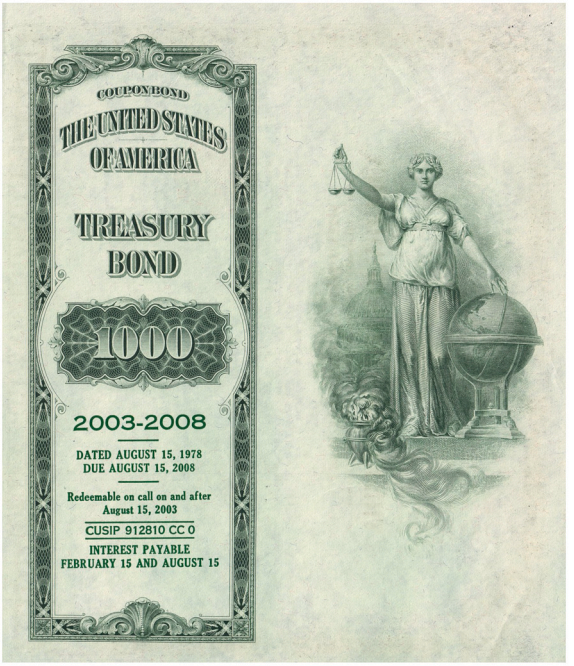


Fig. 5. A real-life interval debt: this 1978 US government bond is payable between 2003 and 2008.

3.1. Hardness results for BANKRUPTCY MINIMIZATION

We begin with our core hardness result.

Theorem 1. *For each of the AoN, PP and FP variants, the problem BANKRUPTCY MINIMIZATION is NP-complete, even when we restrict to IDM instances (G, D, A^0) for which: $T = 1$; G is a directed acyclic graph with a longest path of length 4, has out-degree at most 2 and has in-degree at most 3; the monetary amount of any debt is at most €3; and initial external assets are at most €3 per bank.*

Proof. We build a polynomial-time reduction from the problem 3-SAT-3 to BANKRUPTCY MINIMIZATION so that all target instances satisfy the constraints in the statement of the theorem (the problem 3-SAT-3, defined below, was shown to be NP-complete in [25]).

3-SAT-3

> Instance: a c.n.f. formula ϕ over n Boolean variables $v_1, v_2, ..., v_n$ so that each of the m clauses $c_1, c_2, ..., c_m$ has size at most 3 and where there are exactly 3 occurrences of v_i or $\neg v_i$ in the clauses
> Yes-instance: there exists a satisfying truth assignment for ϕ.

We may (and do, throughout the paper) restrict ourselves to those instances in which every literal appears at least once and at most twice (that is, both v_i and $\neg v_i$ appear in some clause, for $1 \leq i \leq n$) and where no clause contains both a literal and its negation. We define the size of an instance ϕ to be n.

Suppose that we are given a 3-SAT-3 instance ϕ of size n. We construct an IDM instance (G, D, A^0) as follows. For any variable v_i, denote by $count_{v_i}$ (resp. $count_{\neg v_i}$) the number of occurrences of the literal v_i (resp. $\neg v_i$) in ϕ (of course, $count_{v_i} + count_{\neg v_i} = 3$). We build a digraph G with:

- a *source node* s_i, for each variable v_i, so that this node has initial external assets €3 (every other type of node will have initial external assets €0)
- two *literal nodes* x_i and $\neg x_i$, for each variable v_i
- a *clause node* q_j, for each clause c_j
- a *sink node* d.

We then add edges and debts as follows. For every $1 \leq i \leq n$:

- we add the debt (s_i, x_i) with terms $3@1$
- we add the debt $(s_i, \neg x_i)$ with terms $3@1$
- we add the debt (x_i, d) with terms $count_{\neg v_i}@1$ (note that the monetary amount to be paid is either €1 or €2)

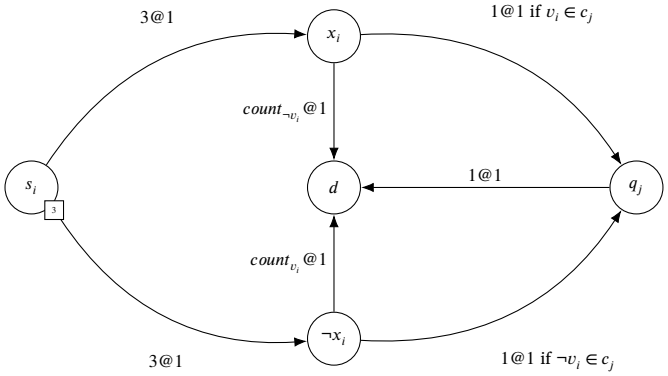


Fig. 6. Construction sketch for an IDM instance from a given formula ϕ. Note that each of x_i and $\neg x_i$ owes €3 in total.

- we add the debt $(\neg x_i, d)$ with terms $count_{v_i}$@1 (note that the monetary amount to be paid is either €1 or €2)
- for every $1 \leq j \leq m$:
  - we add the debt (q_j, d) with terms 1@1
  - if the literal $v_i \in c_j$ then we add the debt (x_i, q_j) with terms 1@1
  - if the literal $\neg v_i \in c_j$ then we add the debt $(\neg x_i, q_j)$ with terms 1@1.

Fig. 6 shows a sketch of this construction (where nodes without any depicted initial external assets start with €0). The IDM instance (G, D, A^0) can clearly be built from ϕ in polynomial-time.

We claim that the instance $((G, D, A^0), 2n)$ of BANKRUPTCY MINIMIZATION as constructed above is a yes-instance of BANKRUPTCY MINIMIZATION (no matter which of the AoN, PP and FP variants we work with) iff ϕ is a yes-instance of 3-SAT-3.

Before we proceed, we have the following remark. Recall that for each $1 \leq i \leq n$, $count_{v_i} + count_{\neg v_i} = 3$; consequently, $count_{v_i} = 2$ iff $count_{\neg v_i} = 1$, and vice versa. For each $1 \leq i \leq n$, node x_i (resp. $\neg x_i$) has a total monetary debt to the clause nodes of $count_{v_i}$ (resp. $count_{\neg v_i}$) and a total monetary debt to the sink node d of $count_{\neg v_i}$ (resp. $count_{v_i}$); so, each literal node has a total monetary debt of €3.

Claim 1. *If ϕ is a yes-instance of 3-SAT-3 then $((G, D, A^0), 2n)$ is a yes-instance of BANKRUPTCY MINIMIZATION.*

Proof. Suppose that ϕ is satisfiable via some truth assignment X. Consider the schedule σ for (G, D, A^0) in which:

- every source node s_i pays €3 (at time 1, as are all payments) to the literal node x_i (resp. $\neg x_i$) if $X(v_i) = True$ (resp. $X(v_i) = False$)
- every literal node x_i (resp. $\neg x_i$) for which $X(v_i) = True$ (resp. $X(v_i) = False$) pays all its debts in full
  - as remarked above, this literal node has total monetary debt €3 but, from above, it receives €3 from s_i
- every clause node pays its €1 debt to the sink node d
  - this is necessarily possible because X is a satisfying truth assignment, meaning every clause node receives at least €1 from some literal node corresponding to a literal in that clause set to $True$ by X.

Note that σ is valid in all three IDM variants. The total number of bankruptcies in σ is $2n$: exactly n bankrupt source nodes and exactly n bankrupt literal nodes. Hence, if ϕ is satisfiable then the schedule σ for (G, D, A^0) results in at most (in fact, exactly) $2n$ bankruptcies. □

Claim 2. *If $(G, D, A^0), 2n)$ is a yes-instance of BANKRUPTCY MINIMIZATION then ϕ is a yes-instance of 3-SAT-3.*

Proof. Suppose that we have a schedule σ for (G, D, A^0) with at most $2n$ bankruptcies. Consider the set of all literals $L = \{v_1, \neg v_1, v_2, \neg v_2, ..., v_n, \neg v_n\}$ w.r.t. ϕ. Define the set of *bankrupt literals* $B \subseteq L$ to consist of every literal whose corresponding (literal) node is bankrupt within σ. Define $X(w) = True$ iff $w \in L \setminus B$. We claim that X is a (complete) truth assignment. Suppose it is not and that $X(v_i) = X(\neg v_i) = True$; so, both x_i and $\neg x_i$ are bankrupt within σ. However, in any valid schedule (no matter what the IDM variant) every source node s_i will necessarily go bankrupt and at least one of the literal nodes x_i and $\neg x_i$ will go bankrupt. Thus, as we have at most $2n$ bankruptcies, our supposition is incorrect. Alternatively, suppose that $X(v_i) = X(\neg v_i) = False$; so, neither x_i nor $\neg x_i$ is bankrupt within σ. But, as stated, this cannot be the case. So, X is a truth assignment; moreover, σ has exactly $2n$ bankruptcies with exactly one of any pair of 'oppositely-oriented' literal nodes bankrupt.

Suppose, for contradiction, that X is not a satisfying assignment. So, there exists at least one clause, c_j say, such that every literal in the clause is made $False$ by X. By definition of X, we have that every literal node corresponding to one of these literals is a bankrupt node. Any such literal node must receive €0 (as the 'oppositely-oriented' literal node is not bankrupt and receives €3);

consequently, the clause node q_j receives €0 and is bankrupt. This yields a contradiction as we have exactly $2n$ bankrupt nodes (as detailed above). □

Consequently, ϕ is satisfiable iff the IDM instance (G, D, A^0) admits a schedule with at most $2n$ bankruptcies (again, this holds for each IDM variant). This concludes our proof. □

Our next result is perhaps rather surprising in that we restrict to IDM instances (G, D, A^0) where G is a fixed digraph.

Theorem 2. *For each of the AoN, PP and FP variants, the problem* BANKRUPTCY MINIMIZATION *is weakly NP-complete, even when we restrict to instances* $((G, D, A^0), k)$ *where* G *is a fixed, specific digraph with 32 nodes and* $k = 16$.

Proof. We build a polynomial-time reduction from EQUAL CARDINALITY PARTITION to BANKRUPTCY MINIMIZATION (EQUAL CARDINALITY PARTITION, defined below, was proven weakly NP-complete in [26]).

EQUAL CARDINALITY PARTITION

Instance: a multi-set of positive integers $S = \{a_1, a_2, ..., a_n\}$ where n is even and with sum $sum(S) = 2k$
Yes-instance: there exist a partition of S into two equal-sized sets S_1 and S_2 such that $sum(S_1) = sum(S_2) = k$.

The size of such an instance is nb where b is the least number of bits so that any integer a_i can be represented in binary using b bits.

Given an instance $S = \{a_1, a_2, ..., a_n\}$ of EQUAL CARDINALITY PARTITION (where $n \geq 1$), we construct the IDM instance (G, D, A^0) that is illustrated in Fig. 7. Every appearance of the shaded node p in Fig. 7 corresponds to the same single node, that we refer to as the sink, and thus this instance has 32 nodes in total. We use the symbol '∞' to denote a suitably high monetary amount (though $2k + n + 1$ suffices) and $T = 10n + 7$. Our IDM instance can trivially be constructed from S in time polynomial in the size of the instance S. We now show that (G, D, A^0) admits a valid schedule with at most 16 bankruptcies iff S is a yes-instance of EQUAL CARDINALITY PARTITION (no matter whether we are in the AoN, PP or FP variant). Until further stated, we will work solely within the PP variant and return to the AoN and FP variants later.

Claim 3. *If the IDM instance* $((G, D, A^0), 16)$ *is a yes-instance of* BANKRUPTCY MINIMIZATION *then* S *is a yes-instance of* EQUAL CARDINALITY PARTITION.

Proof. Suppose that (G, D, A^0_e) admits a valid schedule σ with at most 16 bankruptcies. Note that the 14 nodes m_1, m_3, m^A_4, m^A_6, m^A_8, m^A_{11}, m^A_{13}, m^A_{15}, m^B_4, m^B_6, m^B_8, m^B_{11}, m^B_{13} and m^B_{15} are necessarily bankrupt in *every* valid schedule because they are debtors of debts of monetary amount ∞ at time 1 and no payments are made before time 10 (these nodes are dashed and highlighted in red in Fig. 7 as are the debts at time 1 of monetary amount ∞). Note also that these nodes can never have any cash assets at any time and nor can the nodes m_2, m^A_5, m^A_9, m^A_{10}, m^A_{14}, m^B_5, m^B_9, m^B_{10} and m^B_{14} (as they would otherwise be withholding). Consequently, we can only have at most another 2 nodes going bankrupt within σ. We begin by showing that one of $\{m^A_7, m^A_{12}\}$ must be bankrupt and one of $\{m^B_7, m^B_{12}\}$ must be bankrupt (which will account for all bankrupt nodes).

Suppose that none of the nodes m^A_7, m^A_9, m^A_{10} and m^A_{12} are bankrupt in σ. By considering m^A_{12} at the times $t \in \{11, 21, \ldots, 10n + 1\}$, on at least $\frac{n}{2}$ of these occasions m^A_{12} must have received at least €1 via payments from m^A_{11} (so as to service all its debts to m^A_7). Consider the first occasion $t' \in \{11, 21, \ldots, 10n + 1\}$ that m^A_{12} receives a non-zero payment from m^A_{11} (note that all payments from m^A_{11} to m^A_{12} are made at some time from $\{11, 21, \ldots, 10n + 1\}$). There must have been a payment of €1 from m^A_{10} to m^A_{11} at time t' as well as payments of €1 from m^A_9 to m^A_{10} and from m^A_8 to m^A_9 at time t'. So, m^A_8 must receive at least €1 from m^A_7 and m^A_{11} at time t'. As m^A_{11} makes a non-zero payment to m^A_{12} at time t', any payment from m^A_{11} to m^A_7 at time t' must be for some amount strictly less than €1. Consequently, m^A_8 must receive a non-zero payment from m^A_7 at time t'. The only way that this can happen is if there is an overdue debt from m^A_7 to m^A_8; that is, m^A_7 is bankrupt, which yields a contradiction. Hence, at least one of m^A_7, m^A_9, m^A_{10} and m^A_{12} is bankrupt. An analogous argument shows that at least one of m^B_7, m^B_9, m^B_{11} and m^B_{12} is bankrupt. Hence, exactly one of m^A_7, m^A_9, m^A_{10} and m^A_{12} is bankrupt and exactly one of m^B_7, m^B_9, m^B_{11} and m^B_{12} is bankrupt.

Suppose that m^A_9 is bankrupt. So, m^A_{10} is necessarily bankrupt. Conversely, if m^A_{10} is bankrupt then m^A_9 must be bankrupt also. Consequently, neither m^A_9 nor m^A_{10} is bankrupt and we must have that either m^A_7 or m^A_{12} is bankrupt and analogously either m^B_7 or m^B_{12} is bankrupt. In particular, s, m_2, m^A_5, m^A_{13}, m^A_{15}, m^B_5, m^B_{13} and m^B_{15} are not bankrupt.

Let us turn to analysing the flow of resource via the schedule σ. As σ is valid, we must have that both debts from m^A_{16} and m^B_{16} are paid on time with €$2k$ in total reaching d. The question is: does this resource consist of the €$2k$ emanating from s or does it consist of resource emanating from s but supplemented with resource emanating from m^A_{12} or m^B_{12}? Let us look at the possible debt payments at time 10 (which is the earliest time that payments can be made). In particular, let us look at payments made by m^A_6 to m^A_7 at time 10. Note that all payments made from m^A_6 to m^A_7 are at a time from $\{10, 20, \ldots, 10n\}$.

Case (a): An amount of €$x > 0$ is paid from m^A_6 to m^A_7 at time 10.

There are two essential sub-cases at time 10:

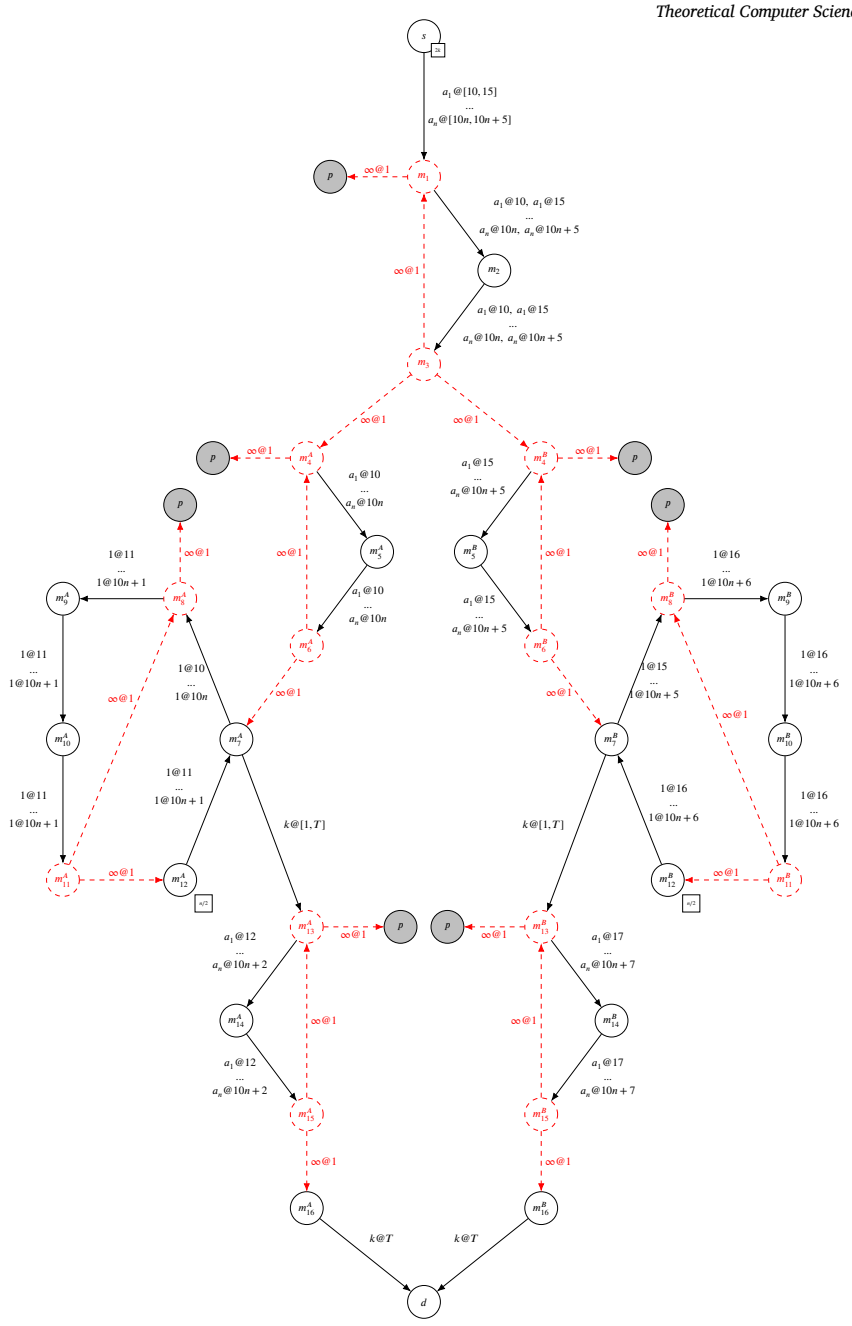


Fig. 7. Construction of an IDM instance (with $k = 16$) corresponding to the EQUAL CARDINALITY PARTITION instance $S = \{a_1, \ldots, a_n\}$. Dashed red edges are "practically infinite" bankrupting debts.

(*i*) m_7^A services its debt to m_8^A, which pays €1 to the sink, with perhaps $€y \geq 0$ paid from m_7^A to m_{13}^A and from there to the sink, so that $€x - y - 1 \geq 0$ resides at m_7^A in cash assets at time 10

(*ii*) m_7^A does not service its debt to m_8^A and pays $€x$ to m_{13}^A with this payment immediately going to the sink, so that €0 resides at m_7^A in cash assets at time 10; hence, m_7^A is bankrupt (note that no cash assets can reside at m_7^A at time 10 as otherwise m_7^A would be withholding).

Now consider what happens at time 11. Suppose that we are in Case (*a.i*). There must be a payment-cycle involving m_8^A, m_9^A, m_{10}^A and m_{11}^A and as $\frac{n}{2} \geq 1$, m_{12}^A must service its debt to m_7^A, with perhaps m_7^A paying $€z \geq 0$ to m_{13}^A which is immediately paid to the sink. The €1 from m_{12}^A does not supplement the resource emanating from s but just 'replaces' €1 which was 'lost' to the sink at time 10. Note that the cash assets of m_{12}^A at time 11 are $\frac{n}{2} - 1$.

Suppose that we are in Case (*a.ii*). Note that as m_7^A is bankrupt, m_{12}^A can never become bankrupt and so must service its debts when required. There are four possibilities as regards what happens at time 11 (bearing in mind the overdue debt from m_7^A to m_8^A):

(1) m_{12}^A pays €1 to m_7^A which pays €1 to m_8^A which immediately goes to the sink, with a payment-cycle involving m_8^A, m_9^A, m_{10}^A and m_{11}^A
(2) m_{12}^A pays €1 to m_7^A which pays €1 to m_8^A which pays €1 to m_9^A which pays €1 to m_{10}^A which pays €1 to m_{11}^A which pays €1 to m_8^A which immediately goes to the sink
(3) m_{12}^A pays €1 to m_7^A which pays €1 to m_{13}^A which immediately goes to the sink, with a payment-cycle involving m_8^A, m_9^A, m_{10}^A and m_{11}^A
(4) m_{12}^A pays €1 to m_7^A which pays €1 to m_8^A which pays €1 to m_9^A which pays €1 to m_{10}^A which pays €1 to m_{11}^A which pays €1 to m_{12}^A; that is, we have a payment cycle involving m_7^A, m_8^A, m_9^A, m_{10}^A, m_{11}^A and m_{12}^A.

In (1-3) above, the €1 from m_{12}^A does not supplement the resource emanating from s but is lost to the sink (as are the €$x > 0$ at time 10). Note that in (3), the debt from m_7^A to m_8^A at time 10 is overdue and cannot be paid at time 11 as m_7^A has no cash assets at time 10; so it remains overdue. In (4), again no supplement is made, although €$\frac{n}{2}$ still resides at m_{12}^A in cash assets at time 11, and €$x > 0$ emanating from s has been lost to the sink.

In both Case (*a.i*) and Case (*a.ii*), at time 12, the only possible non-payment-cycle payments involve s, m_1, m_7^A, m_{13}^A, m_{14}^A and m_{15}^A but any such payments do not affect whether the resources emanating from m_{12}^A or m_{12}^B supplement the resource emanating from s. In Case (*a.ii*.3), the debt from m_7^A to m_8^A at time 10 is overdue and so must be paid from the cash assets of m_7^A at time 12, if it has any and unless all of these assets are paid to m_{13}^A. All such payments by m_7^A will immediately go to the sink.

Case (*b*): No payment is made from m_6^A to m_7^A at time 10.

Consequently, m_7^A cannot pay its debt to m_8^A and becomes bankrupt. At time 11, m_{12}^A must necessarily service its debt to m_7^A and there are four possibilities with these possibilities being exactly the possibilities (1-4) in Case (*a.ii*). As before, at time 12, the only possible non-payment-cycle payments made involve s, m_1, m_7^A, m_{13}^A, m_{14}^A and m_{15}^A but any such payments do not affect whether the resources emanating from m_{12}^A or m_{12}^B supplement the resources emanating from s. Note that in (3), the debt from m_7^A to m_8^A at time 10 is still overdue at time 11 and cannot be paid at time 12 as m_7^A has no cash assets at time 12; so it remains overdue. In (4), again no supplement is made, although €$\frac{n}{2}$ still resides at m_{12}^A in cash assets at time 12.

So, the resources emanating from m_{12}^A cannot supplement the resources emanating from s at any time $t < 15$. An identical argument can be applied to time 15 so as to yield a similar conclusion as regards the resources emanating from m_{12}^B at any time $t < 20$.

Consider the situation at time 20. With regard to the sub-network involving m_6^A, m_7^A, m_8^A, m_9^A, m_{10}^A, m_{11}^A, m_{12}^A and m_{13}^A (that is, the sub-network of study above), the situation is similar to that at time 10 except that there may be additional restrictions on what can happen given: a possible existing overdue debt from m_7^A to m_8^A (the debt due at time 10); possible non-zero cash assets at m_7^A; and possibly reduced cash assets at m_{12}^A of $\frac{n}{2} - 1$. Note that if m_7^A has cash assets prior to time 20 then this can be thought of as m_7^A having acquired these assets from m_6^A at time 20; that is, we are in Case (*a*) above. Given the fact that the situation at time 20 is a restricted version of the situation at time 10 where the resources emanating from m_{12}^A could not supplement those emanating from s, the same is true again. By analysing each time $t = 25, 30, 35, \ldots$, we can see that no resources emanating from either m_{12}^A or m_{12}^B can supplement those emanating from s. Hence, in order to secure total cash assets of €$2k$ at d after time T, we need that all of the €$2k$ resource emanating from s reaches d; that is, none of it is lost to the sink en route (although some of it might have been 'replaced' as per Case (*a.i*)).

We can now repeat the above analysis except that now we know that we cannot lose resource emanating from s unless it is replaced as in Case (*a.i*). This simplifies things considerably. If m_7^A has €$x > 0$ at time $t \in \{10, 20, \ldots, 10n\}$ (either as cash assets or from a payment by m_6^A at time t) then it must be the case that m_7^A services the debt to m_8^A at time t, €1 is lost to the sink and m_7^A retains $x - 1$ in cash assets. At time $t + 1$, m_{12}^A must service its debt to m_7^A so as to replace the lost €1, with €x residing at m_7^A in cash assets at time $t + 1$. Consequently, there can only be at most $\frac{n}{2}$ times in $\{10, 20, \ldots, 10n\}$ when m_7^A receives a payment from m_6^A (recall that m_6^A only makes payments to m_7^A at times from $\{10, 20, \ldots, 10n\}$). When m_7^A either has no cash assets or receives no payment from m_6^A at time $t \in \{10, 20, \ldots, 10n\}$, we either lose €1 of cash assets from m_{12}^A to the sink (and so we also lose some capacity to 'replace' resource emanating from s that is lost to the sink) or we are in case (4) above and have a payment cycle involving m_7^A, m_8^A, m_9^A, m_{10}^A, m_{11}^A and m_{12}^A. Analogous comments can be made as regards the corresponding nodes superscripted B and times in $\{15, 25, \ldots, 10n + 5\}$.

Bearing in mind that none of the resource emanating from s goes to the sink before it reaches either m_6^A or m_6^B, at least n distinct payments are made from s and these payments result in at least n payments in total from m_6^A to m_7^A or from m_6^B to m_7^B. Thus, from above, m_6^A must make exactly $\frac{n}{2}$ payments to m_7^A and m_6^B must make exactly $\frac{n}{2}$ payments to m_7^B. This means that any payment from m_6^A to m_7^A or from m_6^B to m_7^B must be for an amount from $\{a_1, a_2, \ldots, a_n\}$ and we have a partition of $\{a_1, a_2, \ldots, a_n\}$ into equal-sized sets both of whose sum is k; that is, our instance S of Equal Cardinality Partition is a yes-instance and the claim follows. □

Claim 4. *If S is a yes-instance of Equal Cardinality Partition then $((G, D, A^0), 16)$ is a yes-instance of Bankruptcy Minimization.*

Proof. Suppose that our instance $S = \{a_1, a_2, \ldots, a_n\}$ of EQUAL CARDINALITY PARTITION is such that $n = 2m$ and $\sum_{i=1}^{m} a_{\alpha_i} = \sum_{i=1}^{m} a_{\beta_i}$, where $\{\alpha_i, \beta_i : 1 \leq i \leq m\} = \{1, 2, \ldots, n\}$. We need to build a valid schedule σ for (G, D, A^0) with at most 16 bankruptcies. Let $1 \leq i \leq n$ and suppose that $i = \alpha_j$, where $1 \leq j \leq m$. The node s pays its ith debt to m_1 (that is, the debt $a_i@[10i, 10i+5]$) at time $10i$ and this payment is percolated all the way down to m_7^A at time $10i$. The debt $1@10i$ from m_7^A to m_8^A is paid at time $10i$ with this €1 being replaced from m_{12}^A at time $10i+1$ (see Case (*a.i*) from the proof of Claim 3 above). We also have a payment cycle involving m_8^A, m_9^A, m_{10}^A and m_{11}^A at time $10i+1$. The cash assets of a_i at m_7^A are percolated down to m_{16}^A at time $10i+2$. At time $10i+5$, we have suitable payment cycles involving: m_1, m_2 and m_3; m_4, m_5 and m_6; and m_{13}^A, m_{14}^A and m_{15}^A. We also have a payment cycle involving m_7^A, m_8^A, m_9^A, m_{10}^A, m_{11}^A and m_{12}^A (see (4) from the proof of Claim 3 above). An analogous course of action is taken if $i = \beta_j$, for some $1 \leq j \leq m$. The resulting schedule is valid and the 16 nodes m_1, m_3, m_4^A, m_6^A, m_7^A, m_8^A, m_{11}^A, m_{13}^A, m_{15}^A, m_4^B, m_6^B, m_7^B, m_8^B, m_{11}^B, m_{13}^B and m_{15}^B are bankrupt. The claim follows. □

So, our main result holds for the PP variant of BANKRUPTCY MINIMIZATION. Note that everything above holds for the AoN variant too, though Theorem 6 is a strictly stronger result for that setting.

Let us consider now the FP variant. As it happens, an argument similar to that above works within the FP variant although there are more complicated nuances. Rather than repeat the whole nuanced argument in detail, and given the above complete proof for the PP variant, we only sketch the proof for the FP variant. Henceforth, we assume that we are working within the FP variant.

Consider the proof of the corresponding version of Claim 3. The reasoning that establishes that we must have that either m_7^A or m_{12}^A is bankrupt and that either m_7^B or m_{12}^B is bankrupt holds for the FP variant. Consider payments made from m_6^A to m_7^A at time 10.

Case (*a*): An amount of $€x > 0$ is paid from m_6^A to m_7^A at time 10.

There are two essential sub-cases at time 10:

(*i*) m_7^A services its debt to m_8^A, which pays €1 to the sink, with perhaps $€y \geq 0$ paid from m_7^A to m_{13}^A and from there to the sink, so that $€x - y - 1 \geq 0$ resides at m_7^A in cash assets at time 10

(*ii*) m_7^A does not fully service its debt to m_8^A but pays $€w$, where $0 \leq w < 1$, to m_8^A, which immediately goes to the sink, and $€x - w$ to m_{13}^A, which immediately goes to the sink, so that €0 resides at m_7^A in cash assets at time 10; hence, m_7^A is bankrupt.

Consider what happens at time 11. In Case (*a.i*), there must be a payment cycle involving m_8^A, m_9^A, m_{10}^A and m_{11}^A and m_{12}^A services its debt to m_7^A. The €1 from m_{12}^A does not supplement the resource emanating from s but just 'replaces' €1 which was 'lost' to the sink at time 10.

Suppose that we are in Case (*a.ii*). There are two scenarios:

(1) m_{12}^A pays €1 to m_7^A which pays $€1 - w$ to m_8^A of which $€w'$ goes immediately to the sink and $€1 - w - w'$ is paid to m_9^A; m_7^A has cash assets of at most $€w$ as it may be the case that m_7^A also makes a payment to m_{13}^A which goes straight to the sink, or

(2) m_{12}^A pays €1 to m_7^A which pays $€y$ to m_8^A, where $0 \leq y < 1 - w$, of which $€y'$ goes immediately to the sink and $€y - y'$ is paid to m_9^A; also, $€1 - y$ is paid to m_{13}^A which goes straight to the sink.

In (1), it must be the case that m_8^A receives at least $w + w'$ from m_{11}^A (so as to fully service its debt to m_9^A); hence, m_{11}^A pays at most $€1 - w - w'$ to m_{12}^A. In any case, $€x$ have been lost to the sink with m_7^A gaining $€w$ from m_{12}^A (with $x \geq w$). In (2), it must be the case that m_8^A receives at least $€1 - (y - y')$ from m_{11}^A (so as to fully service its debt to m_9^A); hence, m_{11}^A pays at most $€y - y'$ to m_{12}^A. In any case, $€x$ have been lost to the sink with m_7^A gaining nothing from m_{12}^A.

Case (*b*): No payment is made from m_6^A to m_7^A at time 10.

At time 11, it must be the case that m_{12}^A services its debt to m_7^A and then we are essentially in Case (*a.ii*) above.

The outcome is that the resources emanating from m_{12}^A cannot supplement the resources emanating form s at any time $t < 15$. An identical argument can be applied to time 15 so as to yield a similar conclusion as regards the resources emanating from m_{12}^B at any time $t < 20$. The rest of the proof of Claim 3 holds for the FP variant and we have that Claim 3 holds for the FP variant.

The schedule σ described in the proof of Claim 4 is a valid schedule in the FP variant and so Claim 4 also holds for the FP variant. This complete our proof of the main theorem. □

The proof of Theorem 2 clearly demonstrates the intricacies of reasoning in our financial networks. By Theorem 2, 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 footprint. Note that Theorem 2 concerns weak completeness results (in particular, the integers in an instance of EQUAL CARDINALITY MATCHING appear explicitly as monetary debts in the corresponding instance of BANKRUPTCY MINIMIZATION).

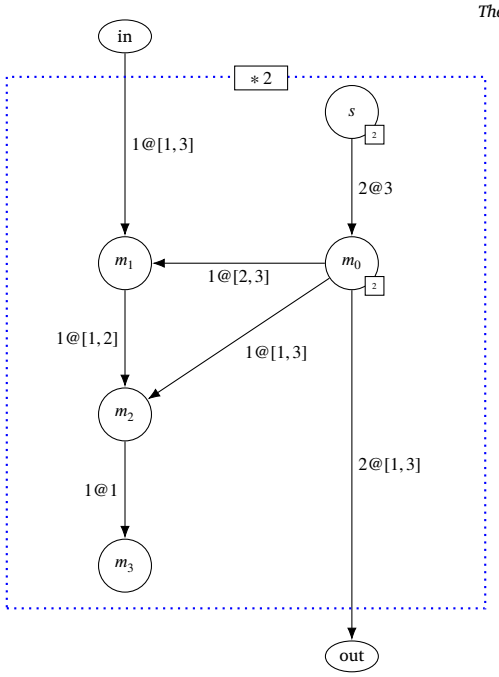


Fig. 8. The multiplier gadget. Intuitively, the gadget "amplifies" payments into it at time 1 by a factor 2.

3.2. Hardness results for PERFECT SCHEDULING

We now turn to PERFECT SCHEDULING. Since this is a subproblem of BANKRUPTCY MINIMIZATION and BAILOUT MINIMIZATION, hardness results in this section also apply to both of those problems.

Theorem 3. *The problem PERFECT SCHEDULING is NP-complete for the AoN and PP variants even when we restrict to IDM instances (G, D, A^0) for which: $T \leq 3$; G is a directed acyclic graph with out-degree at most 3 and in-degree at most 3; the monetary amount of any debt is at most €2; and any initial cash assets of a node is at most €3 per bank.*

Proof. Let us work within the PP variant until further notice. We introduce the *multiplier gadget* shown in Fig. 8 and use this gadget in our main reduction (the gadget sits within the blue dotted line). We claim the following.

Claim 5. *Assume that the node in of the multiplier gadget has initial cash assets of €1.*

(*a*) *In any perfect schedule for the multiplier gadget, if no payment is made by in to m_1 at either time 1 or time 2 then no payment is made by m_0 to out at time 1.*
(*b*) *There is a perfect schedule σ_0 for the multiplier gadget so that a payment is made by in at time 3.*
(*c*) *There is a perfect schedule σ_1 for the multiplier gadget so that a payment is made by in to m_1 at time 1 and a payment of €2 is made from m_0 to out at time 1.*

Proof. Suppose that no payment is made by *in* to m_1 at times 1 and 2; so m_1 receives no payment at time 1 and makes no payment at time 1. As there is a payment of €1 from m_2 to m_3 at time 1, there must be a payment of €1 from m_0 to m_2 at time 1. Suppose that a payment of €1 is made from m_0 to *out* at time 1. If so then m_0 can make no payment to m_1 at time 2 and m_1 is bankrupt which yields a contradiction. Hence, there is no payment from m_0 to *out* at time 1. The statement (*a*) follows.

Consider the schedule whereby:

- at time 1: m_0 pays €1 to m_2; m_2 pays €1 to m_3
- at time 2: m_0 pays €1 to m_1; m_1 pays €1 to m_2
- at time 3: s pays €2 to m_0; m_0 pays €2 to *out*; *in* pays €1 to m_1.

This yields a perfect schedule and statement (*b*) follows.

Suppose that there is a payment of €1 made from *in* to m_1 at time 1; so, we can also make payments of €1 from m_1 to m_2 and from m_2 to m_3 at time 1. Additionally, we can make a payment of €2 from m_0 to *out* at time 1. At time 2, no payments are made. At time 3, we can make payments of: €2 from s to m_0; €1 from m_0 to m_2; and €1 from m_0 to m_1. This yields a perfect schedule and statement (*c*) follows. □

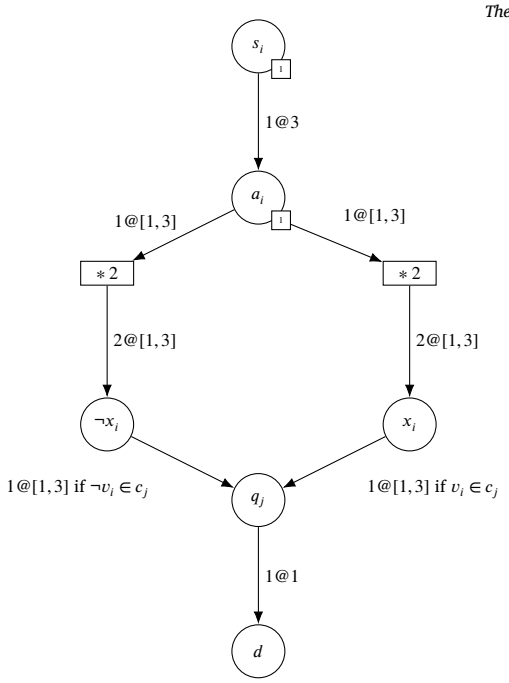


Fig. 9. Illustration of the reduction from 3-SAT 3 to PERFECT SCHEDULING restricted to Directed Acyclic Graphs (DAGs).

Our reduction is from 3-SAT-3 (again) to PERFECT SCHEDULING. As before, we assume that all 3-SAT-3 instances are such that every literal appears at least once and at most twice and that no clause contains both a literal and its negation. Given a 3-SAT-3 instance ϕ involving n Boolean variables and m clauses, we construct an IDM instance (G, D, A^0) as portrayed in Fig. 9 (we omit the formal description of (G, D, A^0) as it can be immediately derived from Fig. 9; moreover, we proceed similarly with other IDM instances that we construct later on). The types of nodes (source, literal, clause and sink) are as in the proof of Theorem 1, although we have additional so-called a-type nodes, and we abbreviate our multiplier gadget as a square box with $*2$ inside (note that we have $2n$ distinct copies of our multiplier gadget where the node *in* is taken as a_i and the node *out* as the literal node x_i or the literal node $\neg x_i$, for $1 \leq i \leq n$; of course, we have m clause nodes, n source nodes, n a-type nodes and one sink node).

Claim 6. *If (G, D, A^0) is a yes-instance of PERFECT SCHEDULING then ϕ is a yes-instance of 3-SAT-3.*

Proof. Define the truth assignment X via: if, within σ, x_i receives a payment of at least €1 at time 1 then $X(v_i) = True$; otherwise $X(v_i) = False$.

Fix $1 \leq j \leq m$. We must have that q_j pays €1 to d at time 1 and so q_j must receive €1 from some node x_i at time 1 or €1 from some node $\neg x_i$ at time 1.

Suppose that q_j receives €1 from x_{τ_j} at time 1; in particular, $v_{\tau_j} \in c_j$. Hence, x_{τ_j} receives at least €1 from its corresponding multiplier gadget at time 1 and so, by definition, $X(v_{\tau_j}) = True$ with the clause c_j satisfied by X.

Alternatively, suppose that q_j receives €1 from $\neg x_{\tau_j}$ at time 1; in particular, $\neg v_{\tau_j} \in c_j$. Hence, $\neg x_{\tau_j}$ receives at least €1 from its corresponding multiplier gadget at time 1. By Claim 5.*a*, there must be a payment of €1 made from a_{τ_j} to this multiplier gadget at either time 1 or time 2. Consequently, no payment is made by a_{τ_j} to the complementary multiplier gadget (that is, the one with a debt to x_{τ_j}) at either time 1 or time 2. By Claim 5.*a*, no payment is received by x_i at time 1 and so, by definition, $X(v_{\tau_j}) = False$ with the clause c_j satisfied by X. The claim follows. □

Claim 7. *If ϕ is a yes-instance of 3-SAT-3 then (G, D, A^0) is a yes-instance of PERFECT SCHEDULING.*

Proof. Let X be a satisfying truth assignment for ϕ. For each clause c_j, let v_{τ_j} be a Boolean variable whose occurrence in c_j, either via the literal v_{τ_j} or the literal $\neg v_{\tau_j}$, leads to c_j being satisfiable. So, we get a list $L = v_{\tau_1}, v_{\tau_2}, \ldots, v_{\tau_m}$ of 'satisfying' Boolean variables, possibly with repetitions although no variable appears in the list more than twice and if a Boolean variable v does appear twice then the corresponding literals in the two corresponding clauses are both positive or both negative (of course, this stems from the format of ϕ as an instance of 3-SAT-3): if the occurrences are positive then we say that v has *positive polarity*, with *negative polarity* defined analogously. Note that any debt from a literal node to a clause node in our IDM instance exists solely because of the occurrence of the corresponding literal in the corresponding clause; in particular, we can never have debts from both literal nodes corresponding to some variable to the same clause node.

Consider the following schedule σ. At time 1, for each $1 \le j \le m$:

- q_j pays €1 to d
- if v_{τ_j} appears in L with positive (resp. negative) polarity then x_{τ_j} (resp. $\neg x_{\tau_j}$) pays €1 to q_j
- if v_{τ_j} appears in L with positive (resp. negative) polarity then a_{τ_j} pays €1 to the multiplier gadget which has a debt to x_{τ_j} (resp. $\neg x_{\tau_j}$).

In addition, for each $1 \le j \le m$, if v_{τ_j} appears in L with positive (resp. negative) polarity then:

- within the multiplier gadget that has a debt to x_{τ_j} (resp. $\neg x_{\tau_j}$), we include the schedule σ_1 from Claim 5.*c*
- within the multiplier gadget that has a debt to $\neg x_{\tau_j}$ (resp. x_{τ_j}), we include the schedule σ_0 from Claim 5.*b*.

At time 3, for each $1 \le j \le m$:

- s_{τ_j} pays €1 to a_{τ_j}
- if v_{τ_j} appears in L with positive (resp. negative) polarity then a_{τ_j} pays €1 to the multiplier gadget which has a debt to $\neg x_{\tau_j}$ (resp. x_{τ_j}).

Finally, having done the above, add any Boolean variable v not appearing in L to L and proceed as above with these Boolean variables and assuming that they have positive polarity (note that the restriction of X to these Boolean variables has no effect on whether X satisfies ϕ). What results is a perfect schedule. □

Given that our IDM instance (G, D, A^0) of PERFECT SCHEDULING can be built from the instance ϕ of 3-SAT-3 in polynomial time, we obtain our result for the PP variant.

Consider now the AoN variant. As it happens, all of the above schedules are perfect schedules within the AoN variant and all associated reasoning still holds. Hence, we have our result for the AoN variant too. □

Theorem 4. *The problem PERFECT SCHEDULING is NP-complete for the PP and AoN variants even when we restrict to IDM instances (G, D, A^0) for which: G is a multiditree with diameter 6; all debts have monetary amount €1; and there is a maximum of 6 debts between any pair of nodes.*

Proof. We show that, given an instance ϕ of 3-SAT-3, involving n Boolean variables and m clauses, we can construct in polynomial-time an IDM instance (G, D, A^0) where G satisfies the criteria stated in the theorem so that (G, D, A^0) admits a perfect schedule iff ϕ has a satisfying truth assignment. As usual, we restrict ourselves to those instances of 3-SAT-3 in which every literal appears at least once and at most twice and where no clause contains both a literal and its negation. In particular, we can label any appearance of any literal in ϕ as the first appearance or the second appearance.

Our reduction is portrayed in Fig. 10. There is a distinct *variable gadget* for each Boolean variable v_i, with $1 \le i \le n$, and a distinct *clause gadget*, for each clause c_j, with $1 \le j \le m$. There is one node r. The debts in any variable gadget or involving the node r are self-evident whereas the debts in a clause gadget are more involved.

- If the literal v_i is in the clause c_j and this appearance is the first (resp. second) appearance of v_i in ϕ then there are:
  - debts $1@10(i-1)+1$ (resp. $1@10(i-1)+3$) from b_j to a_j and from e_j to d_j
  - debts $1@10(i-1)+2$ (resp. $1@10(i-1)+4$) from a_j to b_j and from d_j to e_j.
- If the literal $\neg v_i$ is in the clause c_j and this appearance is the first (resp. second) appearance of $\neg v_i$ in ϕ then there are:
  - debts $1@10(i-1)+6$ (resp. $1@10(i-1)+8$) from b_j to a_j and from e_j to d_j
  - debts $1@10(i-1)+7$ (resp. $1@10(i-1)+9$) from a_j to b_j and from d_j to e_j.
- If the clause c_j has 3 literals (resp. 2 literals) then there are:
  - two separate[2] debts $1@[1,T]$ (resp. a single debt $1@[1,T]$) from a_j to e_j and from e_j to a_j.

The legend in Fig. 10 shows the time intervals corresponding to each literal, which we call the *active windows* of the literals, and the occurrence of x_1 (resp. $\neg x_2$, x_6) in c_j in Fig. 10 is the first (resp. second, first) occurrence.

Claim 8. *If (G, D, A^0) is a yes-instance of PERFECT SCHEDULING then ϕ is a yes-instance of 3-SAT-3.*

Proof. Suppose that there is a perfect schedule σ for (G, D, A^0). Consider a clause gadget, corresponding to the clause c_j. There are 4 debts due within each active window corresponding to a literal in the clause. Suppose that the active windows are $[\alpha_1, \beta_1]$, $[\alpha_2, \beta_2]$

[2] By having only unit debts in the instance we have that every PP schedule is also an AoN schedule, and vice versa; for the PP variant we could instead have a single debt $2@[1,T]$.

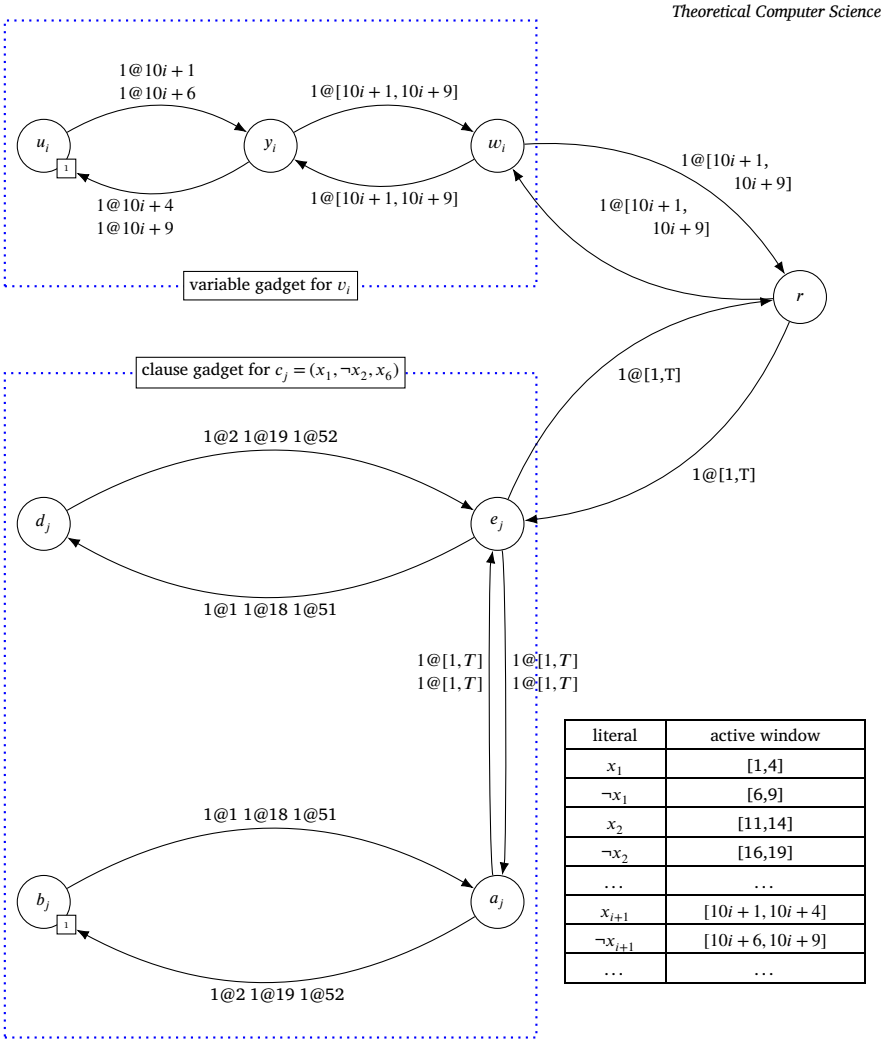


Fig. 10. A reduction from 3-SAT-3 to PERFECT SCHEDULING restricted to multiditrees.

and $[\alpha_3, \beta_3]$, with $\beta_1 < \alpha_2$ and $\beta_2 < \alpha_3$ (we are assuming that our clause has 3 literals but the arguments for clauses with only 2 literals run analogously). If no payment has been received by e_j from r by time $\beta_1 + 1$ then: the €1 at b_j within the clause gadget must have been used in σ:

- to pay the debts of b_j to a_j, a_j to e_j and e_j to d_j at time α_1
- to pay the debts of d_j to e_j, e_j to a_j and a_j to b_j at time $\alpha_1 + 1$,

if the appearance of the corresponding literal is the first; or

- to pay the debts of b_j to a_j, a_j to e_j and e_j to d_j at time $\alpha_1 + 2$
- to pay the debts of d_j to e_j, e_j to a_j and a_j to b_j at time $\alpha_1 + 3 = \beta_1$,

if the appearance of the corresponding literal is the second (note that the payments made towards the debts of a_j to e_j and e_j to a_j are only partial payments). We have an analogous situation as regards the interval $[\alpha_2, \beta_2]$ when no payment has been received by e_j from r by time $\beta_2 + 1$. However, if no payment has been received by e_j from r by time $\beta_3 + 1$ then we obtain a contradiction as the debts from a_j to e_j and from e_j to a_j will have been fully paid and consequently e_j will be bankrupt at time β_3. Hence, within σ, there must be a payment of €1 received by e_j from r and this is the only payment made from r to e_j. Furthermore, there can be no payment from e_j to r strictly before the time at which a payment is made from r to e_j and if a payment is made from e_j to r at the same time that a payment is made from r to e_j then this can be interpreted as no resource leaving or entering the clause gadget, with the external resource satisfying both debts involving r and e_j, which cannot be the case.

Consider now a variable gadget, corresponding to the variable v_i. At times $t \in [0, 10i] \cup [10i+4, 10i+5] \cup [10i+9, T]$ there must be cash assets of (at least) €1 at node u_i. So, outside the active windows of the literals associated with the variable v_i, there must be at least €1 of cash assets at the node u_i. Also, note that the €1 originating at node u_i can only leave and return to the variable gadget at some times in $[10i+1, 10i+5]$ or in $[10i+6, 10i+9]$ but not both. Consequently, at any time, there is at most €1 of resource that originated within a variable gadget outside that particular variable gadget.

Consider the time interval $[1, 9]$. Within this time interval, the €1 originating at u_1 is the only euro that might be possibly 'outside' the variable gadget corresponding to v_1. Call this euro E. Suppose E leaves its variable gadget at some time in $[1, 4]$. It needs to be back at y_1 by time 4 (and will never leave the variable gadget again). Suppose that E is paid from r to e_j, for some j, within the time interval $[1, 4]$. If the literal x_1 is not in c_j then all debts involving only a_j and b_j and all debts involving only d_j and e_j will be due at some time outside $[1, 4]$ and so E is of no use to the clause gadget for c_j. If E is paid from r to e_j at time 1 (resp. by time 3) and the literal x_1 is in clause c_j as the first (resp. second) appearance then E can be used to pay the debts from e_j to d_j at time 1 (resp. time 3) and from d_j to e_j at time 2 (resp. time 4). Note that E cannot be used to pay the debt from a_j to b_j at time 2 or time 4 as it would not get back to its variable gadget by time 4. Given that E leaves the clause gadget by time 4, no more resource either enters or leaves the clause gadget. Alternatively, suppose that E leaves the variable gadget corresponding to v_1 at some time in $[6, 9]$. Exactly the same argument can be made as that above except with respect to the literal $\neg x_1$ appearing in some clause or other.

Let us continue with the time interval $[11, 19]$ and the €1 originating at u_2. An analogous argument to that above holds. Note that all clause gadgets that were previously 'visited' by E, above, are now 'closed' in that they accept or eject no further resource. Indeed, an analogous argument to that above holds for every euro originating at some node u_i. As σ is a perfect schedule, every clause gadget must be visited by some euro originating in some variable gadget and the particular literal corresponding to the active window during which the euro left its variable gadget must appear in the clause. Define the truth assignment X via: $X(v_i) = True$ (resp. $False$) if the euro from the variable gadget corresponding to v_i leaves its variable gadget during the active window $[10i+1, 10i+4]$ (resp. $[10i+6, 10i+9]$) and visits a clause gadget, with any Boolean variables v_i for which $X(v_i)$ has not been defined such that $X(v_i)$ is defined arbitrarily. Given the above discussion, it should be clear that X satisfies ϕ. □

Claim 9. *If ϕ is a yes-instance of 3-SAT-3 then (G, D, A^0) is a yes-instance of PERFECT SCHEDULING.*

Proof. Suppose that X is a satisfying truth assignment for ϕ. Define the schedule σ as follows.

If $X(v_i) = True$ then at time $10(i-1)+1$, the euro originating at u_i is paid from u_i to y_i to w_i to r.

- If the clause c_j containing the first appearance of the literal x_i exists and has not been visited by some euro originating within a variable gadget then at time $10i+1$, the euro is paid from r to e_j and on to d_j. At time $10i+2$, the same euro is paid from d_j to b_j and on to r.
- If there is no clause containing the literal x_i or if the clause gadget corresponding to the first appearance of x_i has already been visited in the schedule σ by some euro originating within a variable gadget, then do nothing.
- If the clause c_j containing the second appearance of the literal x_i exists and has not been visited by some euro originating within a variable gadget then at time $10i+3$, the euro is paid from r to e_j and on to d_j. At time $10i+4$, the same euro is paid from d_j to b_j and on to r.
- If there is no clause containing the literal x_i or if there is no second appearance of the literal x_i or if the clause gadget corresponding to the second appearance of x_i has already been visited in the schedule σ by some euro originating within a variable gadget, then do nothing.
- Our euro at r is paid at time $10i+4$ from r to w_i to y_i to u_i.

If $X(v_i) = False$ then at time $10i+6$, the euro originating at u_i is paid from u_i to y_i to w_i to r.

- If the clause c_j containing the first appearance of the literal $\neg x_i$ exists and has not been visited by some euro originating within a variable gadget then at time $10i+6$, the euro is paid from r to e_j and on to d_j. At time $10i+7$, the same euro is paid from d_j to e_j and on to r.
- If there is no clause containing the literal $\neg x_i$ or if the clause gadget corresponding to the first appearance of $\neg x_i$ has already been visited in the schedule σ by some euro originating within a variable gadget, then do nothing.
- If the clause c_j containing the second appearance of the literal $\neg x_i$ exists and has not been visited by some euro originating within a variable gadget then at time $10i+8$, the euro is paid from r to e_j and on to d_j. At time $10i+9$, the same euro is paid from d_j to e_j and on to r.
- If there is no clause containing the literal $\neg x_i$ or if there is no second appearance of the literal $\neg x_i$ or if the clause gadget corresponding to the second appearance of $\neg x_i$ has already been visited in the schedule σ by some euro originating within a variable gadget, then do nothing.
- Our euro at r is paid at time $10i+9$ from r to w_i to y_i to u_i.

Within any clause gadget corresponding to c_j, the euro originating at b_j is used to pay the debts corresponding to the literal not addressed by the euro from a variable gadget. It can easily be seen that σ is valid and a perfect schedule. □

The result follows, given that the construction of (G, D, A^0) can clearly be undertaken in polynomial-time. □

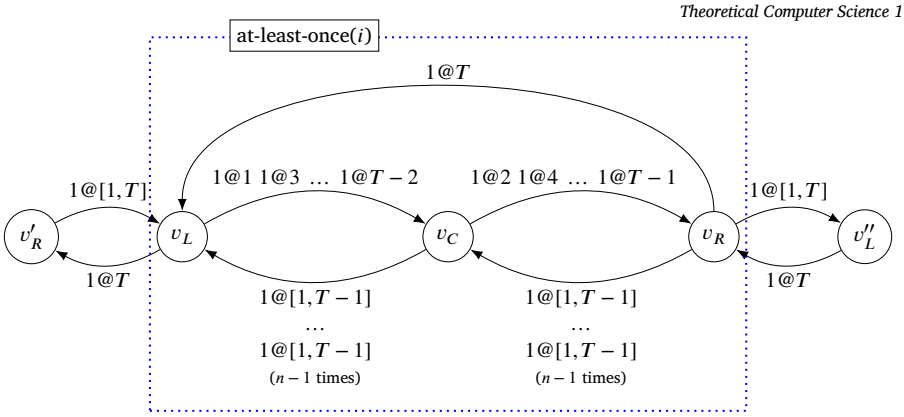


Fig. 11. An at-least-once gadget. Note that as $T = 2n + 1$ there are, n €1 debts owed by v_L to v_C and by v_C to v_R.

In all the above results, the input IDM instance is allowed to have unlimited (i.e., unbounded) total initial assets which might be unrealistic in practically relevant financial systems. We now show that even in the highly restricted case where there is just €1 in initial external assets in total, PERFECT SCHEDULING still remains NP-complete in the AoN and PP variants.

Theorem 5. *The problem PERFECT SCHEDULING is NP-complete in the AoN and PP variants even when the total value of all initial external assets in any instance is €1.*

Proof. The following proof applies to both the AoN and PP variants. Our reduction is a reduction from the problem SOURCED HAMILTONIAN PATH defined as follows.

SOURCED HAMILTONIAN PATH

Instance: a digraph H and a vertex x
Yes-instance: there exists a Hamiltonian path in H with source x.

This problem can be trivially shown to be NP-complete by reducing from the standard problem of deciding whether a digraph has a Hamiltonian path [26].

Let H be some digraph on the n vertices $\{x_i : 1 \leq i \leq n\}$ and w.l.o.g. let $x = x_1$. In order to describe our reduction, we first describe a gadget, namely the *at-least-once* gadget. We have one of these gadgets for each vertex of H and we refer to the gadget corresponding to the vertex x_i of H as at-least-once(i). Our at-least-once gadget can be defined as in Fig. 11 where the value T is defined as $2n + 1$. Note that the gadget is exactly the nodes and debts within the blue dotted box and so contains its own copies of nodes v_L, v_C and v_R and the $4n - 1$ debts involving them. The nodes v'_R and v''_L are not part of the gadget but are nodes in other gadgets as we now explain.

Set $T = 2n + 1$. Our IDM instance (G, D, A^0) can be defined as follows:

- there is the at-least-once(i) gadget, for $1 \leq i \leq n$
- for every edge (x_i, x_j) of H, there is a debt of €1 from v_R of at-least-once(i) to v_L of at-least-once(j) to be paid in the interval $[1, T]$ and a debt of €1 from v_L of at-least-once(j) to v_R of at-least-once(i) to be paid at time T
- all nodes have initial external assets of 0 except for node v_L of at-least-once(1) which has initial external assets of 1.

We refer to the single euro of initial external assets as the *initial euro*. The IDM instance (G, D, A^0) can clearly be constructed in time polynomial in n.

Claim 10. *If (G, D, A^0) is a yes-instance of PERFECT SCHEDULING then (H, x) is a yes-instance of SOURCED HAMILTONIAN PATH.*

Proof. Note that strictly prior to time T, the only payment-cycles that can exist within G involve the two nodes v_L and v_C of some at-least-once gadget or the two nodes v_C and v_R of some at-least-once gadget. Note also that within some gadget there are n debts of €1 from v_L to v_C needing to be satisfied and $n - 1$ debts of €1 from v_C to v_L. An analogous statement can be made as regards v_C and v_R. Consequently, in order to satisfy all debts involving v_L and v_C within some gadget, at some odd time in $[1, T - 1]$, the initial euro must be within that gadget so as to satisfy some debt from v_L to v_C (by moving from v_L to v_C). Similarly, at some even time in $[1, T - 1]$, the initial euro must be within the gadget so as to satisfy some debt from v_C to v_L (by moving from v_C to v_R). Moreover, at any time in $[1, T - 1]$, the initial euro can only satisfy at most one of the debts mentioned above. Hence, given that $T = 2n + 1$, at any time in $[1, T - 1]$, the initial euro must be satisfying exactly one of the above debts.

Suppose that the initial euro satisfies some debt from v_L to v_C in some at-least-once gadget at time t. As the initial euro needs to satisfy one of the above debts at time $t + 1$, we need that at time $t + 1$ the initial euro satisfies a debt from v_C to v_R in the same

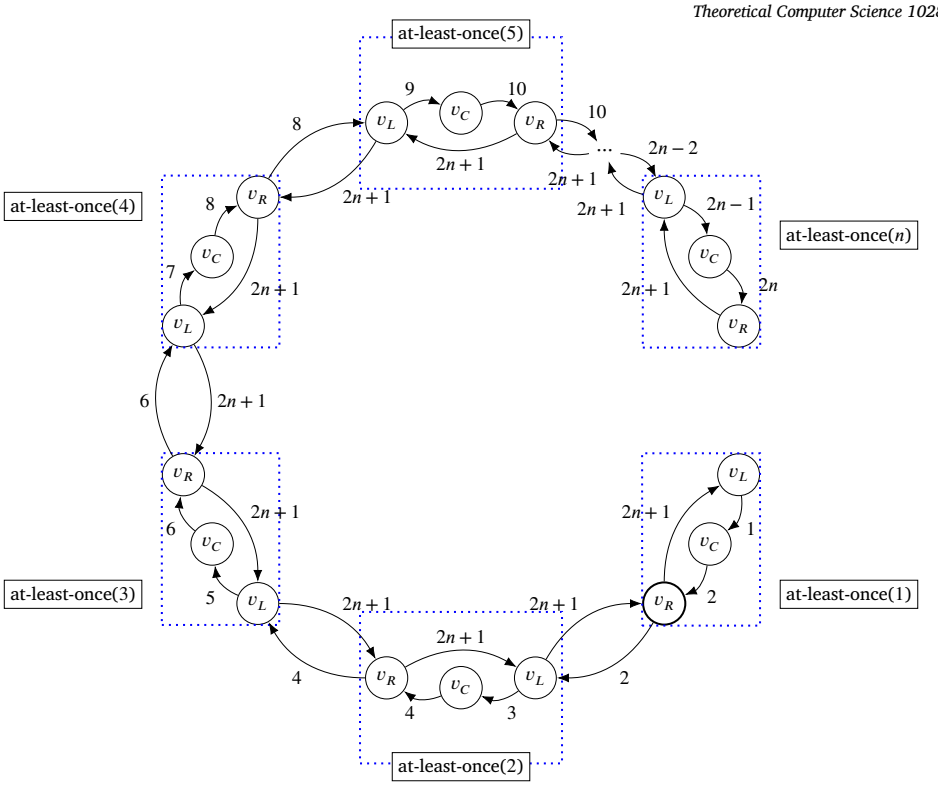


Fig. 12. The path taken by the initial euro straightforwardly corresponds to a sourced Hamiltonian path in the original graph.

gadget. Also, it cannot be the case that a debt from v_C to v_R in some gadget is satisfied by the initial euro before the euro satisfies some debt from v_L to v_R in the same gadget. As the initial euro starts at v_L in at-least-once(1), our schedule must be such that the initial euro 'moves' through the at-least-once gadgets corresponding to every node, entering at the node v_L and exiting at the node v_R. Consequently, its path within G corresponds to a path $x = x_1, x_2, \ldots, x_n$ in H where every node on this path is distinct and where there is a directed edge from node x_i to x_{i+1}, for $1 \leq i \leq n-1$; that is, a Hamiltonian path in H with source x. □

Claim 11. *If (H, x) is a yes-instance of* Sourced Hamiltonian Path *then (G, D, A^0) is a yes-instance of* Perfect Scheduling.

Proof. Let $x = x_1, x_2, \ldots, x_n$ be a Hamiltonian path P in the digraph H. Consider the following schedule σ:

- the initial euro is used so as to pay the following debts:
  - €1 at time $2i-1$ from v_L to v_C in at-least-once(x_i) and €1 at time $2i$ from v_C to v_R in at-least-once(x_i), for $1 \leq i \leq n$
  - €1 at time $2i$ from v_R in at-least-once(x_i) to v_L in at-least-once (x_{i+1}), for $1 \leq i \leq n-1$
- for any v_L and v_C within some at-least-once gadget and at any odd time $t < T$ when a debt is not being paid from v_L to v_C using the initial euro, there is a payment-cycle consisting of payments of €1 from v_L to v_C and of €1 from v_C to v_L
- for any v_C and v_R within some at-least-once gadget and at any even time $t < T$ when a debt is not being paid from v_C to v_R using the initial euro, there is a payment-cycle consisting of payments of €1 from v_C to v_R and of €1 from v_R to v_C
- for any edge (x_i, x_j) of H that does not feature in the Hamiltonian path P, there is a payment-cycle consisting of payments at time T of €1 from v_R in at-least-once(x_i) to v_L in at-least-once(x_j) and of €1 from v_L in at-least-once(x_j) to v_R in at-least-once(x_i)
- the initial euro is used so as to pay the following debts:
  - €1 at time T from v_R to v_L in at-least-once(x_i), for $1 \leq i \leq n$
  - €1 at time T from v_L in at-least-once(x_i) to v_R in at-least-once(x_{i-1}), for $2 \leq i \leq n$.

The 'path' taken by the initial euro within (G, D, A^0) can be visualized as in Fig. 12 where the debt arrows are tagged with the time of payment. It is clear that σ is valid and a perfect schedule. □

Our main result follows. □

Of course, one can obtain additional restrictions on the structure of the IDM instances for Perfect Scheduling in Theorem 5 by looking at NP-completeness results relating to Sourced Hamiltonian Path on restricted digraphs; however, we have refrained from doing so (as nothing of any significance emerges).

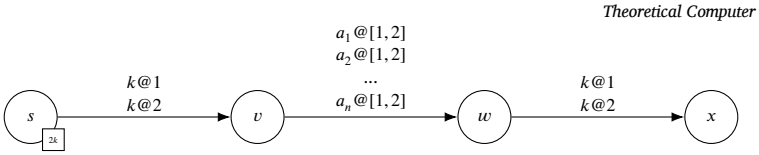


Fig. 13. An IDM instance encoding an instance $\{a_1, \ldots, a_n\}$ of PARTITION with sum $2k$.

We can constraint the digraph G of an instance (G, D, A^0) of PERFECT SCHEDULING even further in the AoN variant; indeed, so that it is always a directed path of length 3. The price we pay is that the initial external assets are potentially large.

Theorem 6. *Consider the problem PERFECT SCHEDULING restricted so that every instance (G, D, A^0) is such that G is a directed path of bounded length.*

(*a*) *If, further, T is restricted to be 2 then the resulting problem is weakly NP-complete in the AoN variant.*
(*b*) *If there are no restrictions on T then the resulting problem is strongly NP-complete in the AoN variant.*

Proof. Consider (*a*). We reduce from the problem PARTITION defined as follows (and proven in [26] to be weakly NP-complete).

PARTITION

Instance: a multi-set of integers $S = \{a_1, a_2, \ldots, a_n\}$ with $sum(S) = 2k$
Yes-instance: there exists a partition of S into two subsets S_1 and S_2 such that $sum(S_1) = sum(S_2) = k$.

In general, an instance $S = \{a_1, a_2, \ldots, a_n\}$ has size nb where b is the least number of bits required to express any of the integers of S in binary.

Let S be an instance of Partition of size nb. Consider the IDM instance (G, D, A^0) in Fig. 13. Note that the time taken to construct (G, D, A^0) from S is polynomial in nb.

Claim 12. *If (G, D, A^0) is a yes-instance of PERFECT SCHEDULING then S is a yes-instance of PARTITION.*

Proof. Suppose that there is a valid schedule σ for (G, D, A^0) that is a perfect schedule. It must be the case that the total amount paid by s at time 1 is $\in k$ and that this payment is immediately paid by v to w and from w to x at time 1. As we are in the AoN variant, it must be the case that the sum of a subset of integers of S amounts to k. An analogous argument applies to the payments made at time 2 and the remainder of the integers in S. the claim follows. □

Claim 13. *If S is a yes-instance of PARTITION then (G, D, A^0) is a yes-instance of PERFECT SCHEDULING.*

Proof. Suppose that S_1 and S_2 is a partition of S such that $\sum(S_1) = \sum(S_2) = k$. Define the schedule σ so that:

- at time 1: s pays $\in k$ to v; if $a_i \in S_1$, for $1 \le i \le n$, then v pays $\in a_i$ to w; and w pays $\in k$ to x
- at time 2: s pays $\in k$ to v; if $a_i \in S_2$, for $1 \le i \le n$, then v pays $\in a_i$ to w; and w pays $\in k$ to x.

The schedule σ is a valid perfect schedule. □

The proof of (*a*) follows. Now consider (*b*). We reduce from the strongly NP-complete problem 3-PARTITION defined as follows (see [27]).

3-PARTITION

Instance: a multi-set of integers $S = \{a_1, a_2, \ldots, a_{3m}\}$, for some $m \ge 1$, with $sum(S) = mk$
Yes-instance: there exists a partition of S into m triplets $S_1, S_2, \ldots S_m$ such that $sum(S_i) = k$, for each $1 \le i \le m$.

In general, an instance $S = \{a_1, a_2, \ldots, a_{3m}\}$ has size mb where b is the least number of bits required to express any of the integers of S in binary.

Let S be an instance of 3-PARTITION of size mb. By multiplying all integers by 4 if necessary, we may assume that every integer of S is divisible by 4 as is k. Consider the IDM instance (G, D, A^0) in Fig. 14. Note that the time taken to construct (G, D, A^0) from S is polynomial in mb.

Claim 14. *If (G, D, A^0) is a yes-instance of PERFECT SCHEDULING then S is a yes-instance of 3-PARTITION.*

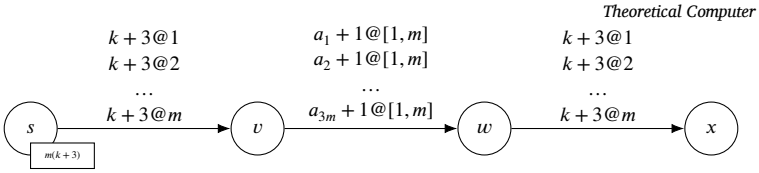


Fig. 14. An IDM instance showing the reduction from 3-PARTITION to AON PERFECT SCHEDULING.

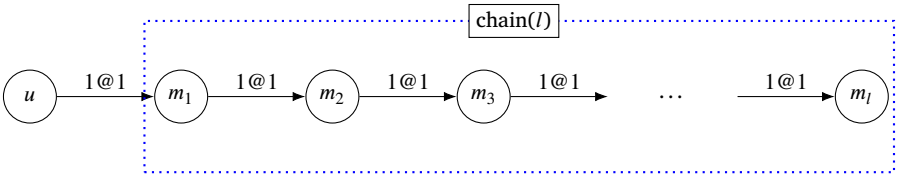


Fig. 15. The chain gadget. In any valid schedule, either $p^1_{u,m_1} < 1$ and all vertices m_i are bankrupt, or $p^1_{u,m_1} = 1$ and no vertices m_i are bankrupt.

Proof. Suppose that there is a valid schedule σ for (G, D, A^0) that is a perfect schedule. It must be the case that the total amount paid by s at time i, for every $1 \leq i \leq m$, is $\in k+3$ and that this payment is immediately paid by v to w and by w to x. Suppose that the payment at time i by v to w pays at least 4 of the debts due. So, there exists another time j, say, where the payments made by v to w pay at most 2 debts. So, we have that either $a_\alpha + a_\beta + 2 = k+3$ or $a_\alpha + 1 = k+3$, for some $1 \leq \alpha \neq \beta \leq 3m$. This yields a contradiction as the right-hand sides of these equations are equivalent to 3 modulo 4 whereas the left-hand sides are not. So, at any time i, for $1 \leq i \leq m$, exactly three debts are paid by v to w at time i. If $1 \leq \alpha, \beta, \gamma \leq 3m$ are distinct so that debts of monetary amounts $a_\alpha + 1$, $a_\beta + 1$ and $a_\gamma + 1$ are paid by v to w at some time then $a_\alpha + 1 + a_\beta + 1 + a_\gamma + 1 = k+3$; that is, $a_\alpha + a_\beta + a_\gamma = k$. So, we have a yes-instance of 3-PARTITION. The claim follows. □

Claim 15. *If S is a yes-instance of 3-PARTITION then (G, D, A^0) is a yes-instance of PERFECT SCHEDULING.*

Proof. Suppose that S can be partitioned into triplets so that the sum of the integers in each triplet is k; so, suppose that $a_{\alpha_i} + a_{\beta_i} + a_{\gamma_i} = k$, for each $1 \leq i \leq m$, where $S = \{a_{\alpha_i}, a_{\beta_i}, a_{\gamma_i} : 1 \leq i \leq m\}$. Define the following schedule σ: at time i, for each $1 \leq i \leq m$, s pays $\in k+3$ to v; v pays $\in a_{\alpha_i} + a_{\beta_i} + a_{\gamma_i} + 3 = k+3$ to w; and w pays $\in k+3$ to x. The schedule σ is valid and a perfect schedule. The claim follows. □

The main result follows. □

3.3. Hardness results for BANKRUPTCY MAXIMIZATION

We now turn to BANKRUPTCY MAXIMIZATION.

Theorem 7. *The problem BANKRUPTCY MAXIMIZATION is NP-complete in the AoN, PP and FP variants even when for an instance (G, D, A^0): $T = 2$; G is a directed acyclic graph with out-degree at most 2, in-degree at most 3; all monetary debts are at most $\in 2$ per edge; and initial external assets are at most $\in 3$ per bank.*

Proof. We build a polynomial-time reduction from the problem 3-SAT-3, with the usual restrictions on instances (see the proof of Theorem 1). Suppose that we have some instance ϕ of 3-SAT-3 where there are n Boolean variables and m clauses. We start with the *chain gadget* chain(l), where $l \geq 1$, as portrayed in Fig. 15 (note that the gadget is the path of l nodes within the blue dotted box). The key point about any chain gadget is that in some schedule: if at time 1, node u does not make a payment to node m_1 then u and all the nodes of the chain gadget are bankrupt; and if at time 1, u pays its debt to m_1 then none of the nodes of the chain gadget is bankrupt. As in our proof of Theorem 2, we first work in the PP variant unless otherwise stated, though the reasoning will apply to the AoN and FP variants as well.

We now define variable nodes $\{s_i : 1 \leq i \leq n\}$, literal nodes $\{x_i, \neg x_i : 1 \leq i \leq n\}$, clause nodes $\{q_j : 1 \leq j \leq m\}$ and a sink node d analogously to as in the proof of Theorem 1 and include the debts as depicted in Fig. 16 so as to obtain our IDM instance (G, D, A^0). Note that the chain gadgets corresponding to the different literal nodes are all distinct and $|c_j|$ denotes the number of literals in the clause c_j of ϕ.

Note that because all debts of (G, D, A^0) are due at an exact time, rather than over an interval, reasoning in the AoN variant is identical to reasoning in the PP variant.

Claim 16. *In any valid schedule σ for (G, D, A^0) in which $c \geq 0$ variable nodes s_i either pay $\in 3$ to x_i or $\in 3$ to $\neg x_i$:*

- *all n variable nodes are bankrupt*

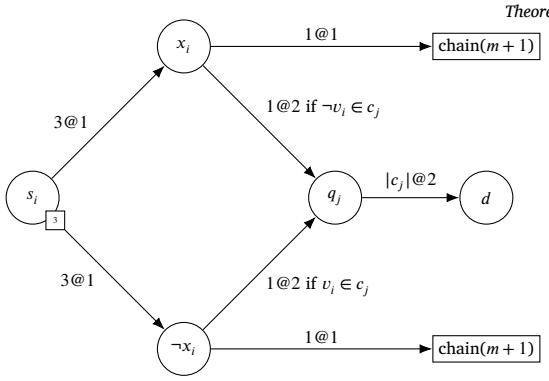


Fig. 16. An IDM instance illustrating the reduction from 3-SAT 3 to Bankruptcy Maximization, using chain gadgets.

- *exactly n literal nodes are bankrupt*
- *exactly $c(m+1)$ chain nodes are bankrupt.*

Consequently, this amounts to exactly $2n + c(m+1)$ bankrupt nodes with any other bankrupt nodes necessarily being clause nodes.

Proof. Suppose that in the valid schedule σ, s_i, for some $1 \leq i \leq n$, pays €1 to a node from $\{x_i, \neg x_i\}$ at time 1 and €2 to the other node from $\{x_i, \neg x_i\}$ at time 1. Since both x_i and $\neg x_i$ have €1 at time 1, both must pay €1 to their corresponding chain gadget; so, none of the nodes of either of these chain gadgets is bankrupt. As any variable and its negation both appear at least once in some clause of ϕ, exactly one of the nodes x_i and $\neg x_i$ is bankrupt at time 2.

Alternatively, suppose that s_j, for $1 \leq j \leq n$, pays €3 to either x_j or $\neg x_j$ at time 1. So, the literal node to which s_j makes no payment is bankrupt at time 1 as are all the nodes of its corresponding chain gadget. As any variable and its negation both appear at least once in some clause of ϕ, exactly one of the nodes x_j and $\neg x_j$ is bankrupt at time 2.

In any case, we have: n variable nodes that are bankrupt; n literal nodes that are bankrupt; and $c(m+1)$ chain nodes that are bankrupt. This results in $2n + c(m+1)$ bankrupt nodes. As the sink node d cannot be bankrupt, the claim follows. □

Claim 17. *If $((G, D, A^0), 2n + n(m+1) + m)$ is a yes-instance of Bankruptcy Maximization then ϕ is a yes-instance of 3-Sat-3.*

Proof. Suppose that there exists a valid schedule σ that results in at least $2n + n(m+1) + m$ bankruptcies. So, by Claim 16, every variable node s_i pays €3 to either x_i or $\neg x_i$ and also every clause node q_j is bankrupt in σ. The reason a clause node q_j is bankrupt is because there is some literal v_i or $\neg v_i$ in clause c_j but where $\neg x_i$ or x_i, respectively, receives no payment from s_i. Define the truth assignment X on the variables of ϕ so that $X(v_i) = True$ iff x_i receives a payment of €3 from s_i, for $1 \leq i \leq n$. This truth assignment satisfies every clause of ϕ. □

Claim 18. *If ϕ is a yes-instance of 3-Sat-3 then $((G, D, A^0), 2n + n(m+1) + m)$ is a yes-instance of Bankruptcy Maximization.*

Proof. Suppose that there is a satisfying truth assignment X for ϕ. Consider the following schedule σ:

- at time 1, every s_i pays: €3 to x_i if $X(v_i) = True$; and €3 to $\neg x_i$ if $X(v_i) = False$
- if x_i (resp. $\neg x_i$) received €3 from s_i at time 1 then:
  - at time 1, it pays €1 to its corresponding chain gadget so as to satisfy all debts in the gadget
  - at time 2, it pays €1 to each clause node q_j for which the literal $\neg v_i \in c_j$ (resp. $v_i \in c_j$)
- if x_i (resp. $\neg x_i$) received no payment from s_i at time 1 then at times 1 and 2 then it can make no payments
- each q_j makes a payment of however many euros it has to d at time 2 (note that it never received more than €$|c_j|$).

The schedule σ is clearly valid. By Claim 16, we have at least $2n + n(m+1)$ bankrupt nodes with any additional bankrupt nodes necessarily clause nodes. Consider some clause node q_j containing some literal v_i so that $X(v_i) = True$. By definition, $\neg x_i$ receives no payment from s_i at time 1 and so the debt of €1 at time 2 from $\neg x_i$ to q_j is not paid. Consequently, q_j is bankrupt. Hence, we have exactly $2n + n(m+1) + m$ bankrupt nodes and the claim follows. □

Note that the above reasoning clearly holds in the AoN variant (since in the schedules we consider all debts are paid either in full or not at all) as well as in the FP variant (since in order for s_i to bankrupt one of the chains attached to x_i and $\neg x_i$ it must pay *strictly* less than £1 to the bankrupt node and hence *at least* £2 to the "surviving" node). As the construction of $((G, D, A^0), 2n + n(m+1) + m)$ can be completed in time polynomial in n, the result follows. □

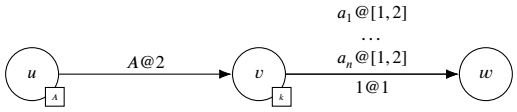


Fig. 17. An IDM instance corresponding to an instance of SUBSET SUM.

Just as we did with PERFECT SCHEDULING in Theorem 6, we can restrict BANKRUPTCY MAXIMIZATION in the AoN variant so that any IDM (G, D, A^0) in any instance is such that G is a directed path of bounded length (here 2).

Theorem 8. *Consider the problem BANKRUPTCY MAXIMIZATION restricted so that every instance $((G, D, A^0), k)$ is such that G is a directed path of length 3. If, further, T is restricted to be 2 then the resulting problem is weakly NP-complete in the AoN variant.*

Proof. We reduce from the weakly NP-complete problem SUBSET SUM defined as follows (see [27]).
SUBSET SUM

Instance: a multi-set of integers $S = \{a_1, a_2, \ldots, a_n\}$ and an integer k
Yes-instance: there exists a subset S_1 of S so that the sum of the numbers in S_1 is k.

In general, an instance $S = \{a_1, a_2, \ldots, a_n\}$ has size nb where b is the least number of bits required to express any of the integers of S in binary. Let S be an instance of SUBSET SUM of size nb. By doubling all integers if necessary, we may assume that every integer of S is at least 2. Consider the IDM instance (G, D, A^0) in Fig. 17 (for which $T = 2$). The value A in Fig. 17 is the sum of all integers in S. Note that the time taken to construct (G, D, A^0) from S is polynomial in nb.

Claim 19. *If $((G, D, A^0), 1)$ is a yes-instance of BANKRUPTCY MAXIMIZATION then S is a yes-instance of SUBSET SUM.*

Proof. Suppose that σ is a valid schedule within which there is at least 1 bankruptcy in the IDM instance (G, D, A^0). The nodes u and w are never bankrupt; so, v must be bankrupt within σ. At time 2, the node v necessarily pays off all of the unpaid debts to w of monetary amount greater than €1, as it receives sufficient funds from u at time 2 to do this. Hence, v must be bankrupt at time 1; that is, v does not pay its debt to w of monetary amount €1 at time 1. As σ is valid, v is not withholding at time 1 and the only way for this to happen is for v to pay debts to w amounting to €k. That is, we have a subset of integers of S whose total sum is k; that is, S is a yes-instance of SUBSET SUM. The claim follows. □

Claim 20. *If S is a yes-instance of SUBSET SUM then $((G, D, A^0), 1)$ is a yes-instance of BANKRUPTCY MAXIMIZATION.*

Proof. Suppose that the subset S_1 of S is such that $sum(S_1) = k$. W.l.o.g. let $S_1 = \{a_1, a_2, \ldots, a_r\}$. Define the schedule σ as: at time 1, v pays the debts $a_1@[1,2], \ldots, a_r@[1,2]$; and at time 2, u pays its debt to v and v pays the debts $a_{r+1}@[1,2], \ldots, a_n@[1,2]$. Note that this is a valid schedule, as no node is withholding at any time, within which v is bankrupt. The claim follows. □

The main result follows. □

3.4. Polynomial-time algorithms

In this section we show that BAILOUT MINIMIZATION in the FP variant is solvable in polynomial-time and also that BAILOUT MINIMIZATION in the PP variant is solvable in polynomial-time when our IDM instances are restricted to out-trees. We begin with the FP variant result.

Theorem 9. *The problem BAILOUT MINIMIZATION in the FP variant is solvable in polynomial time.*

Proof. A solution to FP BAILOUT MINIMIZATION is a bailout vector B of size $|V|$ together with a schedule σ consisting of $|E|T$ payment values p_e^t. We describe below how an instance G, D, A^0, b of BAILOUT MINIMIZATION can be encoded as a linear program (LP), which can then be solved in polynomial time.

Our variables are:

- Bailout variables $\{B[v] | v \in V\}$ (altogether $|V|$ variables),
- Payment variables $\{p_e^t | e \in e, t \in [T]\}$ (altogether $|E|T$ variables), and
- Income variables $\{I_v^t | v \in V, t \in [T]\}$, outgoing variables $\{O_v^t | v \in V, t \in [T]\}$, and cash asset variables $\{c_v^t | v \in V, t \in [0, T]\}$ (altogether $3 \cdot |V| \cdot T$ variables).

In the below, for $a, b \in \mathbb{N}_0$, $[a, b]$ denotes the set $\{a, a+1, \ldots, b\}$, and we write $[b]$ as shorthand for $[1, b]$. Our constraints are:

- The total bailout is at most b:

$$\sum_v B[v] \leq b$$

- The starting cash assets of a node (at time 0) are its external assets (specified by A^0) plus any bailout it receives. For each $v \in V$:

$$c_v^0 = A^0[v] + B[v]$$

- No debt is paid early, and all payments are non-negative. For each $e \in E$ and $t \in [0, T]$:

$$p_e^t \begin{cases} = 0, & \text{if } t < D_{t_1}(e) \\ \geq 0, & \text{otherwise} \end{cases}$$

- The income (resp. outgoings) of a node at some time is obtained by summing over payments into (resp. out of) that node at each time. These then can be used to compute external (cash) assets at all nodes and times. For each $v \in V$ and $t \in [T]$:

$$I_v^t = \sum_{e \in E_{in}(v)} p_e^t \quad (\text{resp. } O_v^t = \sum_{e \in E_{out}(v)} p_e^t)$$
$$e_v^t = e_v^{t-1} + I_v^t - O_v^t$$

- No bank has negative assets at any point. For each $v \in V, t \in [T]$:

$$e_v^t \geq 0$$

- Each debt is paid in full within its interval. This guarantees that there are no bankruptcies in the schedule (and that no banks are withholding, a validity constraint). For each $e \in E$ with $D(e) = (a, t_1, t_2)$:

$$\sum_{t \in [t_1, t_2]} p_e^t = a$$

Recall from our discussion of canonical instances in Section 2.4 that we may assume T is at most $2|E|$. Then we have $O(nm + m^2)$ variables and $O(nm + m^2)$ constraints. If the largest integer in the input instance G, D, A^0, b required β bits to encode, then our constructed LP has size polynomial in $n + m + \beta$. Any assignment to B and to the payment variables p_e^t satisfying the above is necessarily a perfect valid schedule for $((G, D, A^0), b)$. As linear programs can be solved in polynomial-time, our result follows. □

Note the limitations of the use of linear programming for other problems. For Bailout Minimization in the PP variant, proceeding as in the proof of Theorem 9 results is an integer linear program, the solution of which is NP-complete in general. Moreover, we have already proven Perfect Scheduling, the special case of Bailout Minimization with b fixed to 0, to be NP-complete in the PP variant through the proofs in Theorems 3, 4 and 5. As regards trying to use linear programming for Bankruptcy Minimization in the FP variant, it is not possible to express a constraint on the number of bankruptcies through a linear combination of the payment variables; indeed, we have already proven Bankruptcy Minimization in the FP variant to be NP-complete in Theorems 1 and 2.

For the AoN and PP variants, by restricting the temporal properties of the IDM instances considered, we obtain tractability of Bailout Minimization, namely when all debts are due at an exact time.

Theorem 10. *The problem (AoN/PP/FP) Bailout Minimization is solvable in polynomial time when restricted to inputs (G, D, A^0) such that $D_{t_1} = D_{t_2}$.*

Proof. Let (G, D, A^0) be an IDM instance satisfying the above. In such an instance, all debts are due at an exact point in time, rather than an interval. For convenience, we use D_t as shorthand for either of D_{t_1} or D_{t_2}. By definition, for any bailout vector B (including the all-zero vector) a perfect schedule for $(G, D, A^0 + B)$ is one in which every debt is paid in full and on time. Let σ be the schedule defined by $p_e^{D_t(e)} = D_a(e)$ for each edge e, with all other payment variables equal to zero. Clearly, for any vector B, a perfect schedule for $(G, D, A^0 + B)$ exists if and only if σ is a valid schedule (and hence a perfect schedule).

Moreover, we can efficiently compute a vector B of minimum sum such that σ is a perfect schedule for $(G, D, A^0 + B)$. For each vertex v and time t, compute c_v^t under σ for the instance (G, D, A^0). Note that if (G, D, A^0) does not admit a perfect schedule then c_v^t will be negative for some v and t, and σ is not a valid schedule for that instance (without a bailout). Denote the minimum (again, possibly negative) cash assets of v at any time by $c_v^{\min}$. Compute $b_v := \max(-1 \cdot c_v^{\min}, 0)$ for each v, and let $B = (b_v | v \in V)$. By construction, σ is a perfect schedule for $(G, D, A^0 + B)$, and σ is not a perfect schedule for $(G, D, A^0 + B')$ for any B' with $\sum(B') < \sum(B)$.

All of our arguments hold in all three variants (AoN, PP, and FP), and the result follows. □

Interestingly, Theorem 10 is the only positive result we derive for the All-or-Nothing setting. We also obtain tractability results for the problem PP Bailout Minimization if we restrict the structure of IDM instances.

Theorem 11. *The problem* BAILOUT MINIMIZATION *in the PP variant is solvable in polynomial-time when our IDM instances are restricted to out-trees.*

Proof. Let $((G, D, A^0), b)$ be an instance of BAILOUT MINIMIZATION so that G is an out-tree. Suppose that G has node set $\{u_i : 1 \leq i \leq n\}$. We need to decide whether we can increase the initial external assets of each node u_i by b_i so that $\sum_i b_i \leq b$ and $(G, D, A^0 + B)$ has a perfect schedule, where $B = (b_1, b_2, \ldots, b_n)$; that is, whether (G, D, A^0) is 'b-bailoutable' via a *b-bailout vector* B. Our intention is to repeatedly amend (G, D, A^0) so that (G, D, A^0) is 'b-bailoutable' iff the resulting IDM instance is 'b'-bailoutable', for some amended b'; in such a case, we say that the two problem instances are *equivalent*. We will then work with the (simplified) amended instance.

We proceed as follows. First, identify nodes v for which, at any time t, the sum of all debts v must pay by time t minus the sum of all debts which could be paid to v by time t exceeds v's initial external assets c_v^0. We call these nodes *prefix-insolvent*. Note that if a node v is prefix-insolvent then under any perfect schedule σ, we would have $I_v^{[t]} + c_v^0 < O_v^{[t]}$, violating a validity constraint, and hence there is no such perfect schedule. Also note that every insolvent node is prefix-insolvent (namely by taking $t = T$). For any node that is prefix-insolvent, increase the initial external assets by the minimal amount that causes the node to cease to be prefix-insolvent and simultaneously decrease the bailout amount by this value. Our new instance is clearly equivalent with our initial instance. If, in doing this, the bailout amount becomes less than 0 then we answer 'no' and we are done. So, we may assume that none of our nodes is prefix-insolvent.

Before we start, we make a simple amendment to the debts D: we replace any debt from node u to node v of the form $a@[t_1, t_2]$, where $a > 1$, with a distinct debts $1@[t_1, t_2]$. Our resulting instance, with bailout b, is equivalent to our initial instance, with bailout b, as we are working within the PP variant. This amendment simplifies some of the reasoning coming up. Note that it may be necessary to simulate this operation rather than actually performing it (since if a is exponential in the instance size then the operation takes exponential time), but that the reasoning which follows can easily be "scaled up" to deal with non-unit amounts.

Consider a leaf node v and its parent u in the out-tree G. Replace every debt of the form $1@[t_1, t_2]$ from u to v by the debt $1@t_2$ and denote the revised instance by (G, D', A^0). Let σ be a perfect valid schedule for $(G, D, A^0 + B)$ (here, and throughout, we write B to denote some b-bailout vector; that is, some assignment of resource to the nodes of G so that the total bailout amount does not exceed the total available b). Define the schedule σ' for $(G, D', A^0 + B)$ by amending any payment from u to v of some debt $1@[t_1, t_2]$ so that the payment is made at time t_2. The schedule σ' is clearly a perfect valid schedule for $(G, D', A^0 + B)$. Furthermore, any perfect valid schedule for $(G, D', A^0 + B)$ is a perfect valid schedule for $(G, D, A^0 + B)$. Hence, we can replace $((G, D, A^0), b)$ by $((G, D', A^0), b)$, as these instances are equivalent. We can proceed as above for every leaf node and its parent and so assume that all debts from a parent to a leaf are due at some specific time only; that is, have a singular time-stamp and are of the form $1@t$. Note also that no node of G is insolvent.

Suppose that we have two leaf nodes v and w with the same parent u. We can replace v and w with a 'merged' node vw so that all debts from u to v or from u to w are now from u to vw. Our initial problem instance is clearly equivalent to our amended problem instance (note that we never assign bailout resource to either v or w as this is pointless). We can proceed likewise for all such triples (u, v, w). Hence, we may assume that our digraph G is such that: no two leaves have a common parent; all debts to a leaf node have monetary amount €1 and have a singular time-stamp; and no node in G is prefix-insolvent.

Suppose that we have a leaf node w that is the only child of its parent node v whose parent is u (such a node w exists: take a leaf of the tree that is furthest away from the root). We may assume that v has no initial external assets as we would simply use these assets to pay as many debts to w as possible (in increasing order of time-stamp); that is, we could remove all these debts from D along with the corresponding amount from the initial external assets of v. If the result of doing this is that there are no debts from v to w then we remove w from G and any remaining initial external assets from v. We would then repeat all of the above amendments until it is the case that our nodes u, v and w are such that v has no initial external assets.

By the above, every debt from v to w is of the form $1@t$. Let t' be the minimum time-stamp for all debts from v to w and let d_v be a debt from v to w of the form $1@t'$. Consider the debts from u to v: these have the form $1@[t_1, t_2]$. There are various cases:

(*a*) there is a debt d_u from u to v of the form $1@[t_1, t_2]$ where $t_2 \leq t'$
(*b*) there is no debt from u to v of the form $1@[t_1, t_2]$ where $t_2 \leq t'$ but there is a debt from u to v of the form $1@[t_1, t_2]$ where $t_1 \leq t' \leq t_2$
(*c*) there is no debt from u to v of the form $1@[t_1, t_2]$ where $t_1 \leq t'$.

The nodes u, v and w can be visualized as in Fig. 18(d) and the three cases above in Fig. 18(a)–(c).

Case (*a*): We amend G by: introducing a new node v' whose parent is u and a new debt d' from u to v' of the form $1@[t_1, t_2]$; removing the debt d_u from u to v; and removing the debt d_v from v to w. Denote this revised IDM instance by (G', D', A^0). Suppose that there exists a perfect valid schedule σ for $(G, D, A^0 + B)$. Define the schedule σ' for $(G', D', A^0 + B)$ from σ by: changing the payment from u to v, at time t where $t_1 \leq t \leq t_2 \leq t'$ and covering the debt d_u, so that the payment is made from u to v' at time t (so as to cover the new debt d'); and dropping the payment, at time t', that covers the debt d_v. The resulting schedule σ' is clearly valid and perfect. Conversely, suppose that we have a perfect valid schedule σ' for $(G', D', A^0 + B)$. Define the schedule σ for $(G, D, A^0 + B)$ from σ' by: changing the payment from u to v' at time t where $t_1 \leq t \leq t_2 \leq t'$ and covering the debt d', so that the payment is made from u to v at time t, so as to cover the debt d_u; and using the €1 received by v so as to cover the debt d_v from v to w. The resulting schedule σ is clearly a perfect valid schedule for $(G, D, A^0 + B)$. Consequently, $((G, D, A^0), b)$ and $((G', D', A^0), b)$ are equivalent and we can work with $((G', D', A^0), b)$.

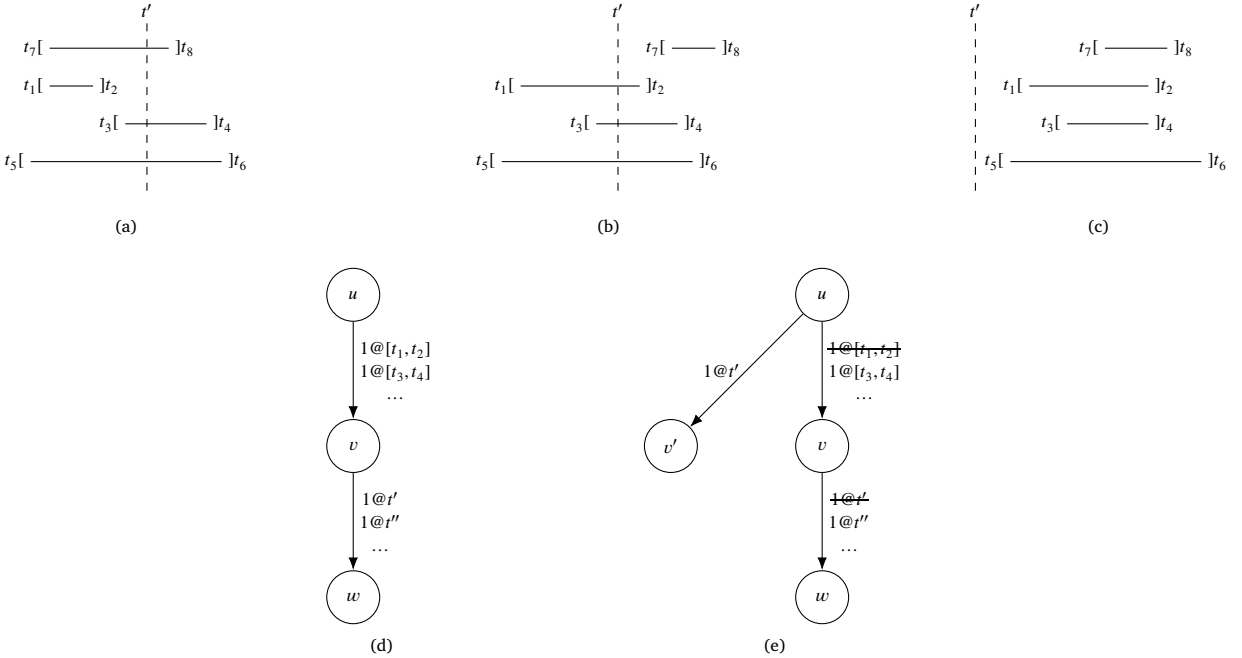


Fig. 18. Cases when a leaf has a parent with one child in our algorithm for PP BAILOUT MINIMIZATION on out-trees.

Case (*b*): Let D_u be the set of debts from u to v and let D_v be the set of debts from v to w. Order the k debts of D_v in increasing order of time-stamp as $d_v = d_1, d_2, \ldots, d_k$ where the corresponding time-stamps are $t' = \bar{t}_1, \bar{t}_2, \ldots, \bar{t}_k$ (there may be repetitions). Suppose that for some $1 \le i \le k$, the number of debts in D_u of the form $1@[t_1, t_2]$ with $t_1 \le \bar{t}_i$ is strictly less than i. Consequently, at time $\bar{t}_i$, the debt d_i cannot be paid and we necessarily need to give v some bailout amount, $\in\!\!c > 1$ say, to cover the c debts that cannot be paid by v at time $\bar{t}_i$. We do this and reduce the overall bailout amount by $\in\!\!c$. We then delete debts $d_1, d_2, \ldots, d_c$ from G and remove the bailout amount of $\in\!\!c$ from v. If doing this results in there being no remaining debts from v to w then we delete w from G. Irrespective of this, the resulting instance $((G', D', A^0), b')$, where $b' = b - c$, is equivalent to $((G, D, A^0), b)$. We would then repeat all of the amendments above and so w.l.o.g. we may assume that we are in Case (*b*) and for every $1 \le i \le k$, the number of debts in D_u of the form $1@[t_1, t_2]$ with $t_1 \le \bar{t}_i$ is at least i. In particular, there are at least k debts in N_u.

Suppose that σ is a valid perfect schedule for $(G, D, A^0 + B)$, for some B where v receives a bailout amount of $\in\!\!c > 0$. Suppose further that there is no B' where there is a valid perfect schedule for $(G, D, A^0 + B')$ with v receiving a bailout of less than $\in\!\!c$. The reason that v receives the bailout amount of $\in\!\!c$ is that in the schedule σ, if we ignore the payments by v that use the bailout amount at v then there are c debts from $d_1, d_2, \ldots, d_k$ that are not paid on time; let us call these debts the 'bad' debts. Note that for each bad debt, there is a debt from u to v that *might* have been paid at a time early enough to cover the debt but wasn't. Let $e_1, e_2, \ldots, e_c$ be distinct debts from D_u that might have been paid earlier so as to enable the payment of the bad debts. Amend the bailout B so that the $\in\!\!c$ formerly given as bailout to v is now given to u and denote this revised bailout by B'. Revise the schedule σ so that for each $1 \le i \le c$, $\in\!\!1$ of bailout at u is used to pay the debt e_i at the earliest time possible. Doing so results in us being able to pay all bad debts on time; hence, we have a perfect schedule for $(G, D, A^0 + B')$ where v receives no bailout. This yields a contradiction and so if there is a bailout B and a valid perfect schedule for $(G, D, A^0 + B)$ then there is a bailout B' and a valid perfect schedule for $(G, D, A^0 + B')$ where v receives no bailout funds. We shall return to this comment in a moment.

From the debts of D_u, we choose a debt $d_u = 1@[t_1, t_2]$, where $t_1 \le t' \le t_2$, so that from amongst all of the debts of D_u of the form $1@[t_3, t_4]$, where $t_3 \le t' \le t_4$, we have that $t_2 \le t_4$; that is, from all of the debts of D_u that 'straddle' t', d_u is a debt whose right-most time-stamp is smallest. We amend G by: introducing a new node v' and a new debt d' from u to v' of the form $1@t'$; removing the debt d_u from u to v; and removing the debt d_v from v to w. Denote this revised IDM instance by (G', D', A^0); it can be visualized as in Fig. 18(e).

Suppose that σ is a valid perfect schedule for $(G, D, A^0 + B)$, for some B. From above, we may assume that there is no bailout to node v in B. Consider the payment by v to w of the debt d_v. If the actual $\in\!\!1$ that pays this debt came from the payment of a debt from $D_u \setminus \{d_u\}$ of the form $1@[t_3, t_4]$ (where $t_3 \le t' \le t_4$), then we can amend σ so that we use this $\in\!\!1$ to pay the debt d_u at the time t' and use the $\in\!\!1$ that paid the debt d_u to pay the debt $1@[t_3, t_4]$ (at whatever time d_u was paid); that is, we swap the times of the payment of the debts d_u and $1@[t_3, t_4]$ in σ except that we now pay d_u at time t'. If the actual $\in\!\!1$ that pays d_v came from the payment of d_u then we can amend the payment time of d_u to t' (if necessary).

Build a schedule σ' in $(G', D', A^0 + B)$ from σ by: instead of paying d_u (at time t'), we pay the new debt d' from u to v'; and we remove the payment of the debt d_v. The schedule σ' is clearly a valid perfect schedule of $(G', D', A^0 + B)$. Conversely, if σ' is a valid perfect schedule of $(G', D', A^0 + B)$, we can build a schedule σ for $(G, D, A^0 + B)$ from σ' by: instead of paying the debt d' (at

time t'), we pay the debt d_u at time t'; and we use this €1 to pay immediately the debt d_u. The schedule σ is clearly a valid perfect schedule of $(G, D, A^0 + B)$. Hence, $((G, D, A^0), b)$ and $((G', D', A^0), b)$ are equivalent and we can work with $((G', D', A^0), b)$.

Case (c): Suppose that there is no debt from u to v of the form $1@[t_1, t_2]$ where $t_1 \leq t'$. This case cannot happen as we have ensured that no node of G is insolvent.

By iteratively applying all of the amendments to the instance $((G, D, A^0), b)$, as laid out above, we reduce $((G, D, A^0), b)$ to an equivalent instance $((G', D', (A')^0), b')$ where G consists of a solitary directed edge and all debts have a singular time-stamp. The process of reduction can clearly by undertaken in time polynomial in the size of the initial instance $((G, D, A^0), b)$ and the resulting instance $((G', D', (A')^0), b')$ can clearly be solved in time polynomial in the size of the initial instance $((G, D, A^0), b)$. Hence, BAILOUT MINIMIZATION is solvable in polynomial-time. □

Our polynomial-time algorithm for BAILOUT MINIMIZATION, and so PERFECT SCHEDULING, in the PP variant in Theorem 11 when we restrict to out-trees contrasts with the NP-completeness of PERFECT SCHEDULING when we restrict to directed acyclic graphs or multiditrees, as proven in Theorem 3 and Theorem 4, respectively.

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.

An interesting result of ours is the (weak) NP-completeness of BANKRUPTCY MINIMIZATION on a fixed, 32-node footprint graph (with edge multiplicity unbounded) in Theorem 2. It is noteworthy that constantly many nodes suffice to express the complexity of any problem in NP, and this leads to several open questions. Does the same hold when integers must be encoded in unary? We know this is true for the AoN case (as shown in Theorem 6). What is the smallest number n such a family of n-node (FP/PP) BANKRUPTCY MINIMIZATION instances is NP-complete? From the other side, what is the largest number n such that any n-node (FP/PP) BANKRUPTCY MINIMIZATION instance may be solved in polynomial time, and with what techniques?

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 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? What hardness or tractability results hold for variants in which the objective is, instead of the number of bankruptcies, the total amount of unpaid debt (or any other objective, for that matter)?

CRediT authorship contribution statement

Tom Friedetzky: Supervision. **David C. Kutner:** Writing – original draft. **George B. Mertzios:** Supervision. **Iain A. Stewart:** Supervision. **Amitabh Trehan:** Supervision.

Funding

This work was partially supported by Engineering and Physical Sciences Research Council grant EP/P020372/1.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

References

[1] L. Eisenberg, T.H. Noe, Systemic risk in financial systems, Manag. Sci. 47 (2) (2001) 236–249.
[2] L.C. Rogers, L.A. Veraart, Failure and rescue in an interbank network, Manag. Sci. 59 (4) (2013) 882–898.
[3] S. Schuldenzucker, S. Seuken, S. Battiston, Finding clearing payments in financial networks with credit default swaps is PPAD-complete, in: 8th Innovations in Theoretical Computer Science Conference, ITCS, in: LIPIcs, vol. 67, 2017.
[4] P.A. Papp, R. Wattenhofer, Sequential defaulting in financial networks, in: 12th Innovations in Theoretical Computer Science Conference, ITCS, in: LIPIcs, vol. 185, 2021.
[5] N. Bertschinger, M. Hoefer, D. Schmand, Flow allocation games, Math. Oper. Res. (2024).
[6] P.A. Papp, R. Wattenhofer, Network-aware strategies in financial systems, in: ICALP 2020, vol. 168, 2020.
[7] B. Egressy, R. Wattenhofer, Bailouts in financial networks, CoRR, arXiv:2106.12315, 2021.
[8] P. Kanellopoulos, M. Kyropoulou, H. Zhou, Forgiving debt in financial network games, in: IJCAI-22, 2022.
[9] P. Kanellopoulos, M. Kyropoulou, H. Zhou, On priority-proportional payments in financial networks, Theor. Comput. Sci. 1014 (2024) 114767, https://doi.org/10.1016/j.tcs.2024.114767.
[10] A.G. Haldane, R.M. May, Systemic risk in banking ecosystems, Nature 469 (7330) (2011) 351–355.
[11] M. Bardoscia, P. Barucca, S. Battiston, F. Caccioli, G. Cimini, D. Garlaschelli, F. Saracco, T. Squartini, G. Caldarelli, The physics of financial networks, Nat. Rev. Phys. 3 (7) (2021) 490–507.
[12] L. Eisenberg, A summary: Boolean networks applied to systemic risk, in: Neural Networks in Financial Engineering, 1996, pp. 436–449.
[13] J.-C. Rochet, X. Vives, Coordination failures and the lender of last resort: was Bagehot right after all?, J. Eur. Econ. Assoc. 2 (6) (2004) 1116–1147.
[14] W. Bagehot, Lombard Street: a Description of the Money Market, HS King & Company, London, 1873.
[15] M. Papachristou, J. Kleinberg, Allocating stimulus checks in times of crisis, in: Proceedings of the ACM Web Conference 2022, 2022, pp. 16–26.
[16] B. Tesfaye, N. Augsten, M. Pawlik, M. Böhlen, C. Jensen, Speeding up reachability queries in public transport networks using graph partitioning, Inf. Syst. Front. 24 (2022) 11–29.
[17] J. Enright, K. Meeks, G.B. Mertzios, V. Zamaraev, Deleting edges to restrict the size of an epidemic in temporal networks, J. Comput. Syst. Sci. 119 (2021) 60–77.
[18] D.C. Kutner, L. Larios-Jones, Temporal reachability dominating sets: contagion in temporal graphs, in: K. Georgiou, E. Kranakis (Eds.), Algorithmics of Wireless Networks, Springer, Cham, 2023, pp. 101–116.
[19] P. Holme, J. Saramäki (Eds.), Temporal Networks, Springer, London, 2013.
[20] O. Michail, An introduction to temporal graphs: an algorithmic perspective, Internet Math. 12 (4) (2016) 239–280.
[21] D. Kempe, J.M. Kleinberg, A. Kumar, Connectivity and inference problems for temporal networks, in: Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC), 2000, pp. 504–513.
[22] N. Klobas, G.B. Mertzios, H. Molter, R. Niedermeier, P. Zschoche, Interference-free walks in time: temporally disjoint paths, in: Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI), 2021, pp. 4090–4096.
[23] E.C. Akrida, G.B. Mertzios, P.G. Spirakis, V. Zamaraev, Temporal vertex cover with a sliding time window, J. Comput. Syst. Sci. 107 (2020) 108–123.
[24] Audience participation at the Cambridge LIPNE workshop on computational complexity and economic decision making, 2024.
[25] C.A. Tovey, A simplified NP-complete satisfiability problem, Discrete Appl. Math. 8 (1984) 85–89.
[26] R.M. Karp, Reducibility among combinatorial problems, in: R.E. Miller, J.W. Thatcher (Eds.), Complexity of Computer Computations, in: The IBM Research Symposia Series, Plenum Press, New York, New York, 1972, pp. 85–103.
[27] M.R. Garey, D.S. Johnson (Eds.), Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman and Company, United States, 1979.