Integral B2B Debt Netting: Model and Static Algorithm

Joannès Guichon, Sylvain Contassot-Vivier, Nazim Fatès

▶ **To cite this version:**

Joannès Guichon, Sylvain Contassot-Vivier, Nazim Fatès. Integral B2B Debt Netting: Model and Static Algorithm. 2025. hal-04947742

HAL Id: hal-04947742
https://hal.science/hal-04947742v1

Preprint submitted on 14 Feb 2025

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution - NonCommercial - NoDerivatives 4.0 International License

Integral B2B Debt Netting: Model and Static Algorithm

Joannès Guichon[1], Sylvain Contassot-Vivier[1], and Nazim Fatès[1]

Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France
joannes.guichon@inria.fr

1 Introduction

In many economies, trade credit—in the form of deferred payments to suppliers—serves as a critical driver for short-term financing. However, the prevalence of late invoice payments often leads to interlocking debts: a company can be simultaneously a creditor to some businesses and a debtor to others. Over time, these unpaid invoices accumulate into a network of obligations.

Such a networked build-up poses significant liquidity risks. If some companies cannot collect what is owed to them, they may be unable to fulfil their own payment obligations, triggering a domino effect of financial stress or defaults [4].

To mitigate these risks, netting procedures (i.e., mutual debt compensations) have been explored, particularly in the banking sector [6]. Notably, some existing work focuses on *partial* netting, wherein invoices are split to settle only part of the debt [3], or on more complex methods involving external liquidity such as *integral* netting [2,8]. Our objective here is *integral* netting in a global B2B context : we aim to cancel entire invoices fully by using an external financer who injects liquidity at strategically chosen points in the B2B network that is a large economic system.

The integral debt reduction approach simplifies accounting operations. Indeed, compared to partial netting where the amount of the invoices can be reduced, here invoices are either left untouched or are completely settled and removed. Yet, finding an optimal selection of invoices to reduce is NP-complete [2], requiring efficient heuristics for large-scale networks.

In this paper, we concentrate on the purely *static* version of the integral netting procedure, that is, we focus on a single state of the network of all unpaid invoices (i.e. debts) at a given time. Our main contributions are:

- a formal representation of debts as a weighted directed multi-graph;
- a more complete set of metrics (amplification factor, settlement inclusion, gain, etc.) to evaluate the efficiency of the netting procedure;
- a new algorithm for integral debt reduction, relying on an external financer whose capital is deployed to trigger chains of debt settlement;
- illustrative experiments on small synthetic data to compare with optimal solutions;
- results on larger graphs extracted from real-world data.

This work is a first step before working on more evolved frameworks, for example in the case where invoices dynamically arrive and expire.

The rest of the paper is organized as follows. In Section 2, we define the multi-directed graph structure to represent debts. Section 3 provides our main metrics for assessing netting quality. In Section 4, we present our static netting algorithm. Then, we show experimental results on small graphs in Section 5, including a comparison with the optimal solutions. Finally, experimental results on large graphs extracted from real data are provided in Section 6.

2 Model of B2B debt as a multi-directed graph

We represent invoices between companies using a weighted directed *multi*-graph, allowing multiple edges from one company to another.

Definition 1 (B2B Invoice Graph). *Let V represent a set of companies exchanging invoices in a specific context. A B2B invoice graph is a weighted directed multi-graph $G = (V, E)$, where each edge $e \in E$ is a tuple (s, d, w):*

- *$s \in V$ is the debtor (source),*
- *$d \in V$ is the creditor (destination),*
- *$w > 0$ is the amount owed from s to d.*

Since multiple invoices between the same pair of companies can occur, G may contain multiple edges with the same parameters. For each edge $e \in E$, we define the functions $w(e), s(e)$, and $d(e)$ associated with each element of e.

2.1 Internal metrics of an invoice graph

An invoice graph G can be measured by different metrics that are defined specifically for it and are useful later for the integral reduction process.

We denote by D_G the *total amount of debt* in a graph G. Formally, D_G is defined as the sum of the weights of all edges in the graph G:

$$D_G = \sum_{e \in E} w(e). \tag{1}$$

Next, we define the *financing* required to reduce all the debt in the network represented by G. This financing is defined as the sum of the negative net positions of its nodes, where the net position of a node v is the difference between the total incoming edge weights and the total outgoing edge weights.

For a node v, we define in graph G the sets of outgoing edges $O_G(v)$ and incoming edges $I_G(v)$ as follows:

$$O_G(v) = \{e \in E \mid s(e) = v\}, \qquad I_G(v) = \{e \in E \mid d(e) = v\}, \tag{2}$$

The *net position of node v* (or the *balance* of v) in graph G is given by:

$$\mathrm{B}_G(v) = \sum_{i \in I_G(v)} w(i) - \sum_{o \in O_G(v)} w(o). \tag{3}$$

The *minimal external financing* required to eliminate all debts in G is the positive sum of negative net positions across all nodes:

$$\phi_G = \sum_{v \in V} \max\left(0, -\mathrm{B}_G(v)\right). \tag{4}$$

3 Evaluation metrics for integral debt netting

In this section, we introduce the metrics that are used in our reduction process to choose the reduced edges. Some of these metrics are also used to evaluate the global quality of the reduction process.

The evaluation of a given reduction is crucial inside the reduction algorithm in order to choose the best reduction candidates, and also for comparing the quality of the results between different approaches. To facilitate this evaluation, we define a subgraph R of the initial network $G = (V, E)$, where $R = (V, S)$ corresponds to the selection of edges $S \subseteq E$ to be integrally settled. We define several metrics to evaluate the effectiveness of selecting R. D_R and ϕ_R are primary metrics for our reduction process, in particular ϕ_R since it gives us the total amount required to perform the reduction.

The efficiency of the reduction algorithm is evaluated using two main parameters:

- the *amplification factor* captures how much total debt is cleared per unit of financing:
$$\alpha(R) \;=\; \frac{D_R}{\phi_R}. \tag{5}$$
- the *settlement inclusion* factor measures the ratio of the debt cleared over the total weight:
$$I_G(R) \;=\; \frac{D_R}{D_G}. \tag{6}$$

The higher $\alpha(R)$ is, the higher the efficiency of using an external funding for the debt reduction process. However maximizing α alone is not sufficient as we also want to maximize the quantity of debt cleared. Also, using the inclusion factor alone presents some limitations as, at some point, it degrades the amplification factor. So, the netting problem can be seen as a multi-criteria optimization where a good trade-off must be obtained between those two metrics.

For instance, the reduction process may arrive in situations where the remaining possibilities consist in financing low-weight invoices with no significant subsequent reduction. Such small invoices slightly enhance the inclusion factor but also decrease the amplification factor. Indeed, while one might argue that financing low-weight invoices has a negligible cost due to their small influence, our primary objective is to facilitate netting between companies. Therefore, it is undesirable to include invoices that do not contribute meaningfully to the overall reduction process.

To propose a more relevant metrics than the inclusion, we define a *gain* measure that is the netted "surplus" over the maximum netted surplus:

$$g_G(R) \;=\; \frac{D_R - \phi_R}{D_G - \phi_G}. \tag{7}$$

This measure reflects the proportion of netted debt (i.e. the surplus of debt eliminated beyond the financing itself) captured in R. It can be shown that the gain measure is a normalized metric (see Appendix for a short proof). A value of 1 means that R captures *all* the nettable debt in G, while 0 means none. It is worth noticing that a value of 1 does not imply that we have an optimal solution according to the amplification.

4 Static reduction algorithm

We now present an algorithm which acts on static graphs, or a *static algorithm* for short. Our goal is to find a subgraph R with the largest inclusion according to a given minimal amplification factor. This algorithm shares some similarities with a first approach that was developed to tackle the *integral* debt reduction problem [2], but its core components are fundamentally different.

4.1 Integral debt reduction

The key concept of this method is to introduce an external financer that allows for monetary contributions at certain points in the system to enable the integral reduction of some invoices. Our algorithm relies on the propagation of debt reduction through the network and on the creation of subsequent debts to the financer.

4.2 Needed measurement

We define the *potential of an edge* as the total amount of debt that can be cleared when injecting in the graph exactly the value of the considered invoice. Formally, we define the *outward potential* of the edge e as :

$$U_G^-(e) \;=\; w(e) \;+\; \max_{\substack{L \subseteq O_G(d(e)) \\ \sum_{o \in L} w(o) \leq w(e)}} \sum_{o \in L} U_G^-(o), \tag{8}$$

where :

- $w(e)$ is the weight (debt) of edge e;
- $O_G(d(e))$ is the set of all edges leaving the destination node $d(e)$;
- L is any subset of $O_G(d(e))$ whose total weight does not exceed $w(e)$.

Intuitive explanation : If the financer pays edge e, then the destination node $d(e)$ receives the amount $w(e)$ and can pay some of its own debts with this amount. Then we aim to chose a combo of edges that pays the most invoices possible in total using only the money coming from edge e. In addition, those edges are chosen in order to maximize the sum of their respective potentials.

Computing the exact value of $U_G^-(e)$ is computationally intensive due to the combinatorial nature of the maximization (exhaustive search). In practice, depending on the problem size, it is either computed using dynamic programming or approximated with a greedy method.

We extend this notion of outward potential to the *inward potential* $U_G^+(e)$ of an edge, by replacing the set $O_G(d(e))$ by the set $I_G(s(e))$ in Eq. 8. It corresponds to a set of upstream edges converging towards e, whose total financing does not exceed $w(e)$. This allows us to redirect the initial financing of e to some of its upstream edges. We respectively name L_e^- the subset of edges that maximizes $U_G^-(e)$ and L_e^+ the subset of edges that maximizes $U_G^+(e)$.

If we consider a path $p = (e_1, e_2, \ldots, e_k) \subset E$, a sequence of edges of $G = (V, E)$, where $k > 1$ and $\forall i \in \{1, \ldots, k-1\}, d(e_i) = s(e_{i+1})$, we define the *potential of a path* as the total debt reducible associated with the edges in the path p:

$$U_p = \sum_{e \in p} w(e). \tag{9}$$

4.3 Overview of the algorithm

Our reduction algorithm can now be defined as follows:

1. **Identify trivial 2-cycles**: If G has pairs of edges $e_1 = (s, d, w)$ and $e_2 = (d, s, w)$, both can be cancelled outright with zero external financing. We directly remove these edges from G and do not include them in our reduction process. Indeed, such a bilateral settlement can be realised directly by the companies themselves.
2. **Search for "germs"**: These germs are short paths in the graph with high potential-to-cost ratio. Each path p is financed by its first (highest-weight) edge only, and subsequent edges become self-financing if they have lesser weights. We build a set of non-overlapping germs by iteratively choosing candidates with maximal value of ratio $\rho(p) = U_p/\phi_p$ over a given threshold. Those selected paths form the base of the reduction graph R[1]. It is worth mentioning that perfect cycles of length greater than two are found in this step, inducing $\rho(p) = \infty$ for such path p.
3. **Extend the selected germs**: We iteratively add edges with high potential. Specifically, from each node that is part of a germ, we look for outward and inward edges that can attach if they present a potential-to-cost ratio above $\rho_{\min}$. We select the edge e with the highest $\rho(e)$, i.e. $\rho^+(e) = U_G^+(e)/\phi_{L_e^+}$ or $\rho^-(e) = U_G^-(e)/\phi_{L_e^-}$ (depending on which is higher) and the associated edges in L_e^+ or L_e^- depending on the direction with highest potential-to-cost-ratio. This extension process is iterative and continues until no further edges

[1] Another method for finding germs consists in focusing on *interesting* cycles [1]. However, this method is quite demanding in terms of computations and such cycles do not necessarily provide much leftover net positions to improve the expansion process.

can be added without violating $\rho_{\min}$. Doing this we aim to maximize $g_G(R)$ while controlling ϕ_R. This ensures that only edges contributing significantly are included in the reduced graph R (sub-graph of G).

4.4 External financer and resulting financing

Once R is fixed, the external financing ϕ_R (Eq. 4) must be injected into the network onto the negative-net position nodes to settle all edges in R. In practice, the financer is introduced as a special node v^*. For each node v with $B_R(v) < 0$, we add an edge $(v, v^*, -B_R(v))$ denoting the debt of v toward the financer. This money is then used by v to pay the edges in R for which it is the debtor.

5 Experimental results on small synthetic graphs

As a first experiment, we work on small random synthetic graphs of 20 edges. Indeed, for such small size, exact optimum can be found for our metrics, allowing us to evaluate the capacity of our process to find optimal solutions.

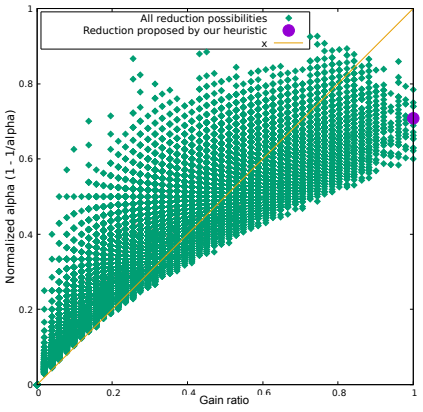

Fig. 1. Result of our algorithm on a random graph with 20 edges in terms of normalized alpha and gain compared to all possible reductions

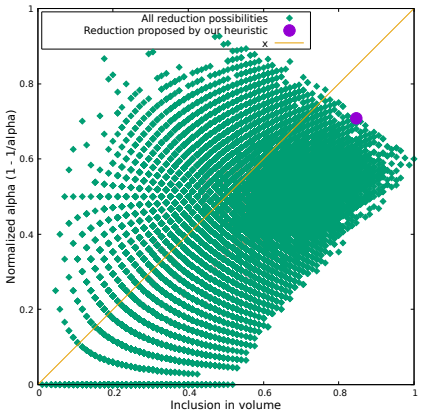

Fig. 2. Result of our algorithm on a random graph with 20 edges in terms of normalized alpha and volume compared to all possible reductions

Figure 1 displays the normalized amplification as a function of the gain for a typical example of graph. In this plot, the best solutions are those which are the closer to the upper-right corner (100% normalized amplification and gain), assuming an equal weight granted for each parameter. As there exists trade-off between these parameters, the optimal points form a kind of Pareto distribution. Our algorithm picks a sub-graph that achieves the best gain possible. However, the obtained amplification does not coincide with the best amplification for that

gain. Indeed, our method can sometimes require injecting slightly more financing since we work with $\rho^+(e)$ or $\rho^-(e)$, while the best solution may be a more complex construction.

Similarly, if we look at settlement inclusion $I_G(R)$ vs. amplification $\alpha(R)$ in Figure 2, our algorithm is on the Pareto frontier, meaning that for a given inclusion, the amplification is maximized. This induces that the algorithm presents a good trade-off between the *graph coverage* and the *efficiency per euro financed*.

For larger graphs, the Pareto frontier cannot be found in reasonable time as the problem is NP-complete [2]. Hence, it is likely that our algorithm will be suboptimal. Despite this limitation, we observed that the results obtained on large graphs are of a rather good quality, as we examine in the next section.

6 Algorithm for static graphs on a real-world dataset

6.1 B2B invoice graphs debt data

We work on a set of 22 million invoices above 200 euros, emitted between 2 million companies from Italy over the span of the 2019 year, for a total amount of 89 billion euros. This dataset is provided by InfoCert, an Italian electronic invoices operator and was explored in details in a previous article [7]. Each invoice is composed of the following data:

- unique identifier to register the invoice;
- unique identifier of the debtor company;
- unique identifier of the creditor company;
- due debt in euros;
- date of the invoice emission.

This data is partial since it presents only B2B invoices given to InfoCert and not whole set of the existing invoices. The weight limitation also adds to this partiality, however the breadth and depth of this dataset are adequate to validate our approach. In this context, breadth refers to the extent of edges distribution within the network, indicating that the network is not overly concentrated on a small subset of nodes, but rather well spread across various entities; and *depth* relates to the capacity of the dataset to capture long paths of invoices.

We observe that when we consider the complete year of data, there is a concentration of net positions around zero, which confirms a phenomenon already remarked by Fleischman et al. [5]. However, our network presents much more imbalances, with some companies having large negative or positive net positions, as highlighted in Figure 3. Contrary to the data used by other authors [5], our data has a lower percentage of overlapping cyclic structures, meaning that we cannot apply their techniques to have a fair comparison. Those differences probably come from the fact that both data sets are subparts of the complete economic networks whose selection is based on different criteria.

Indeed, as shown in a previous work, our economic dataset presents log-normal and scale-free distributions for the amounts and the node degrees respectively [7]. A bias towards an in-degree higher than the out-degree is observed and

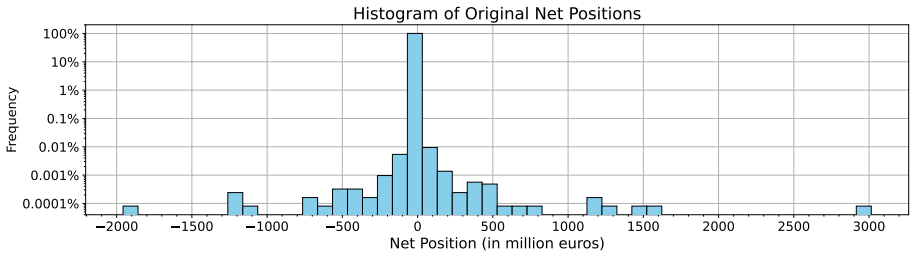

Fig. 3. Repartition of the net position on the overall year for companies in our data

we used it to our advantage in Section 4 when looking for germs, diminishing the computational need by working with outgoing paths. Furthermore, we observe patterns over the volume and number of invoices inside weeks and months [7]. Typically, the last day of each month concentrates a large percentage of the total exchanges within a month. Such days should allow for more reduction as they contain much more invoices than other days, and the more data, the more flexible and efficient the reduction can be.

6.2 Analysis of the results

Measuring the effect of reducing the debt of a network is mandatory in order to assess the usefulness of our algorithm. We applied our reduction algorithm to daily data to get a first insight about the capacity of our algorithm to reduce debts in actual data.

It is interesting to compare graphs of different sizes and see how the reduction algorithm applies on them. Figure 4 presents our algorithm's results through the iterations of the reduction process on three different days of representative sizes. Day 26 is a small day in terms of edges and volume, day 25 is a mid one and day 212 is one of the 12 largest one as it is the end of July. The graphs do not imply the same amount of iterations since they don't have the same size and are also dependant on their germs. For all three graphs, the amplification factor stays above 2 through the iterations of the process and the final gain is between 65% and 72%. An inclusion factor of at least 20% is also obtained. It can be observed that when restricting the addition of edges with an amplification factor above $\rho_{min} = 1.95$, all the reducible debt inside the graphs is not cleared. It is important to recall that clearing all the reducible debt is not our goal as it leads generally to very low amplification factors (close to 1).

An important observation is that larger days generally yield better results than small ones. Nevertheless, the structure of the graph plays a role too, and the best results are not systematically obtained with the largest graphs.

Moreover, the computation time of largest graphs is significantly higher ($\times 20$) than middle sized ones. This implies that accumulating invoices from a large span of time (week, month,...) is not efficient in practice; which means that a good

strategy consists in processing small periods and applying a divide-and-conquer approach. This strategy is detailed in the next section.

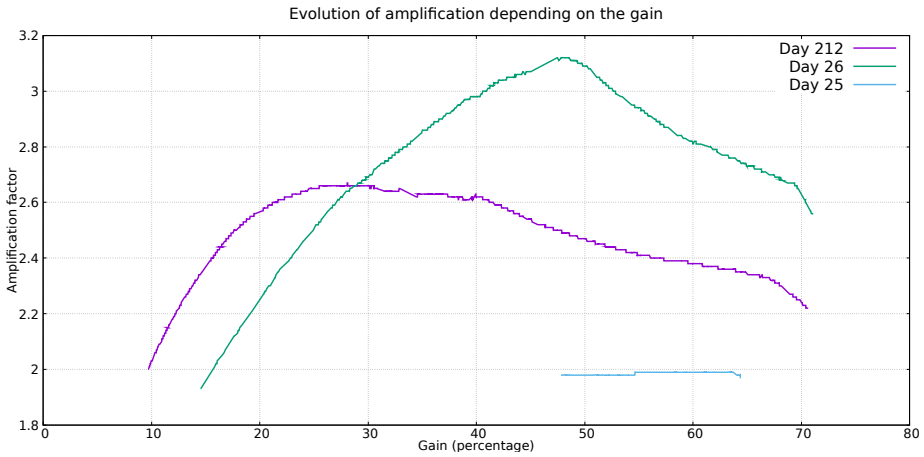

Fig. 4. Amplification factor as a function of the gain for three specific days

6.3 Graph partitioning inside the reduction process

As noticed before, an important issue when processing real data is the large size of some graphs that induces very large computation times. In order to speed up the calculation and to obtain acceptable times, we partition the initial graphs in two parts.

The partitioning used maximises the number of edges in each sub-graph. In order to preserve all the edges from the initial graph, the edges at the frontier between the two parts must be placed in one of them. We choose the part with the lowest overall degree to ensure an almost even repartition of the invoices. After reducing each part, we just merge them and recompute the financing. This is a minimalist approach that has merely no computational cost. However, such a strategy misses the additional reductions induced by merging both reduced parts. Finding all those possible reductions is quite expansive. However, some of them, namely the ones that do not require any additional financing, are easier to detect with an iterative greedy algorithm applied on the nodes.

This strategy has been tested over a representative set of graphs and scored an average acceleration around 54. Figure 5 presents the results of our experiment and highlights how important the time saved is for large graphs. Figure 6 shows that the quality loss of this division strategy is around 17%, in terms of overall debt cleared. By adding the simple greedy mechanism discussed above, we keep a similar acceleration (52.5) while reducing the loss to only 12%.

Although these results are already quite good, we think that these partitioning and merging strategies deserve a deeper analysis for additional optimization and will be the subject of a future work.

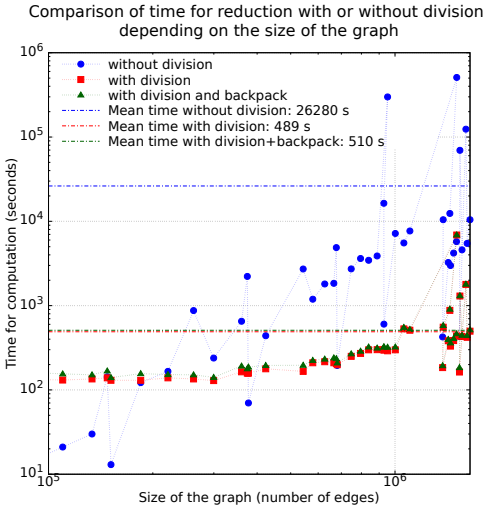

Fig. 5. Time comparison between division and no division depending on the size of the graph

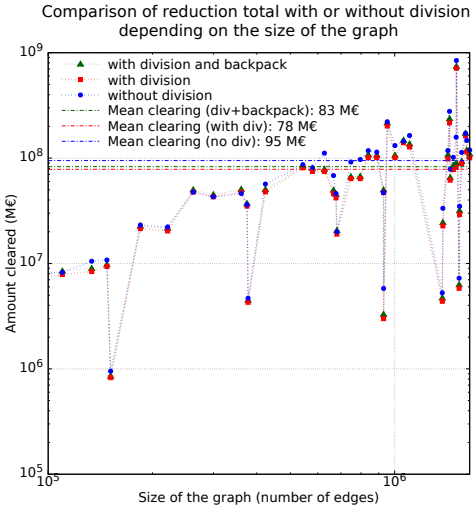

Fig. 6. Debt clearing comparison between division and no division depending on the size of the graph

7 Conclusion

In this work, a static model of B2B invoice networks was presented together with an algorithm for integral debt netting. The presence of an external financer that injects some capital at strategic points in the network allows the complete settlement of a great number of invoices, simplifying accounting processes and lowering the need for liquidity of companies.

Through our proposed methodology, we defined a multi-directed graph representation of invoice obligations and introduce or redefine key metrics, like the amplification factor and gain, to evaluate the efficiency of debt reduction algorithms. Our algorithm identifies high-impact invoices and sub-graphs, ensuring a good balance between debt clearance and minimal external financing. This allows for the settlement of high-impact debts.

Experimental evaluations on both synthetic and real-world datasets allowed us to estimate the performance of our method. On real invoice data, we achieve a significant amplification factor above 2.0 with a gain of at least 65%, and an inclusion of minimum 20% demonstrating the efficiency of our method. While our static model successfully mitigates debt in individual time frames, it also provides a foundation for future extensions incorporating dynamic and temporal aspects. For practical use, a first graph partitioning strategy was presented, which allows

a significant reduction of the computation time, at the cost of a slight quality reduction.

Our next step consists in extending this static framework to dynamic settings, where invoices are issued and settled continuously over time. By iteratively applying our reduction strategy across multiple periods, we aim to develop a liquidity-saving mechanism that optimally manages debt clearance. Other major goals consist in exploring other optimization heuristics to enhance the process of reductions, as well as other graph-partitioning strategies.

Acknowledgment

This work is supported by the Inria's Exploratory Action Murene.

References

1. Adriaens, F., Aslay, C., De Bie, T., Gionis, A., Lijffijt, J.: Discovering interesting cycles in directed graphs. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM '19, pp. 1191–1200. Association for Computing Machinery. https://doi.org/10.1145/3357384.3357970
2. Amato, M., Fatès, N., Gobbi, L.: The economics and algorithmics of an integral settlement procedure on b2b networks. SSRN Electronic Journal (2024). https://doi.org/10.2139/ssrn.3915380
3. Balinski, M.L., Gomory, R.E.: A primal method for minimal cost flows with applications to the assignment and transportation problems. Mathematics of Operations Research (2024). URL https://www.jstor.org/stable/2627433
4. Battiston, S., Delli Gatti, D., Gallegati, M., Greenwald, B., Stiglitz, J.E.: Credit chains and bankruptcy propagation in production networks **31**(6), 2061–2084. https://doi.org/10.1016/j.jedc.2007.01.004
5. Fleischman, T., Dini, P., Littera, G.: Liquidity-saving through obligation-clearing and mutual credit: An effective monetary innovation for SMEs in times of crisisl. Journal of Risk and Financial Management (2024). https://doi.org/10.3390/jrfm13120295
6. Gaffeo, E., Gobbi, L., Molinari, M.: The economics of netting in financial networks. Journal of Economic Dynamics and Control (2024). https://doi.org/10.1007/s11403-018-0229-4
7. Guichon, J., Fatès, N., Contassot-Vivier, S., Amato, M.: Properties of b2b invoice graphs and detection of structures. In: H. Cherifi, L.M. Rocha, C. Cherifi, M. Donduran (eds.) Complex Networks & Their Applications XII, pp. 444–455. Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-53472-0_37
8. Leclerc, M., Güntzer, M., Jungnickel, D.: Efficient algorithms for the clearing of interbank payments. European Journal of Operational Research (2024). https://doi.org/10.1016/S0377-2217(97)00265-8

Appendix: Proof that $g_G(R)$ is bounded by 1

The aim is to prove that $\forall\ G=(V,E)$ such that $D_G \neq F_G$ (meaning that some reduction by netting is possible), $\forall R=(V,S), S\subseteq E$:

$$\begin{aligned} & g_G(R) = \frac{D_R - F_R}{D_G - F_G} \leq 1 \\ \Leftrightarrow\ & D_R - F_R \leq D_G - F_G && (D_G \geq F_G \geq 0) \\ \Leftrightarrow\ & F_G - F_R \leq \underbrace{D_G - D_R}_{D_{G\setminus R}} && (\ D_G = D_{G\setminus R} + D_R) \end{aligned}$$

Note that $D_{G\setminus R} \geq F_{G\setminus R}$. If we prove that $F_{G\setminus R} \geq F_G - F_R$ then the proof is complete. We use the definitions of financings, we need to prove :

$$\sum_{v\in V} \max\left(0, -B_{G\setminus R}(v)\right) \geq \sum_{v\in V} \max\left(0, -B_G(v)\right) - \max\left(0, -B_R(v)\right)$$

We will prove that $\forall v \in V$:

$$\underbrace{\max\left(0, -B_{G\setminus R}(v)\right)}_{\mathcal{X}} \geq \underbrace{\max\left(0, -B_G(v)\right) - \max\left(0, -B_R(v)\right)}_{\mathcal{Y}}$$

Note that :

$$\begin{aligned} B_{G\setminus R}(v) &= \sum_{e\in I_G\setminus R(v)} w(e) - \sum_{e\in O_G\setminus R(v)} w(e) \\ &= \left(\sum_{e\in I_G(v)} w(e) - \sum_{e\in I_R(v)} w(e)\right) - \left(\sum_{e\in O_G(v)} w(e) - \sum_{e\in O_R(v)} w(e)\right) \\ &= \left(\sum_{e\in I_G(v)} w(e) - \sum_{e\in O_G(v)} w(e)\right) - \left(\sum_{e\in I_R(v)} w(e) - \sum_{e\in O_R(v)} w(e)\right) \\ &= B_G(v) - B_R(v) \end{aligned}$$

By case disjunction on $B_G(v)$ and $B_R(v)$:

Case $B_G(v) < 0, B_R(v) < 0$: $\quad \mathcal{Y} = -B_G(v) + B_R(v) = -B_{G\setminus R}(v) \leq \mathcal{X}$
Case $B_G(v) \geq 0, B_R(v) < 0$: $\quad \mathcal{Y} = B_R(v) \leq 0 \leq \mathcal{X}$
Case $B_G(v) \geq 0, B_R(v) \geq 0$: $\quad \mathcal{Y} = 0 \leq \mathcal{X}$
Case $B_G(v) < 0, B_R(v) \geq 0$:
$\mathcal{Y} = -B_G(v) \leq -B_G(v) + B_R(v) = -B_{G\setminus R}(v) \leq \mathcal{X} \quad (B_G(v) < 0 \leq B_R(v))$
$\Leftrightarrow \forall v \in V,$ and in every possible case of net positions, $\mathcal{Y} \leq \mathcal{X}$
□