
Multilateral Trade Credit Set-off (MTCS)

MIT, February 2025

Tomaž Fleischman
tomaz@informal.systems

Content

- Late Payment
- Obligation Network
- Formalising MTCS with MCMF
- MTCS Algorithm
- Empirical Setting
- Obligation Network Topology
- Liquidity Injection

The Problem of Late Payment

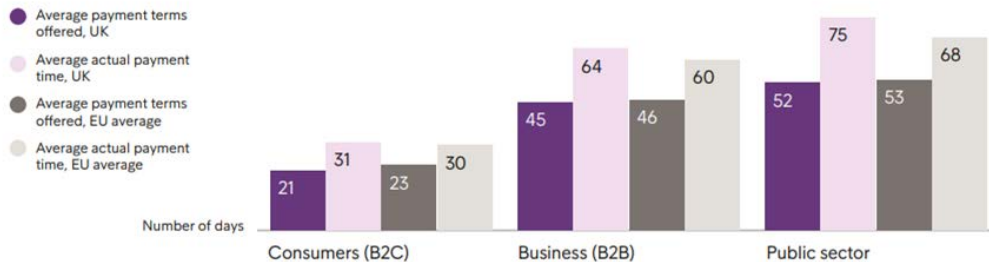
The basic impact of late payment

- It raises costs associated with financing working capital;
- It depletes cash reserves in businesses, while often losing them the interest which could have been accrued in the meantime;
- It escalates administrative costs associated with collections and recoveries;
- It drains labour productivity, and causes businesses to pass up further profitable work;
- It creates substantial distractions from everyday work;
- Despite healthy client registers, it creates losses for businesses which would otherwise seem profitable on paper;
- It often places the burden of financing the entire upstream supply chain on the smallest companies;
- It causes unemployment and bankruptcy;
- By killing otherwise profitable business ideas, late payment stifles competition and hampers business progress;
- It increases the barriers to entry for businesses in industries with the worst payment practices.

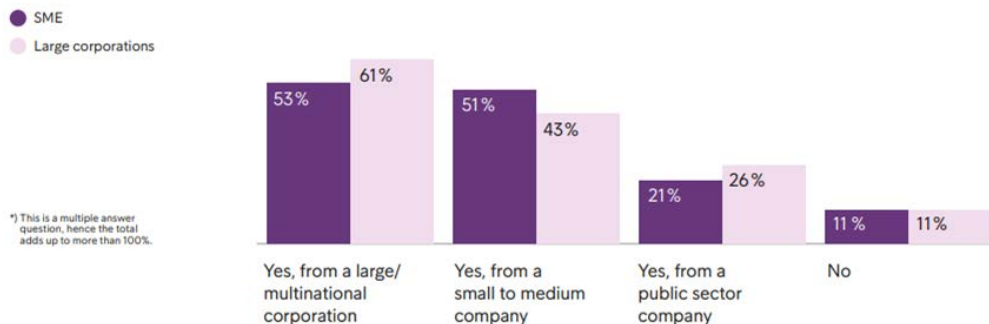
Late payment UK 2020

- Gap in payment terms and actual payment in UK is comparable but slightly larger than EU average
- Just 11% of companies in UK claim that they are comfortable with longer payment terms
- Late payment as a practice is rising sharply, increasing the risk of payment defaults.

Gap in payment terms offered and actual payment duration:



Have you accepted longer payments than you feel comfortable with over the past 12 months?*

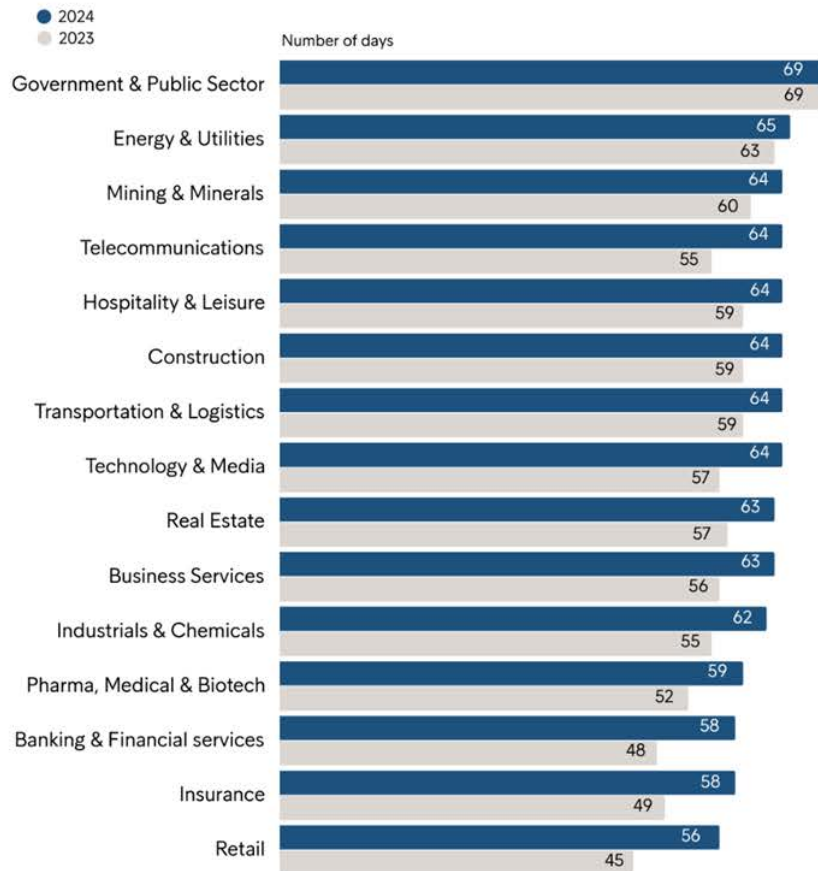


To what extent do you see risk of late/non-payments from your company's debtors developing during the next 12 months?

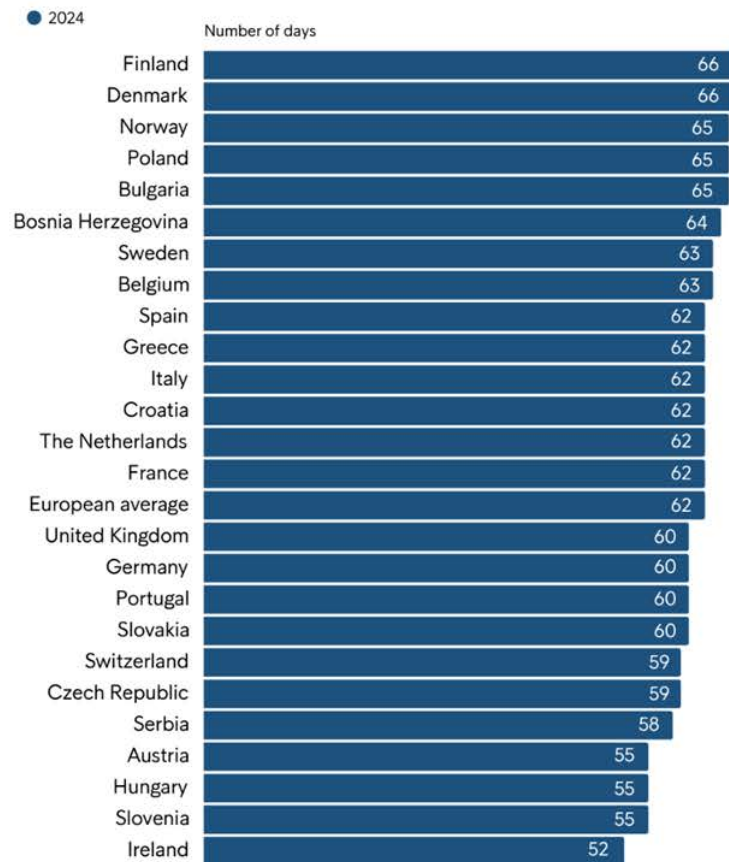


Source: Intrum Justitia - European Payment Report 2020

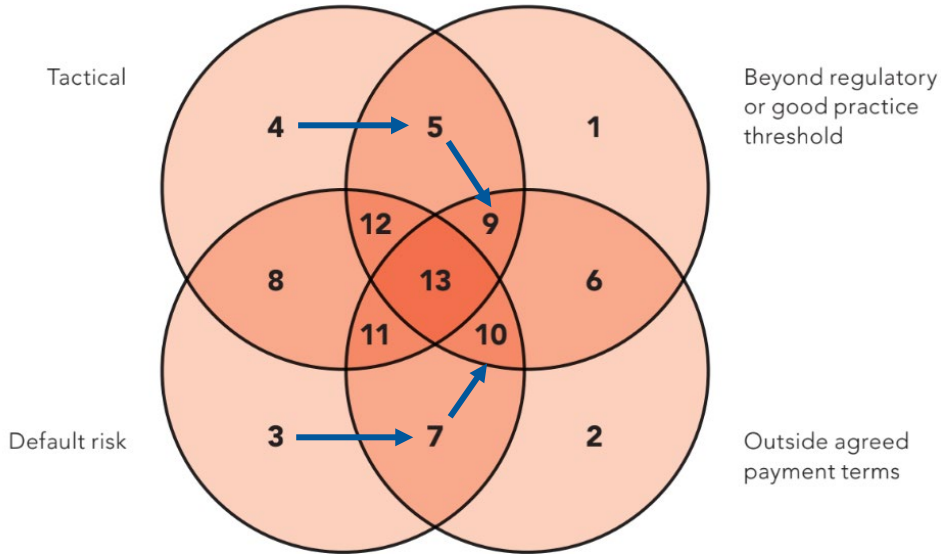
What is the average time taken by your corporate customers (B2B) in each of the following industries to make their payments?



Countries where corporate customers (B2B) on average take longest to make payments

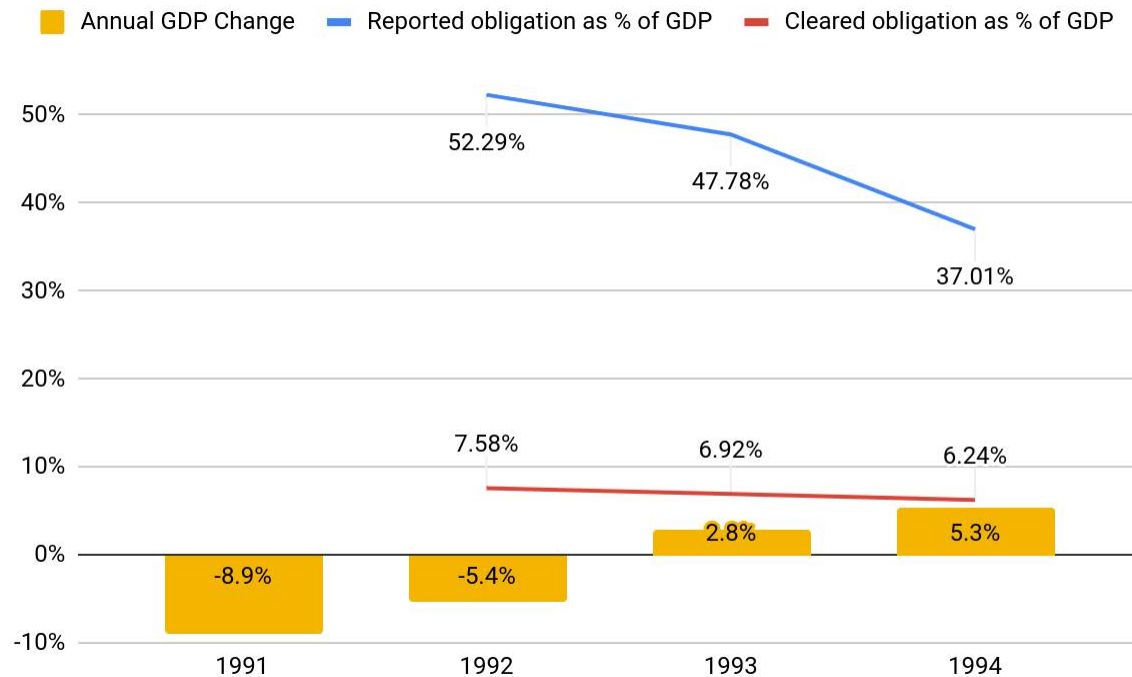


Defining late payment



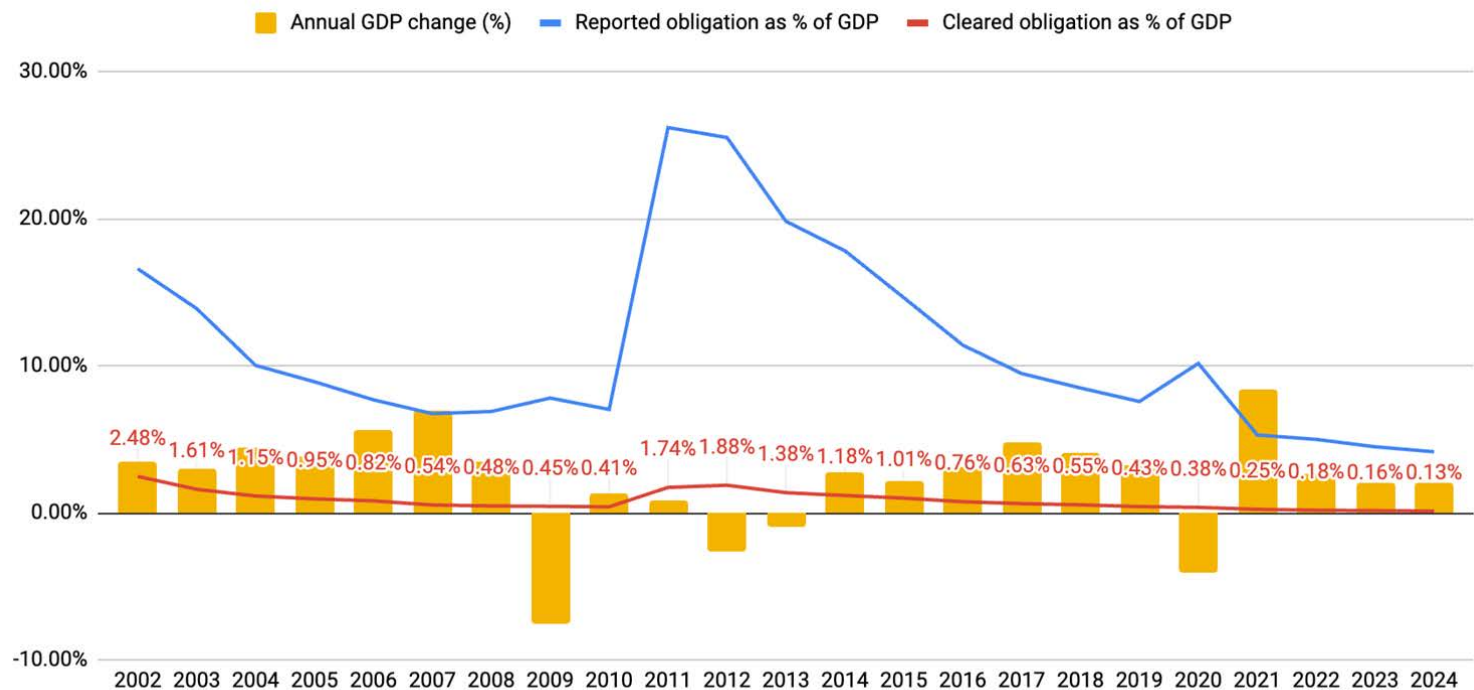
1. Industry standard credit terms that are long by the standards of other industries
2. Routine administrative delay or dispute
3. Low probability provision for bad debt
4. Routine de prioritisation of suppliers (no dilution)
5. Extended terms or prompt payment discounts demanded by a dominant buyer
6. Non routine administrative delay or dispute (with potential for legal recourse)
7. Short term forbearance/major invoice dispute
8. High probability provision for bad debt
9. Extended terms or prompt payment discounts demanded unilaterally by a dominant buyer; tactical invoice disputes (with potential for legal recourse)
10. Medium term forbearance/protracted major invoice dispute
11. Late payment with supplier dilution
12. Extended credit terms with potential supplier dilution (including provisions for bad debt and potential for legal recourse)
13. Buyer default in bad faith.

Source: Ending late payment. The Association of Chartered Certified Accountants, February 2015. © ACCA. All rights reserved. This content is excluded from our Creative Commons license. For more information, see <https://ocw.mit.edu/help/faq-fair-use/>.



Source: Official Gazette of the National Assembly of the Republic of Slovenia 1993 - 1995

Obligations cleared in Slovenia by multilateral set-off in the 1991–1994 liquidity crisis.



Source: https://www.ajpes.si/Bonitetne_storitve/Vecstranski_pobot/Porocila

Obligations cleared in Slovenia by multilateral set-off in the 2009–2012 financial crisis.

Late Payment Obligations in a Network, Directed Edges in a Graph

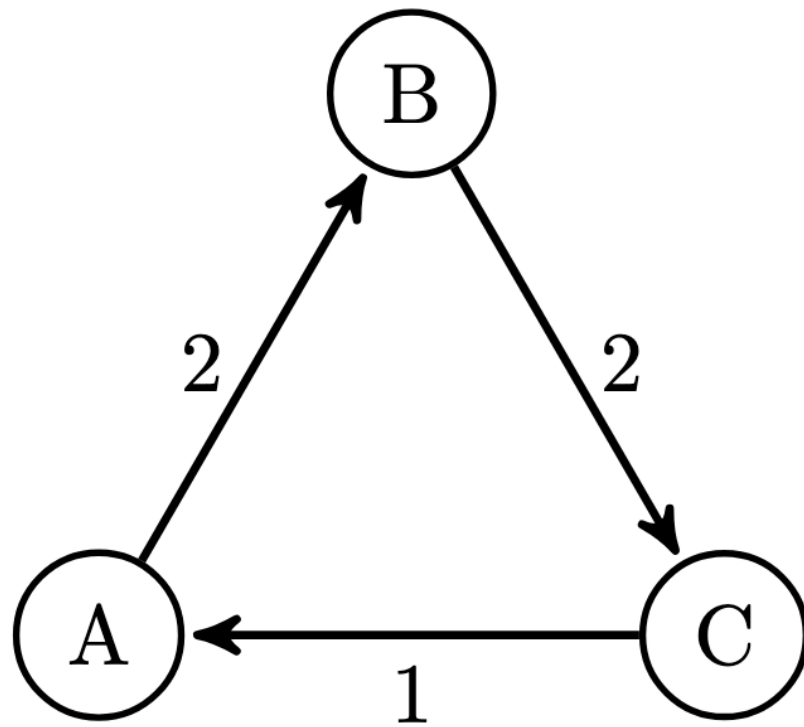
Trade Credit Network

For the basic example think of three firms that have accounts receivable from and payable to each other, as a result of business. An input supplier, as seller, not yet paid has an account receivable from the input purchaser, as buyer. This is a trade credit contract.

The initial situation among Alice, Bob, and Charlie can be described as follows:

- Alice, Bob, and Charlie are **nodes** of a graph.
- They have obligations represented as directed **edges** in a graph, which are overdue.
- Each edge is **weighted** with a non-negative value representing the amount owed and late.
- All amounts are in the same **unit of account**.

Detecting a cycle looks simple, but how do we do it in large networks!



Problem reformulation

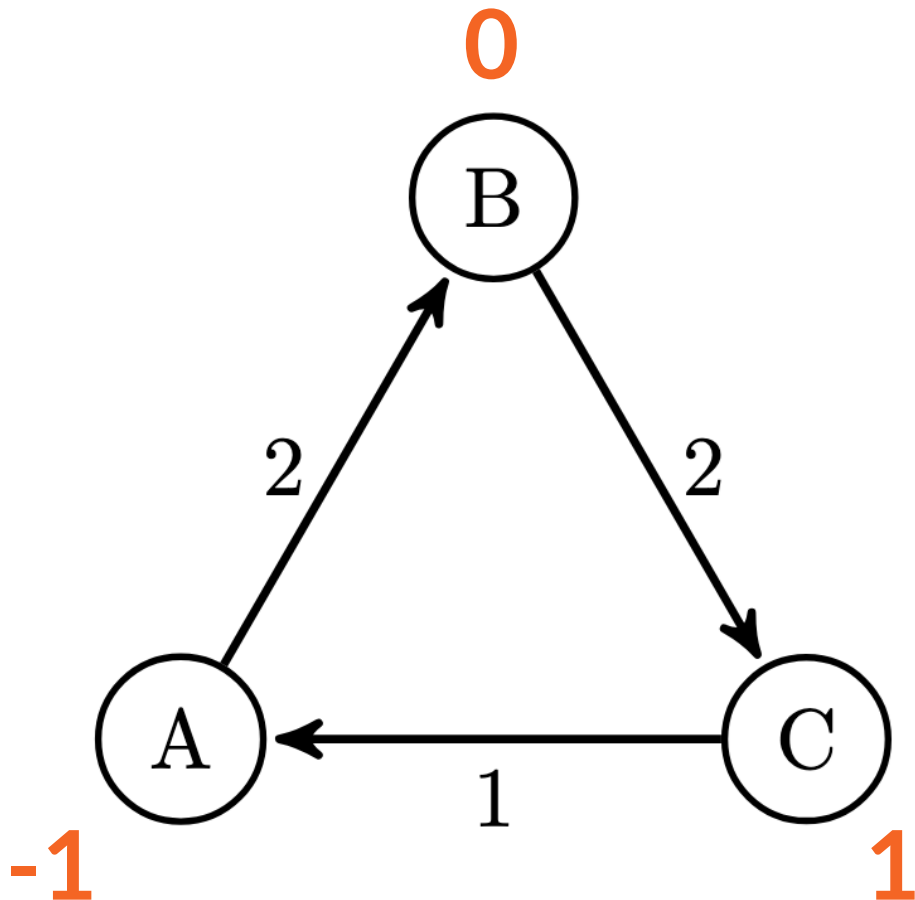
To address the complexity of the problem, we convert the issue of finding individual cycles in an obligations network into a flow problem.

the original obligations network typically lacks *balance*; this means individual node inflows and outflows do not match.

In our example, Alice has an inflow of 1 and an outflow of 2, giving her a balance of -1 due to insufficient inflow.

Conversely, Charlie enjoys a positive balance of 1, as his inflows surpass outflows.

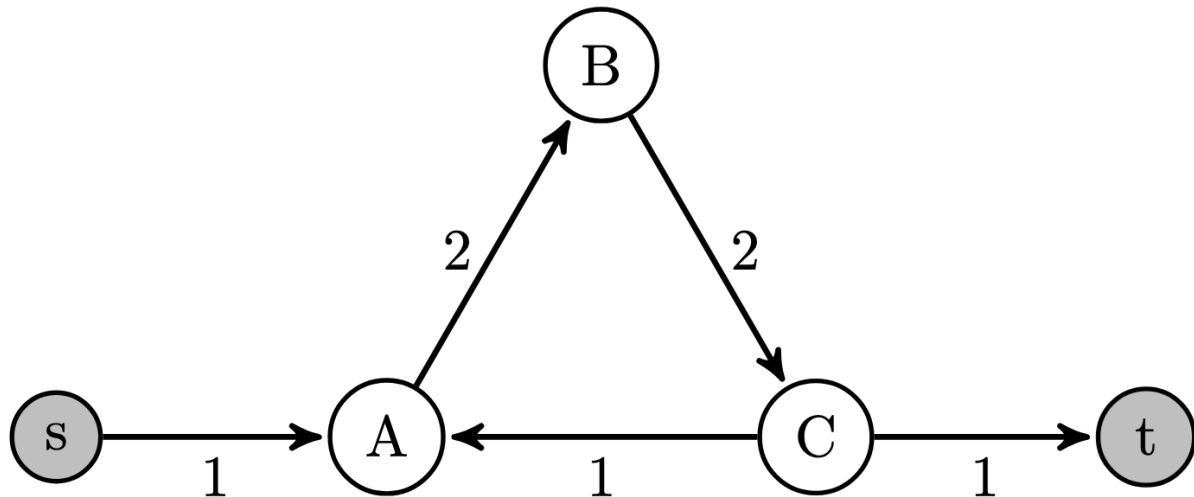
Bob's situation is neutral, as his inflows match his outflows.



Balancing with imaginary liquidity

To balance the obligations network, for each participant, we introduce two external nodes to the original obligations graph: a **source** (s) and a **sink** (t) or target. These nodes help balance an edge from the source to Alice with a weight of 1 and an edge from Charlie to the sink with a weight of 1.

This ensures all obligation network participants have inflows that equal their outflows.

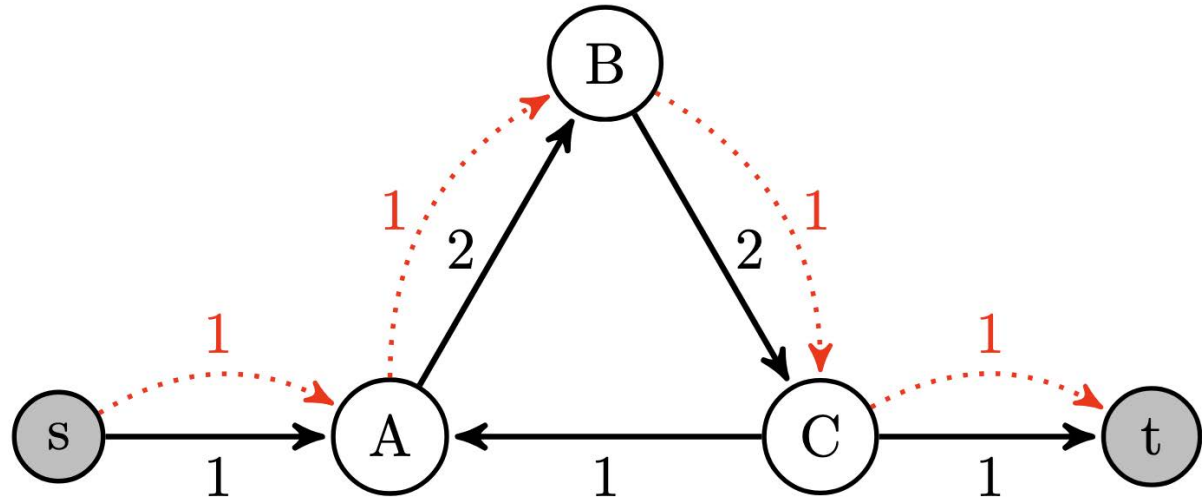


Saturating s-t flow

The key to the solution is reached by identifying a **saturated s - t flow** from the source (s) to the sink (t).

By construction, all flows originating from the source must equal those reaching the sink. Therefore, a **saturated flow** is also **the maximum flow** from source to sink that reflect the capacity constraints of indebtedness.

In this basic example, a saturated flow path from source to sink is (s), (A), (B), (C), (t) with a value of 1.



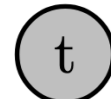
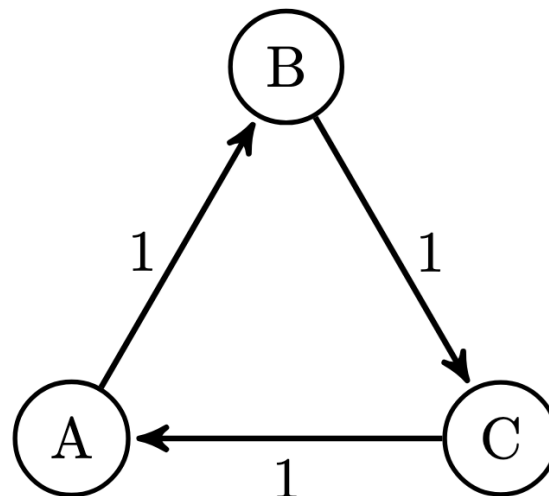
Finding a Cycle

How to find internal cycles with offset?

The solution to this cycle detection problem is obtained by *removing a saturated $s - t$ flow* from the *balanced network*.

In our simple example we have one cycle:

Alice, Bob, Charlie, Alice of 1.



Formalizing MTCS with Minimum Cost Maximum Flow (MCMF)

Introducing source and sink for MCMF

Initially, the obligation graph $G^0 = (V^0, E^0)$, where V^0 represents a set of firms and E^0 represents a set of payment obligations. The amount that firm u owes to firm v is shown as edge capacity c_{uv} . For every node i within V^0 in the original graph G^0 , we define its balance in the following way:

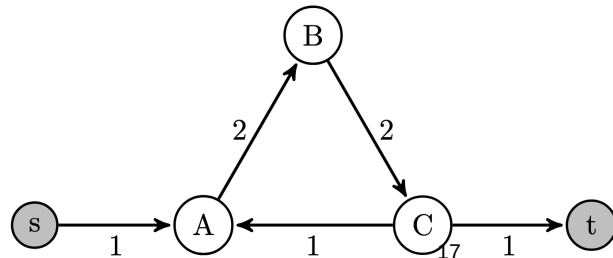
$$\forall i \in V^0 \quad b_i = \sum_{(u,i) \in E} c_{ui} - \sum_{(i,v) \in E} c_{iv}$$

We now add source node s and sink node t to form a balanced graph

$$\begin{aligned} G &= (V, E) \\ V &= V^0 \cup \{s, t\}, \\ E^0 &\subset E. \end{aligned}$$

To balance the graph, for each node except source s and sink t , an edge for balancing is added either from the source or to the sink.

$$\forall i \in V^0 : \begin{cases} (s, i) \text{ with capacity } c_{si} = -b_i, & \text{if } b_i < 0 \\ (i, t) \text{ with capacity } c_{it} = b_i, & \text{if } b_i > 0 \end{cases}$$



Balanced graph

- Capacity

$$c_{uv} \geq 0 \quad \forall (u, v) \in E$$

- Cost per unit flow

$$w_{uv} \quad \forall (u, v) \in E$$

- A source s and a sink t

$$s, t \in V$$

- Balanced for all nodes except source and sink

$$\sum_{v \in V} c_{uv} - \sum_{v \in V} c_{vu} = 0, \quad \forall u \neq s, t$$

Base case has cost of flow equal for all edges.

$$\forall (u, v) \in E, \quad w_{uv} = 1$$

Apply MCMF to find a saturating s-t flow

The minimum cost maximum flow function $f : E \rightarrow \mathbb{R}^+$ is defined as follows:

- Satisfies capacity constraints:

$$0 \leq f_{uv} \leq c_{uv}, \quad \forall (u, v) \in E$$

- Respects flow conservation (except at s and t):

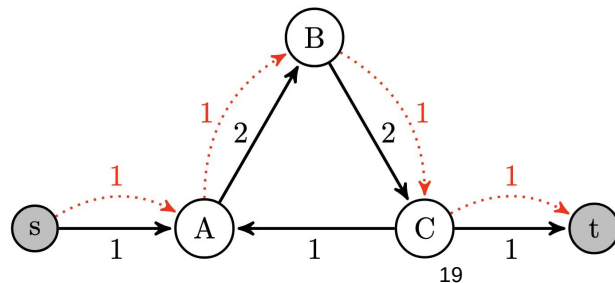
$$\sum_{v \in V} f_{uv} - \sum_{v \in V} f_{vu} = 0, \quad \forall u \neq s, t$$

- Maximizes total flow:

$$\max \sum_{v \in V} f_{sv} = \sum_{v \in V} c_{sv} = \sum_{v \in V} c_{vt} = \max \sum_{v \in V} f_{vt}$$

- Minimizes total cost:

$$\min \sum_{(u,v) \in E} f_{uv} \cdot w_{uv}$$

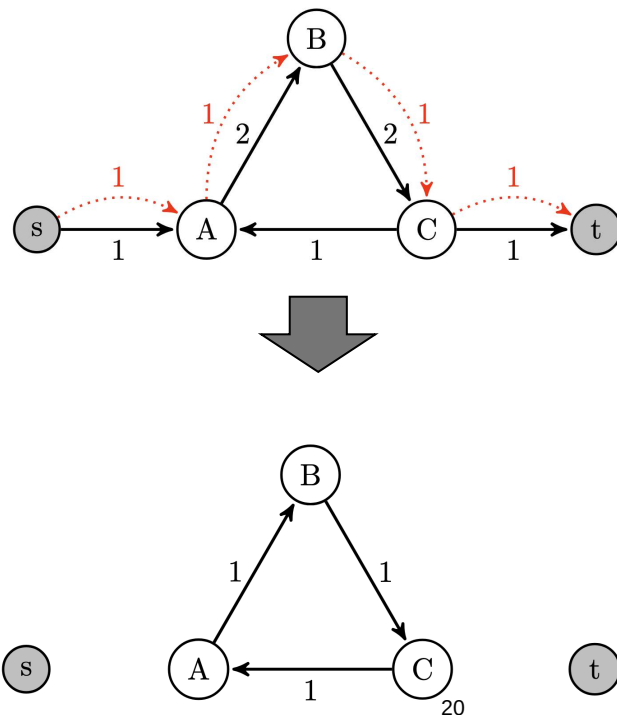


Subtract MCMF flow...

$$b_u = \sum_{v \in V} c_{vu} - \sum_{v \in V} c_{uv}$$

Let $c'_{uv} = c_{uv} - f_{uv}$ be the residual capacity of edge (u, v) . The balance b'_u for a node u in the residual graph G' can be expressed as follows:

$$\begin{aligned} b'_u &= \sum_{v \in V} c'_{vu} - \sum_{v \in V} c'_{uv} \\ &= \sum_{v \in V} (c_{vu} - f_{vu}) - \sum_{v \in V} (c_{uv} - f_{uv}) \\ &= \left(\sum_{v \in V} c_{vu} - \sum_{v \in V} c_{uv} \right) - \left(\sum_{v \in V} f_{vu} - \sum_{v \in V} f_{uv} \right) \\ &= b_u - \left(\sum_{v \in V} f_{vu} - \sum_{v \in V} f_{uv} \right) \end{aligned}$$



... to find a cyclic structure

$$\sum_{v \in V} f_{vi} - \sum_{v \in V} f_{iv} = 0.$$

Flow conservation

Consequently, for $i \in V^0$, we obtain:

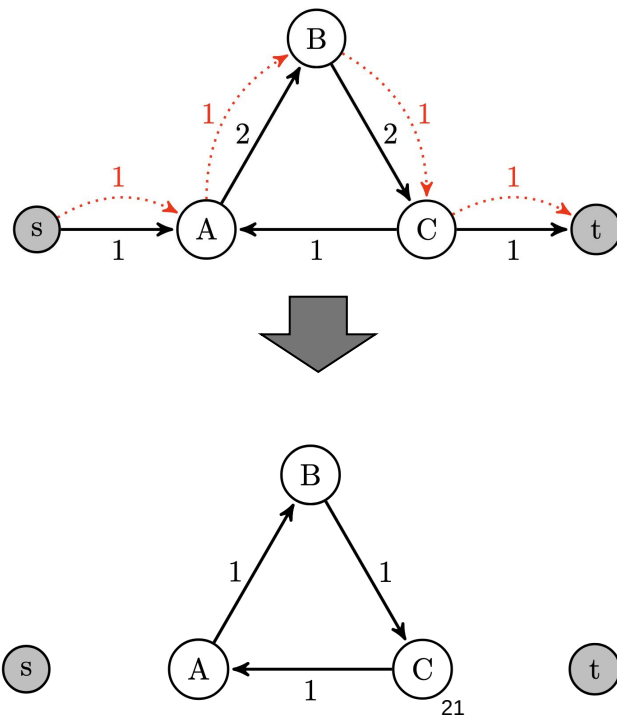
$$b'_i = b_i - \left(\sum_{v \in V} f_{vi} - \sum_{v \in V} f_{iv} \right) = 0 - 0 = 0$$

We still need to demonstrate that the nodes s and t in the residual graph G' are not connected to the rest of the graph. Balance of a source s in residual graph is the starting balance minus all the flows from the source:

$$b'_s = b_s - \left(\sum_{v \in V} f_{vs} - \sum_{v \in V} f_{sv} \right) = - \sum_{v \in V} c_{sv} - \left(0 - \sum_{v \in V} c_{sv} \right) = 0.$$

The similar goes for the sink:

$$b'_t = b_t - \left(\sum_{v \in V} f_{vt} - \sum_{v \in V} f_{tv} \right) = \sum_{v \in V} c_{vt} - \left(\sum_{v \in V} c_{vt} - 0 \right) = 0.$$



MTCS algorithm

MTCS pseudo code

The algorithm starts by mapping out all debts and credits, to add fictitious "source" and "target" to **balance** the entire system.

Then, it figures out the most efficient way to **move fictitious amount from the source to the target**, kind of like finding the best routes for water flowing through a network of pipes.

After **removing that efficient flow**, it leaves behind a **set of circular debts**, where each firm owes money back to itself through a chain of other firms, which can be safely "set off," or removed from the system without any real money changing hands.

Balancing by
adding s,t
nodes

Finding
saturated s-t
flow

Subtracting
saturated flow

```
1: function MTCS(G)
2:   // Input: Weighted directed graph G representing liabilities
3:   // Output: Weighted directed graph G' representing the set-offs
4:   // Graph is a set of weighted edges (debtor, creditor, amount)
5:
6:   s ← "source", t ← "target" // define the source and target nodes
7:
8:   for firm in G.nodes do // For each node in the graph
9:     net_position ←  $\sum_{d \in \text{debtors}(firm)} G_{d, firm} - \sum_{c \in \text{creditors}(firm)} G_{firm, c}$ 
10:    if net_position < 0 then
11:      // Add edge from source to firm if net position is negative
12:      G ← G ∪ {(s, firm, -net_position)}
13:    else
14:      // Add edge from firm to target if the net position is positive
15:      G ← G ∪ {(firm, t, net_position)}
16:    end if
17:  end for
18:
19:  if s ∈ G.nodes then
20:    // Find saturating flow if G is not balanced
21:    sat_flow ← min_cost_max_flow(G, s, t)
22:  else
23:    // Return empty flow if G is balanced
24:    sat_flow ← {}
25:  end if
26:
27:  G' ← {}
28:  for (debtor, creditor, capacity) in G.edges do // Go through all the edges
29:    if (debtor, creditor) ∉ sat_flow then
30:      // Add to cyclic structure if not in the max flow
31:      G' ← G' ∪ {(debtor, creditor, capacity)}
32:    else
33:      // Subtract max flow from capacity
34:      remaining_flow ← capacity - sat_flow[debtor, creditor]
35:      if remaining_flow > 0 then
36:        // Add the remaining flow to the cyclic structure
37:        G' ← G' ∪ {(debtor, creditor, remaining_flow)}
38:      end if
39:    end if
40:  end for
41:  return G'
42: end function
```

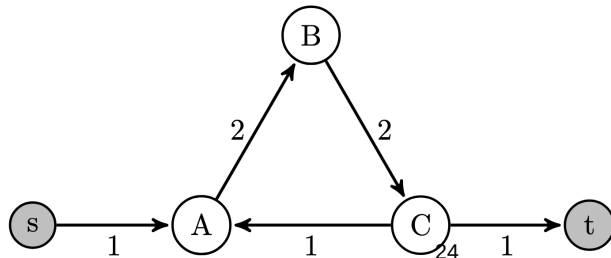
Balancing by adding s,t nodes

Net position is imply a difference between what debtors **owe** firm and what firm **owes** to creditors.

Those with a **negative net position** require additional **flow from source** to balance.

Those with a **positive net position** need to direct surplus **flow to target** or liquidity sink.

```
6:  s ← "source", t ← "target" // define the source and target nodes
7:
8:  for firm in G.nodes do // For each node in the graph
9:    net_position ←  $\sum_{d \in \text{debtors}(\text{firm})} G_{d, \text{firm}} - \sum_{c \in \text{creditors}(\text{firm})} G_{\text{firm}, c}$ 
10:   if net_position < 0 then
11:     // Add edge from source to firm if net position is negative
12:     G ← G ∪ {(s, firm, -net_position)}
13:   else
14:     // Add edge from firm to target if the net position is positive
15:     G ← G ∪ {(firm, t, net_position)}
16:   end if
17: end for
```



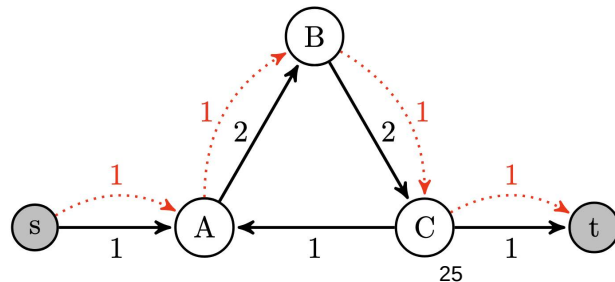
Finding saturated s-t flow

If there is **no source node** in the network, it means the initial network G is **balanced**.

Unbalanced networks can have cycles.

Balanced network can be set-off completely!

```
19: if  $s \in G.nodes$  then  
20:   // Find saturating flow if  $G$  is not balanced  
21:    $sat\_flow \leftarrow min\_cost\_max\_flow(G, s, t)$   
22: else  
23:   // Return empty flow if  $G$  is balanced  
24:    $sat\_flow \leftarrow \{\}$   
25: end if
```



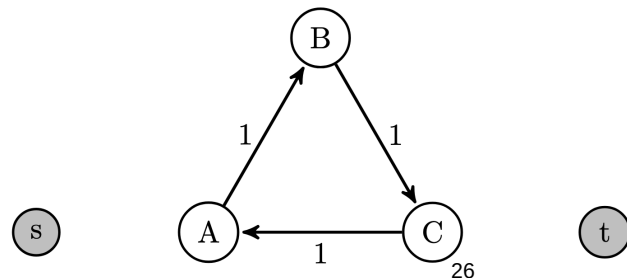
Subtracting saturated flow

MTCS set-off network G' is a subnetwork of obligation network G .

We get G' by subtracting the saturating s-t flow from the G .

Set-off network G' is balanced and can be decomposed into individual cycles. So, we also name it Cyclic Structure.

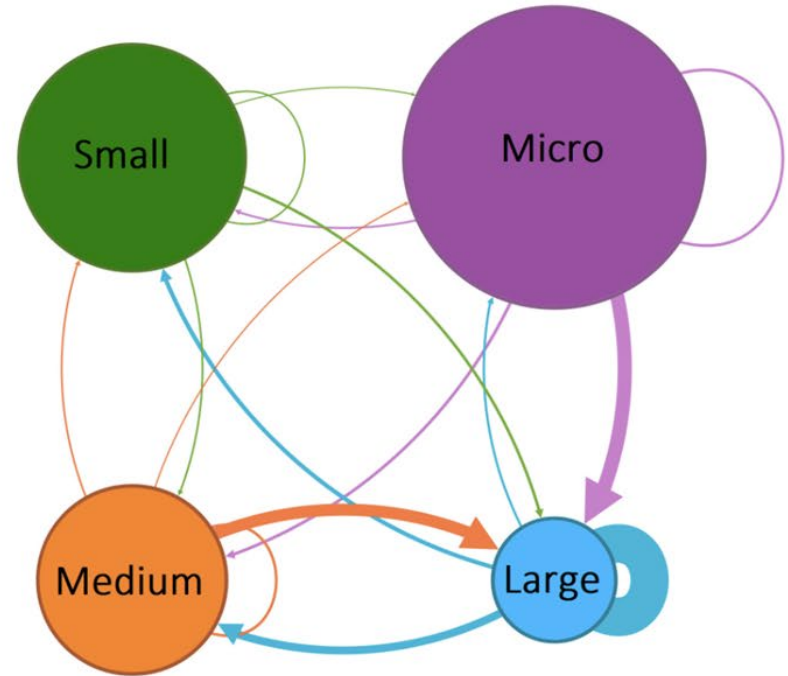
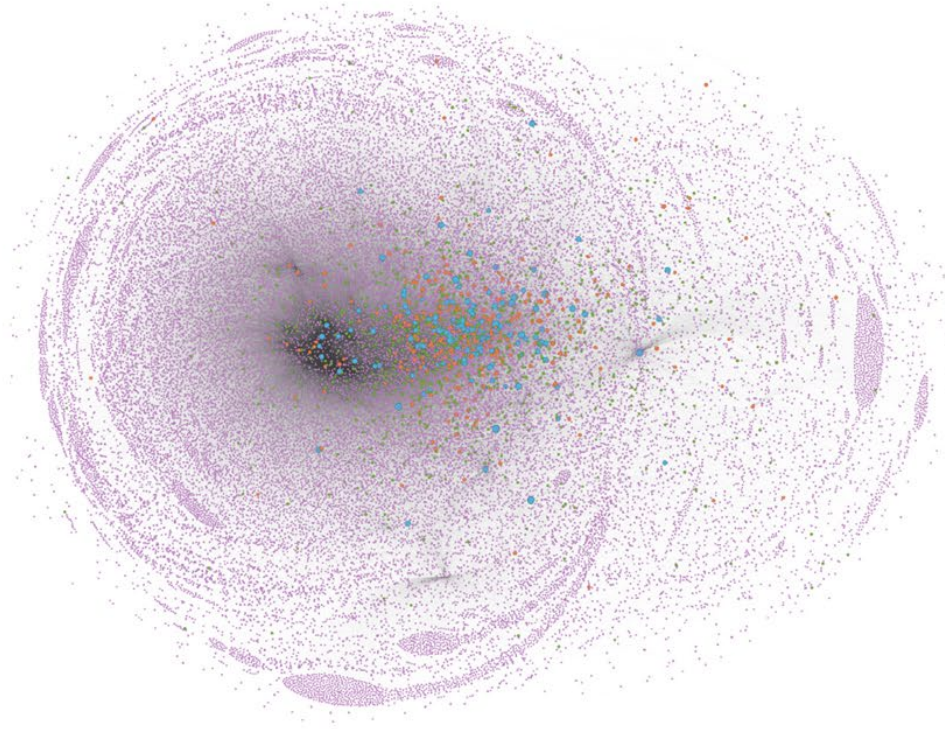
```
27:  $G' \leftarrow \{\}$ 
28: for ( $debtor, creditor, capacity$ ) in  $G.edges$  do // Go through all the edges
29:   if ( $debtor, creditor$ )  $\notin sat\_flow$  then
30:     // Add to cyclic structure if not in the max flow
31:      $G' \leftarrow G' \cup \{(debtor, creditor, capacity)\}$ 
32:   else
33:     // Subtract max flow from capacity
34:      $remaining\_flow \leftarrow capacity - sat\_flow_{debtor, creditor}$ 
35:     if  $remaining\_flow > 0$  then
36:       // Add the remaining flow to the cyclic structure
37:        $G' \leftarrow G' \cup \{(debtor, creditor, remaining\_flow)\}$ 
38:     end if
39:   end if
40: end for
41: return  $G'$ 
42: end function
```



Empirical Setting

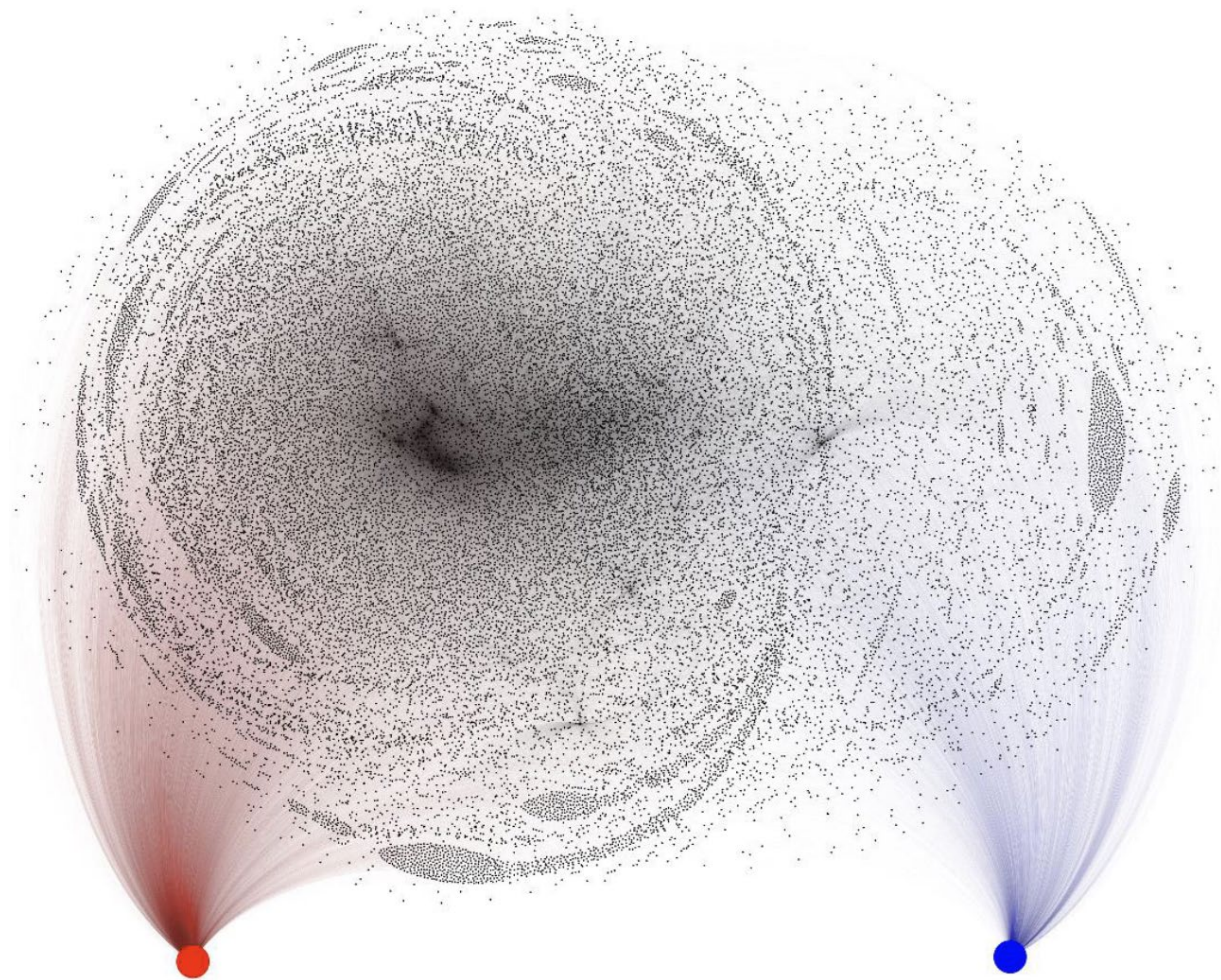
Infocert B2B network

Network of trade credit claims



Balancing the obligations network

Mapping out all debts and credits, to add fictitious "source" and "target" to balance the entire system.



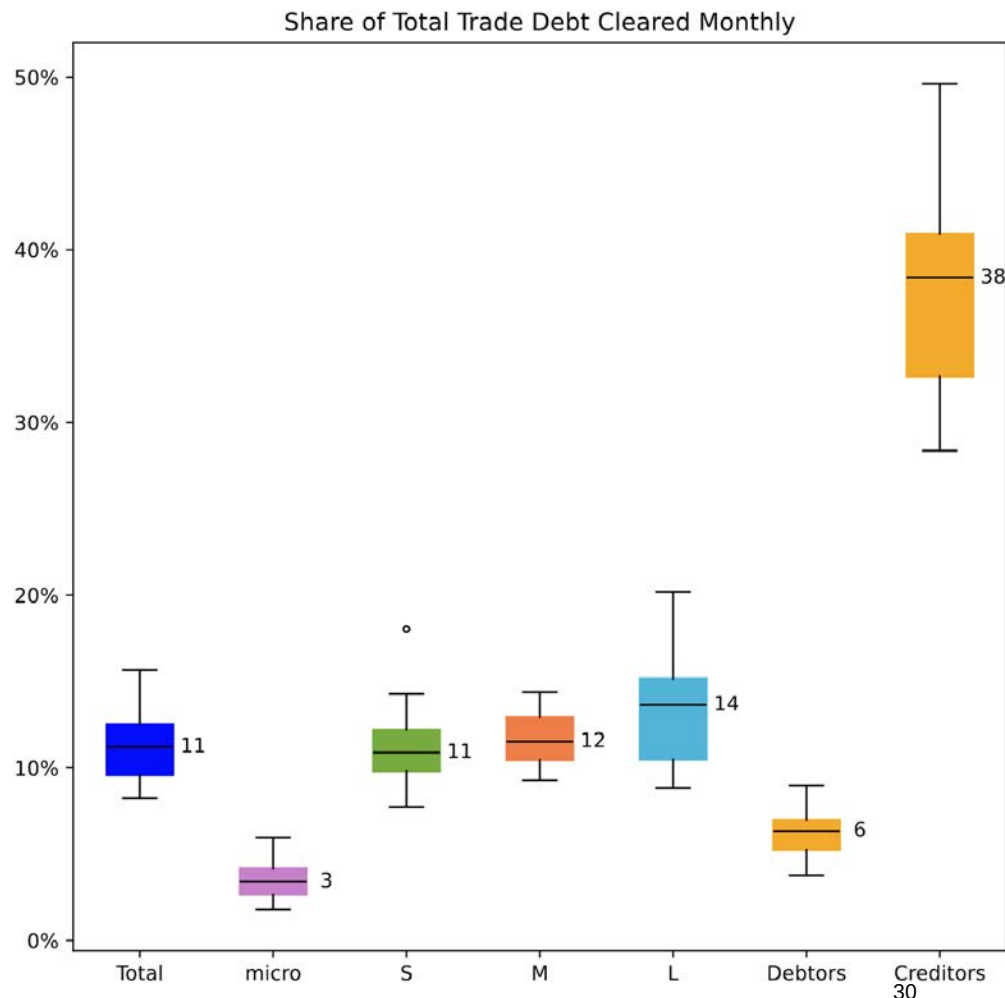
Monthly MTCS results on italian dataset

The results of the MTCS algorithm presented as aggregated monthly set-off amounts over two years period (2019, 2020).

We show the total and the results for subgroups by firm size and firm net position.

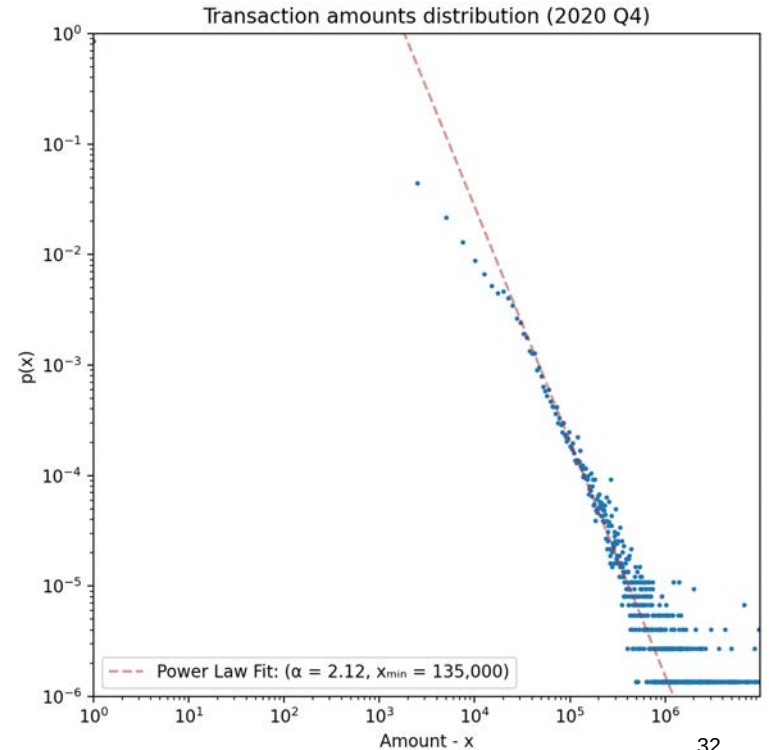
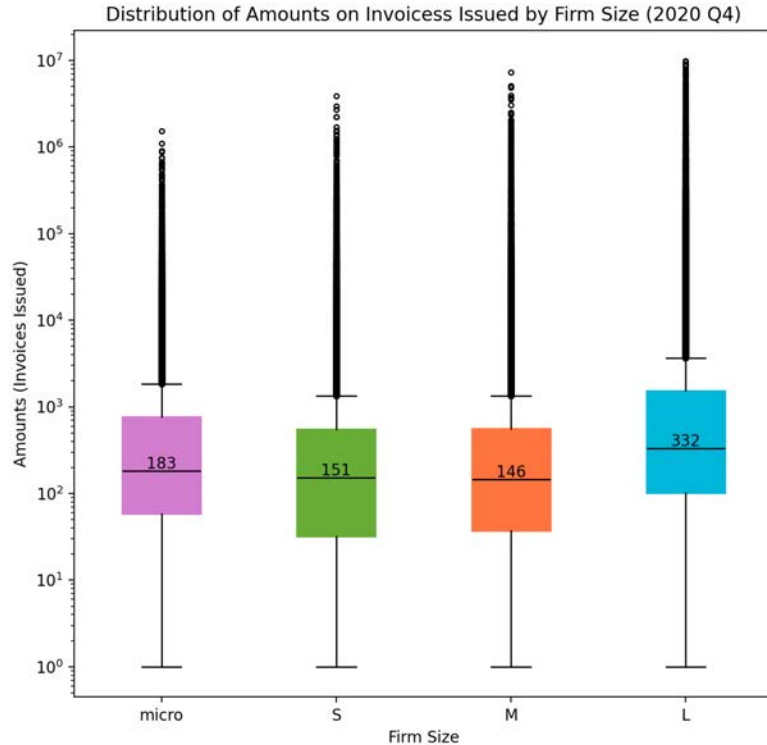
The results are presented as a share of total debt each month.

What sets the limits to MTCS?

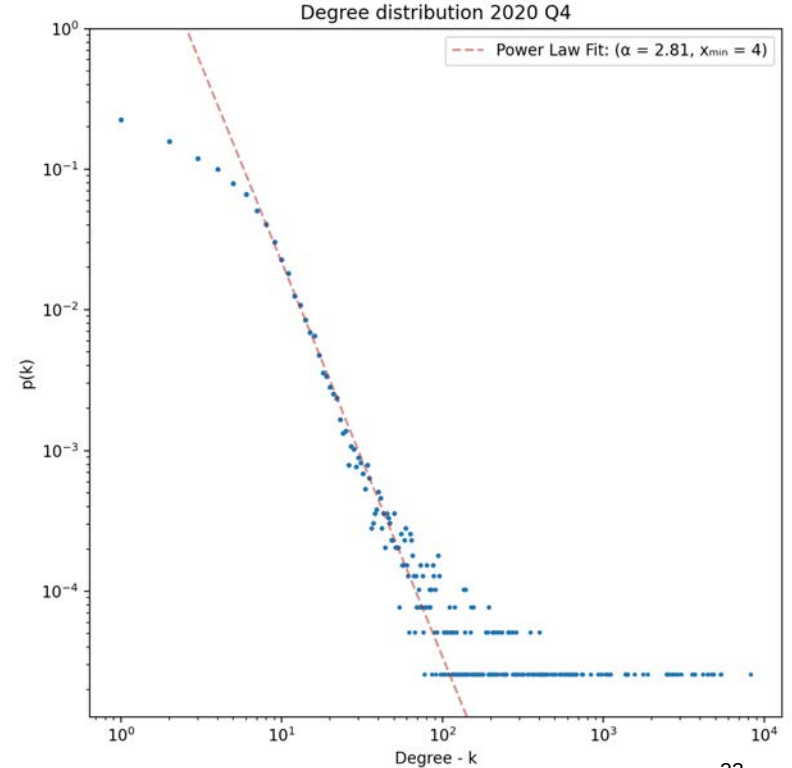
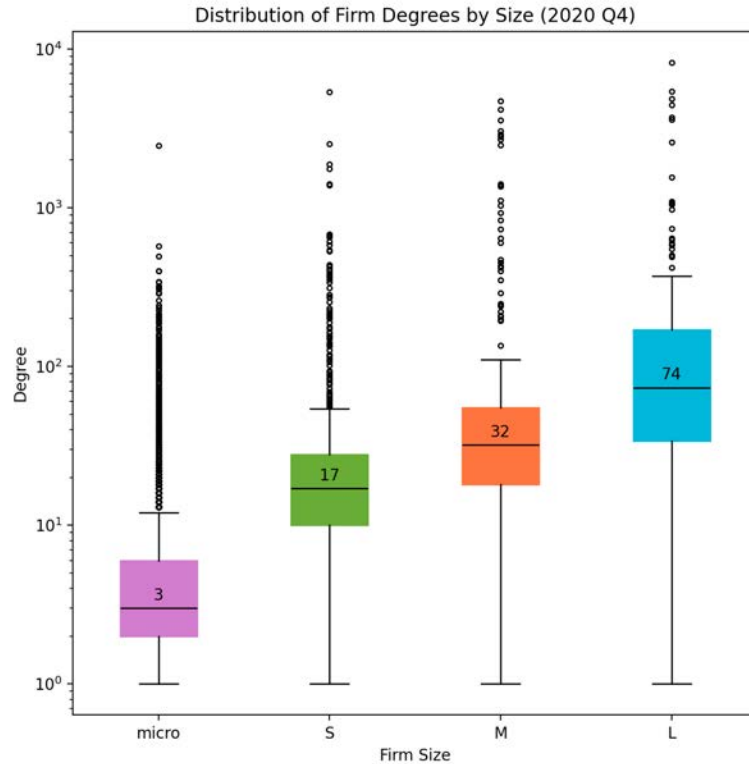


Obligation network Scale Free Topology

Average amounts are small (power law distribution)

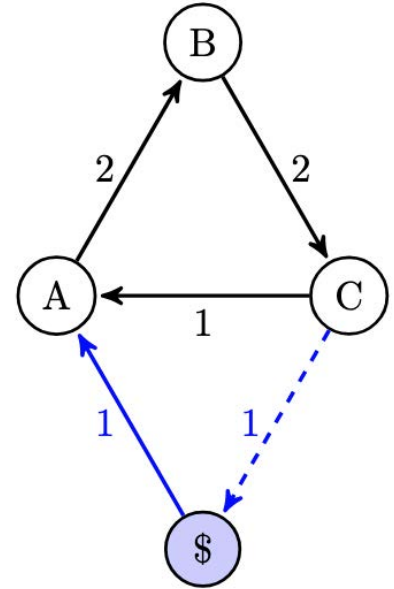
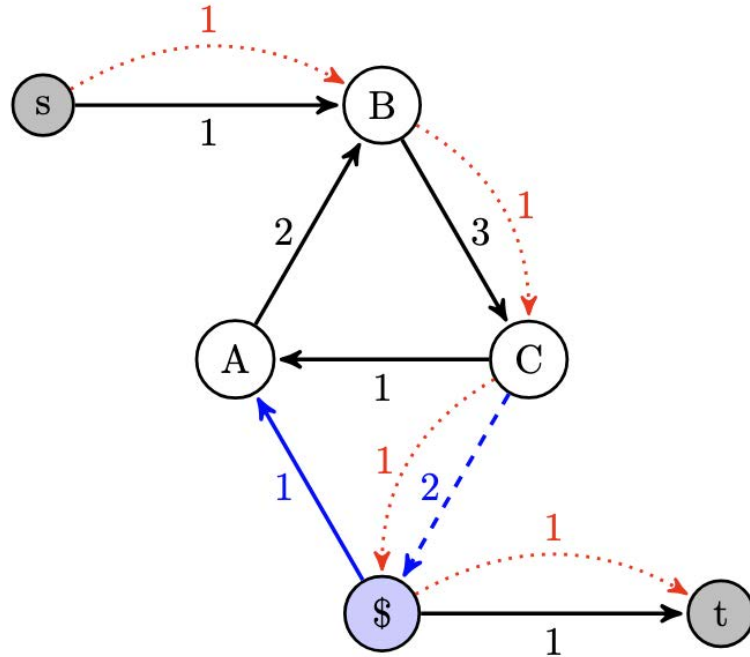
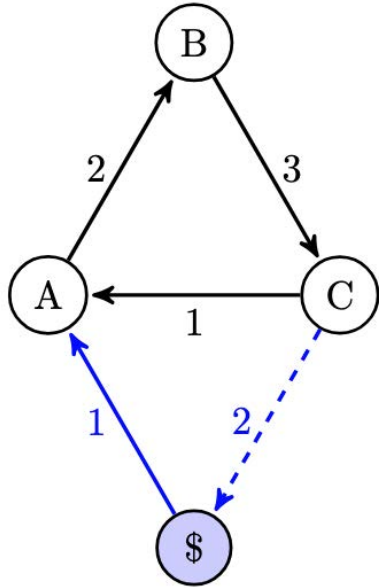


Power law degree distribution



Liquidity Injection

Introducing a liquidity source



Injecting liquidity

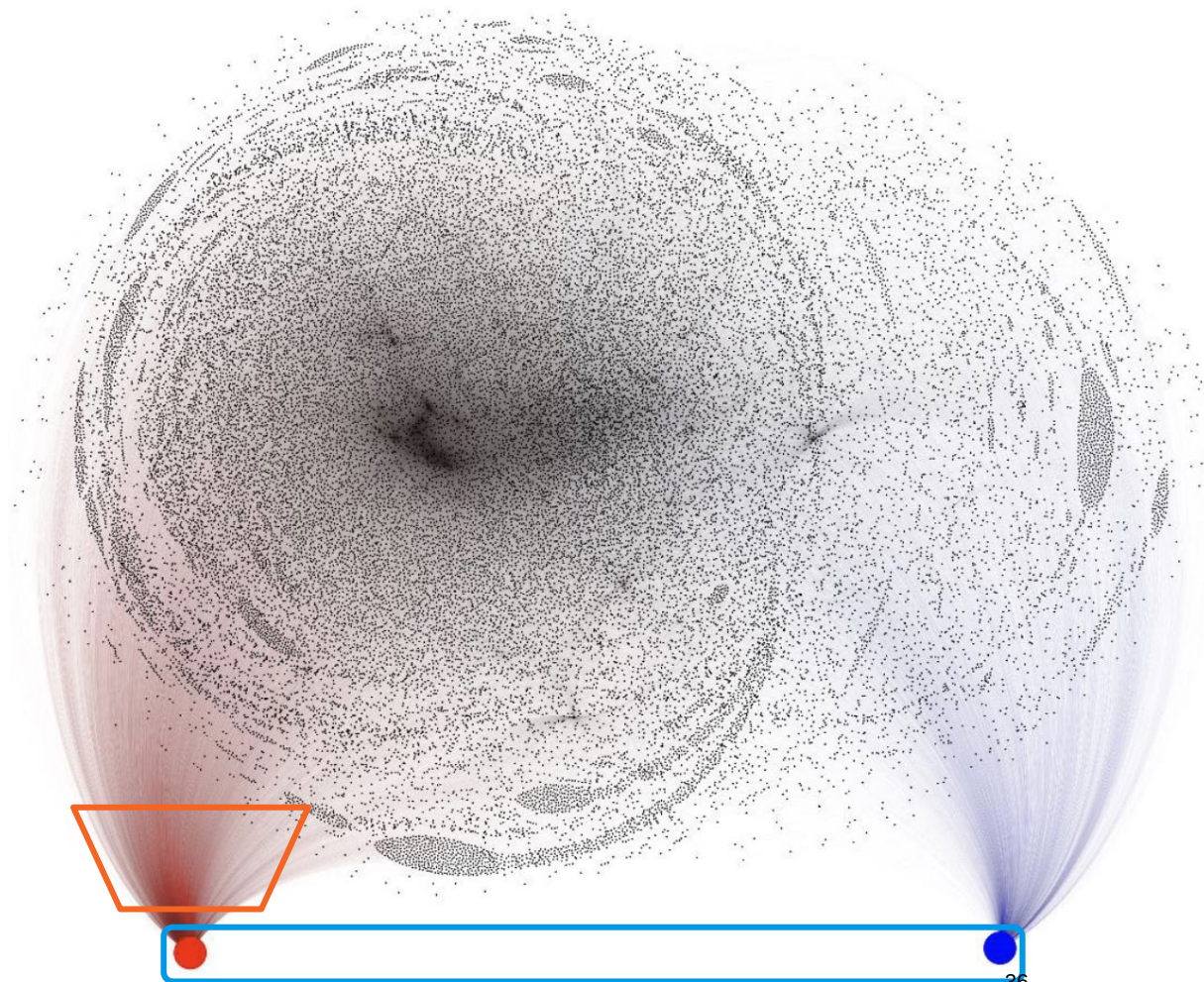
Start with a balanced network

Create liquidity source by **merging source and sink into one liquidity source node**.

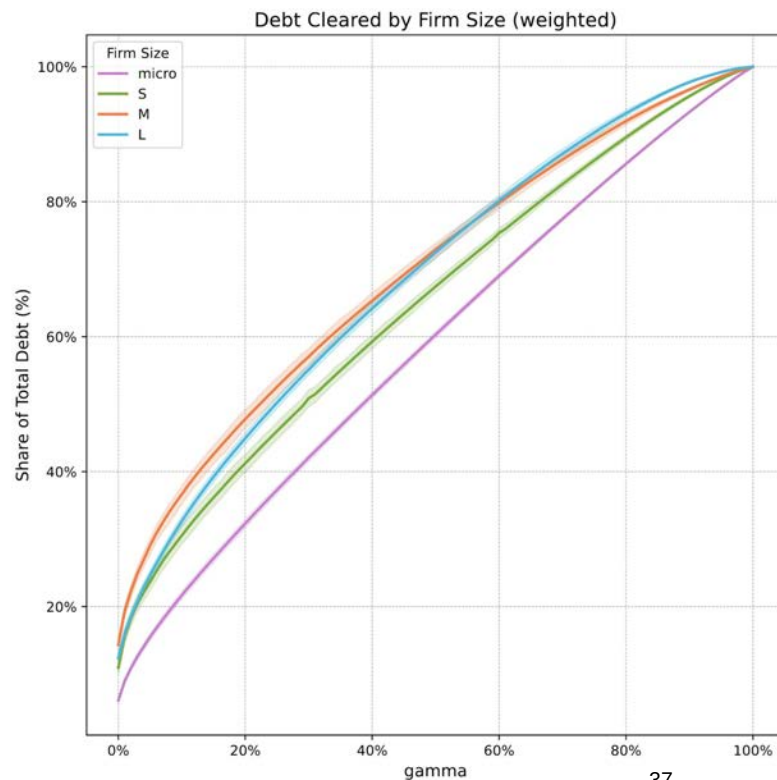
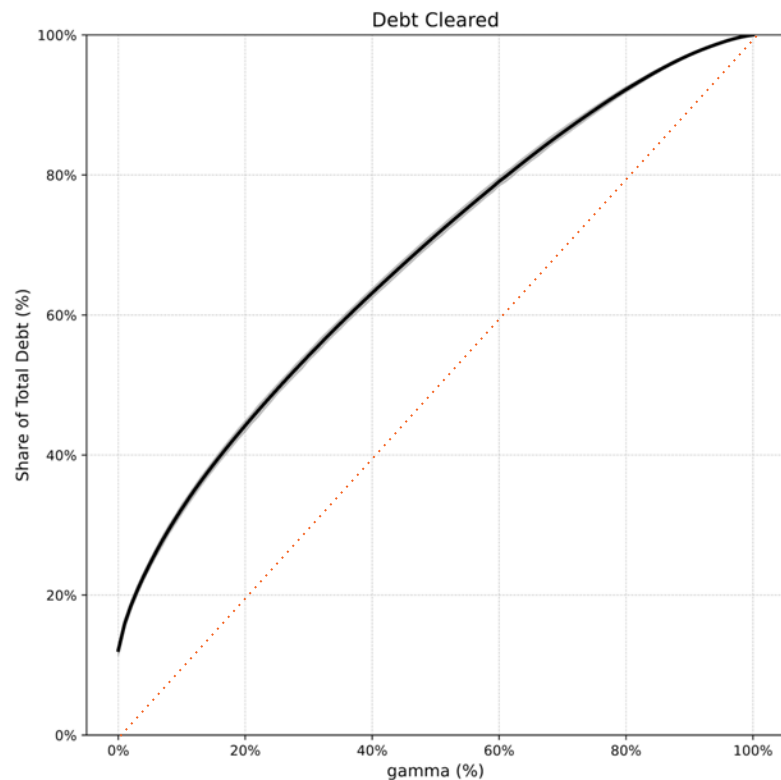
Multiply amounts on red edges with a factor

gamma from interval $[0,1]$,
to set the liquidity injection level.

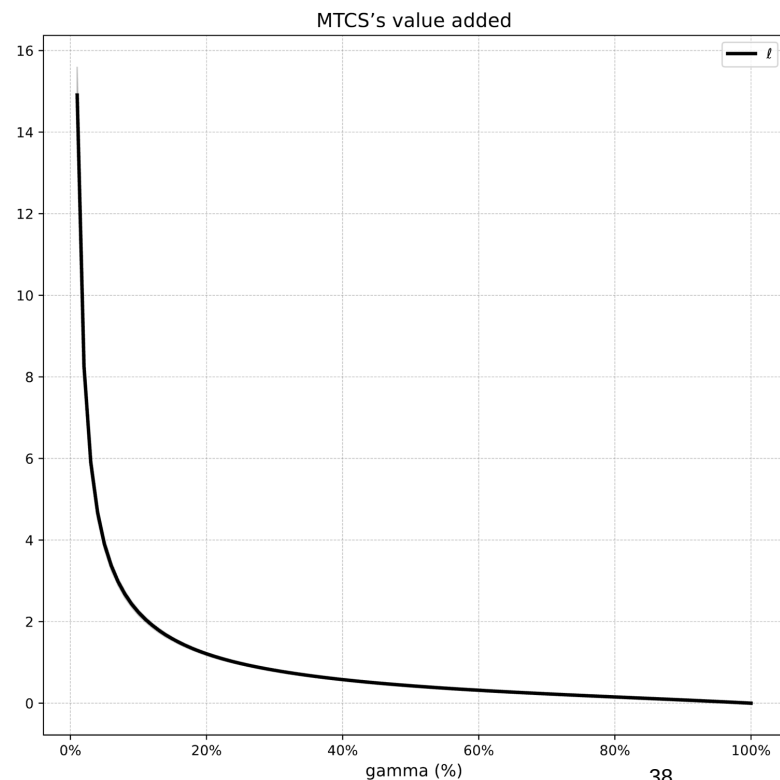
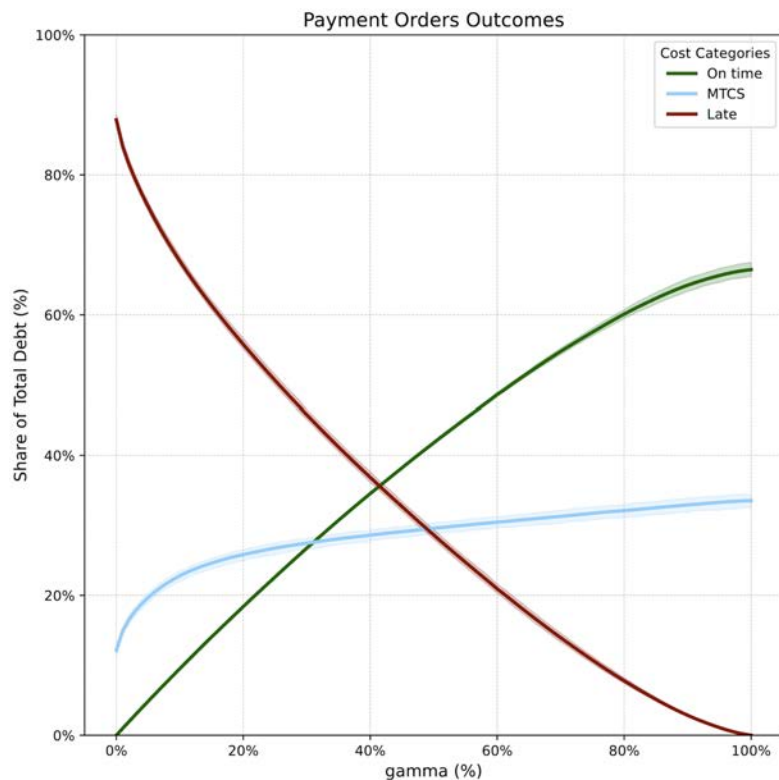
Rebalance the network with new source and sink nodes to inject funds with maximum discharge of debt in the network.



Debt cleared - liquidity injection



Payment orders outcome and MTCS value added



Why does it matter?

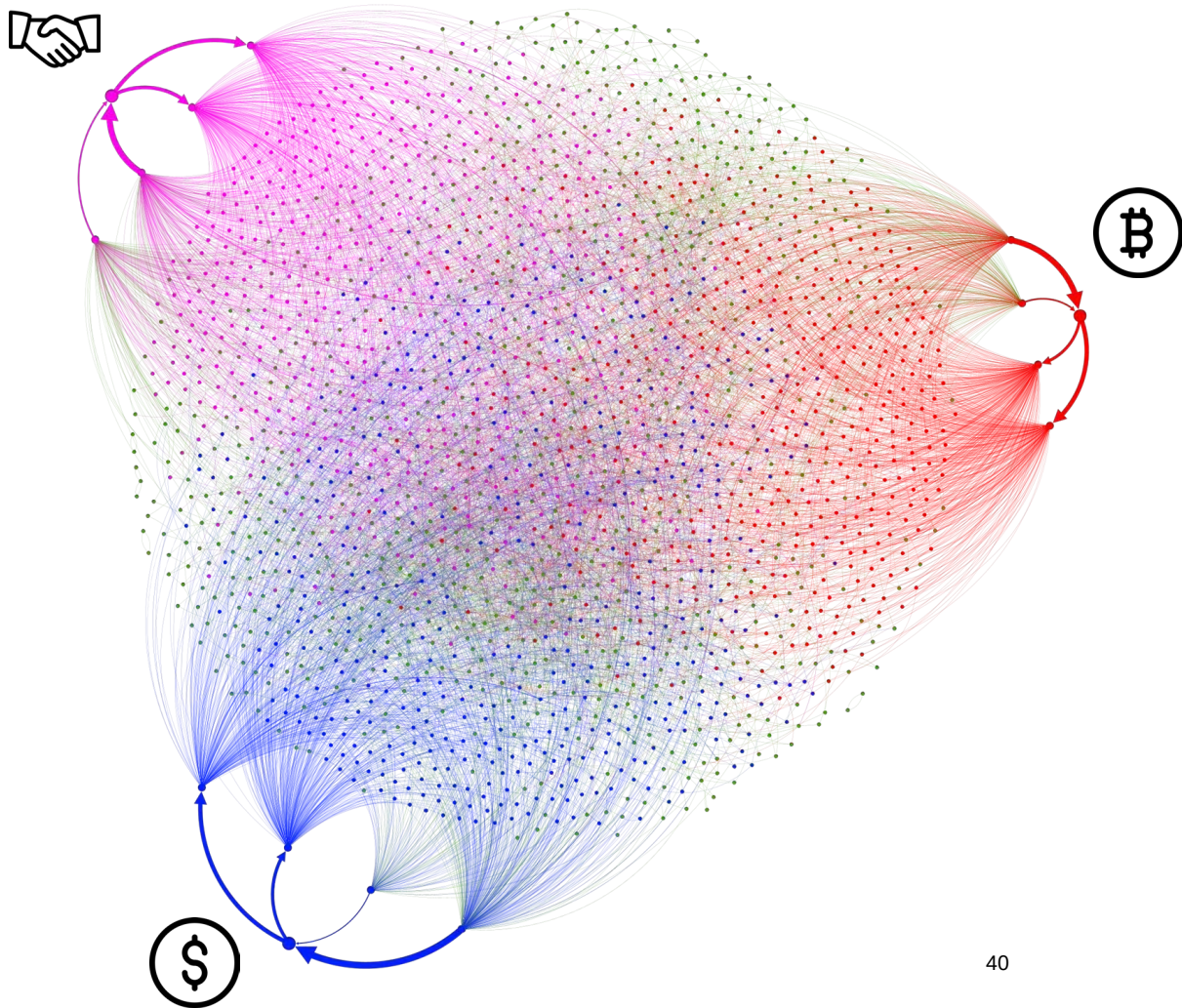
Liquidity comes in many forms

Efficient discharge of obligations is possible even if liquidity provided comes from completely *disconnected liquidity* sources.

- Bank credit
- Digital Assets
- Mutual Credit

Subject of further research and development!

Liquidity provision can be incentivised by inclusion of *premium/discount* scheme to create a liquidity market.



Cycles

The Open Clearing Protocol

MTCS blockchain implementation

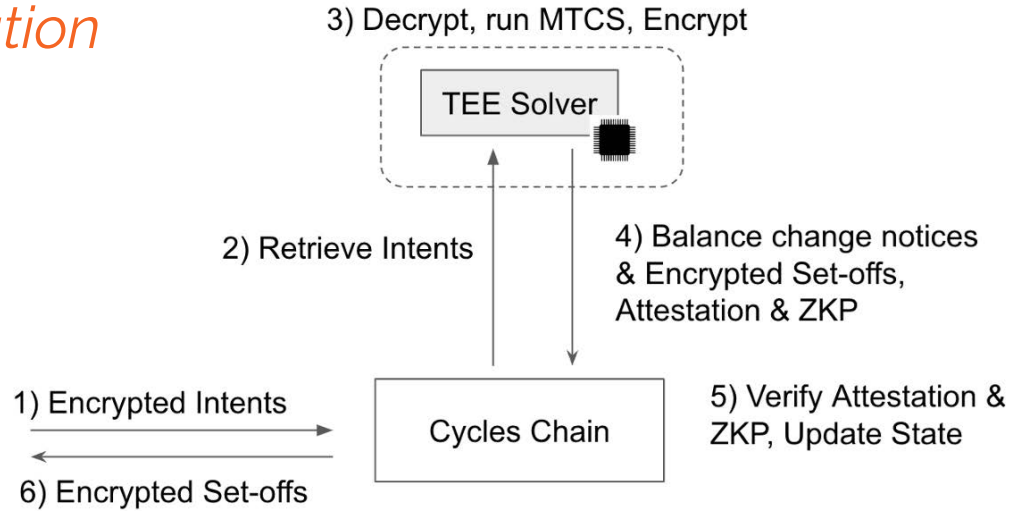
Cycles is a decentralized protocol for **clearing, settlement, and issuance**, designed to overcome **payment inefficiencies** and reduce working capital costs.

It leverages **diverse liquidity** sources, including crypto, to **clear more** debt **with less** money.

Cycles uses a **privacy-preserving** multilateral settlement platform based on **graph optimization**, recognizing that liquidity is accessible through optimized **settlement flows** within payment network.



Users



Source:

<https://cycles.money/whitepaper.pdf>

CYCLES

The Open Clearing Protocol

Tomaž Fleischman

✉ tomaz@informal.systems

🦋 [@t-fleischman.bsky.social](https://bsky.social/t-fleischman)

🌐 cycles.money

MIT OpenCourseWare

<https://ocw.mit.edu>

14.129 Advanced Contract Theory Spring 2025

For more information about citing these materials or our Terms of Use, visit <https://ocw.mit.edu/terms>.