International Journal of Business and Economics Research
2020; 9(2): 83-93
http://www.sciencepublishinggroup.com/j/ijber
doi: 10.11648/j.ijber.20200902.15
ISSN: 2328-7543 (Print); ISSN: 2328-756X (Online)

The Gridlock-Proof Functionality in Real Time Gross Settlement Systems

Vladimir Kulipanov

Department of Control and Applied Mathematics, Moscow Institute of Physics and Technology, Moscow, Russian Federation

Email address:
vladimir.kulipanov@gmail.com

To cite this article:
Vladimir Kulipanov. The Gridlock-Proof Functionality in Real Time Gross Settlement Systems. *International Journal of Business and Economics Research*. Vol. 9, No. 2, 2020, pp. 83-93. doi: 10.11648/j.ijber.20200902.15

Received: February 21, 2020; **Accepted**: March 5, 2020; **Published**: March 10, 2020

Abstract: The last two decades of the XX century marked the starting point for the central banks across the globe to move their payment and settlement systems into Real Time Gross Settlement (RTGS) mode. Despite the fact that RTGS systems can effectively eliminate the credit exposure between the paying bank and the receiving bank at the interbank level by means of fast final and irrevocable money transfer, there is another serious problem associated with these systems. RTGS systems turned out to be liquidity-demanding arrangements, as opposed to deferred net settlement systems. Thus, the efficiency of liquidity management arrangements is the precondition of smooth RTGS operation, especially in tough times when liquidity is a systemic shortage. If liquidity management is inefficient, the RTGS system may stop operating properly by terminating in the grid-lock state brining chaos to the national economy. In this research we suggest the practical approach to solve the problem of seeking the maximization of aggregate value of payment instructions under liquidity shortage, including the most severe scenarios. The classification of the RTGS queue statuses is suggested and discussed. Some complementary results are articulated, including: (a) the statement that the formal mathematical optimization problem lies in the NP class of the computational complexity (the class of problems solved in polynomial time by nondeterministic Turing machine); (b) the equivalence between MaxFlow-MinCost problem (from the network flow theory) and the dual linear problem of the linear program relaxation of the initial optimization problem; (c) the illustration that no optimization strategy, other than the suggested one, can deliver substantially better optimization results. Despite enormous efforts, there were no previous research results reasonably claiming the near optimality of liquidity optimization strategy in RTGS systems under severe liquidity shortages. The results of this research may help the central banks and other RTGS system operators to ensure the protection of their payment systems from future liquidity crises and bring the resilience of respective national economies to the next level of sustainability.

Keywords: Operational Research, Bank Clearing Problem, Max Flow Min Cost Problem, Gridlock Resolution, Liquidity Efficiency, Liquidity Saving Mechanism, Integer Linear Programming

1. Introduction

1.1. Historical Notes

Back in August 1998 just after few days the Russian Government announced it's default of State Short-Term Government Bonds the payment system of the Central Bank of the Russian Federation was pressed by the most severe shock in its history. The liquidity in banking system was wiped away by the panic of clients and by the increased value of their obligations in Russian currency. The interbank lending market froze at zero volumes. The queues of unsettled payment instructions due to the lack of funds on the accounts of credit institutions in the CBR payment system started to appear and grow. At the end-of-day procedures the quantity of rejected payment instructions was measured in dozens per cent of the total flow.

As part of the response measures to the dramatically changing conditions the Bank of Russia undertook a sequence of actions, where the alternate calculation algorithm was one of those. However, it took several hours to complete the calculations and the algorithm could not be admitted as an acceptable solution at that moment.

The research to improve the characteristics of the algorithm were resumed after the shock of the liquidity crisis disappeared. This publication outlines the key academic results of this research, including the proof that the formal problem belongs to NP class, the calculation strategy based on identified peculiarities of the real-world instances of the problem, the illustration that no other gridlock resolution strategy may bring results substantially better than those illustrated here.

1.2. The Statuses of the Queue of Payments in RTGS System

To deal with possible liquidity shortage modern RTGS systems have the queueing mechanism. When the funds are insufficient to immediately process the new-come payment instruction, it is put in the queue in anticipation of future settlement opportunity. The settlement is triggered either by the offsetting payment instruction or at the specified time later during the day.

We say that at the time when the settlement is triggered, the queue at the RTGS system is in the "general" status, where it is possible to select the sub-collection of the payments and simultaneously execute them as a group. The remaining payment instructions continue to wait the next settlement opportunity. The settlement procedure is thus trying to transform the status of the queue from "general" to some other category.

It is natural to request that the sub-collection of payments subject for execution is as large as possible. Therefore, the remainder of the queue should not contain further immediate easy opportunities for the settlement.

We call the status of the queue as "gridlocked" when no further settlement is possible, provided that FIFO rule and priorities are respected. Thus, the transformation from the "general" status to "gridlocked" status will be the first type of the queue transition.

We call the status of the queue as "deadlocked" when no further settlement is possible, with no requirement to observe the FIFO rule and priorities. Thus, the transformation from the "general" status to "deadlocked" status will be the second type of the queue transition.

We call the status of the queue as "minimally deadlocked" when the collection of the remaining payments is minimal across all "deadlocked" queues, to which the initial "general" status can transit. Thus, the transformation from the "general" status to "minimally deadlocked" status is the third and the most wanted type of the queue transition. The respected reader may reasonably ask, why this third type of transition is so welcome, when the banks usually tend to respect FIFO rule and payment priorities, while this type of transition seems not to. The answer is that the response from the banking community is seriously depended on the form of the question they are being asked. If we reformulate the same question (whether they would prefer to respect the FIFO rule and payment priorities) to read as "what would you prefer: to leave waiting a handful of prioritized hundred dollar payments in the queue OR ALTERNATIVELY to leave there dozens of thousands of other payments most of which may be of higher value". It is unlikely that the banks will produce the same answer.

The picture below illustrates the statuses and the transition options.

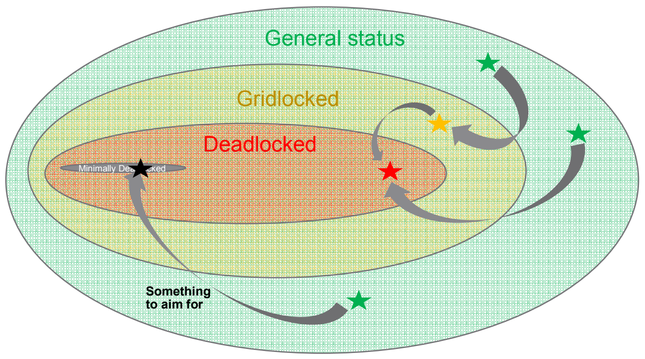

Figure 1. *The statuses and the transition options.*

If the RTGS system has the third option of payment queue transition, it never ends up in the "gridlocked" state. It demonstrates the unparalleled resilience to severe liquidity shortages, producing no headache to the payment system operator and participating banks, and can be called the gridlock-proof.

2. The Proposed Solution Strategy

2.1. Mathematical Formalism

Let us consider the payment instructions submitted for settlement procedure to the payment system by the participating banks. Each of such instructions has it's own unique identification number, as well as identifies the account to be debited, the account to be credited and the value to be moved.

Let n be the total quantity of the accounts affected in any way by at least one of those payment instructions.

Let B_i – be the value of the resource kept at the account number i and available for settlement during the settlement cycle (account balance plus overdraft limit minus active restrictions).

We consider the general case, where there can be more than one payment instruction that tries to debit account number i and to credit the account number j. Let a_{ijk} be the value of the $k-th$ such payment instruction. We then introduce the decision variable x_{ijk}, taking it's value =1, if the corresponding payment instruction can be settled, and it's value =0 in the opposite case. At some stages of the solving procedure we shall refrain from obeying the integrality restriction to the decision variables. In such cases we consider the variables x_{ijk} to be within

$$0 \leq x_{ijk} \leq 1 \tag{1}$$

interval.

Since there can be no such account with negative resource at the end of settlement cycle, the difference between the aggregated value of debiting payment instructions and aggregated value of crediting payment instructions should not exceed B_i. The corresponding inequality restriction has the form

$$\sum_{jk} a_{ijk} * x_{ijk} - \sum_{pq} a_{piq} * x_{piq} \leq B_i \qquad (2)$$

It can be easily verified that the quantity of such restrictions is equal to the quantity of the accounts, that is n.

We seek to maximize the aggregate value of all settled payment instructions, therefore we shall maximize the Objective function

$$F = \sum_{ijk} a_{ijk} * x_{ijk} \qquad (3)$$

In order to change the character of the restriction in (1) and (2) from inequality to equality we shall introduce the ancillary non-negative variables ξ_{ijk} and ζ_i.

$$0 \leq \xi_{ijk} \qquad (4)$$

$$0 \leq \zeta_i \qquad (5)$$

With their help the collection of formulas (1) - (3) will be re-written as follows:

$$\begin{cases} 0 \leq x_{ijk} \\ 0 \leq \xi_{ijk} \\ 0 \leq \zeta_i \\ x_{ijk} + \xi_{ijk} = 1 \\ \sum_{jk} a_{ijk} * x_{ijk} - \sum_{pq} a_{piq} * x_{piq} + \zeta_i = B_i \end{cases} \qquad (6)$$
$$x_{ijk} - integer$$
$$F \to max$$

And here we get the formal optimization problem. Michael M. Güntzer, Dieter Jungnickel, Matthias Leclerc gave it the name “Bank clearing problem”, providing each sub-class of the problem with subscript showing the number of participating banks (accounts) [5]. For example, they called the sub-class of the problem where 20 accounts are visible, as BCP_{20}.

We shall refrain from sub-classification and providing the subscripts for the reason that regardless of the quantity of the participating banks (accounts), this problem falls into the one wide class of integer linear programming.

The point, where

$$\begin{cases} x_{ijk} = 0 \\ \xi_{ijk} = 1 \\ \zeta_i = B_i \end{cases} \qquad (7)$$

will be the initial feasible basic solution.

2.2. Computational Complexity

There is a widely accepted perception that the problem (6) is NP-complete [5]. However, the available proof is based on the illustration that the BCP_2 is actually the Subset-Sum Problem. Thus, the proof is formally valid only for the case where only two banks participate.

The correct proof for the entire BCP problem (6) must be based on the reduction of any known NP-complete problem to the BCP.

Below we demonstrate the less strict property of the BCP and show that it lies in NP class by reducing it to 0-1 integer linear programming problem, which is known to be NP-complete.

Theorem. Problem (6) lies in NP.

The unknown variables x_{ijk} are 0-1 integers. The integrality of x_{ijk} and the equation $x_{ijk} + \xi_{ijk} = 1$ yield that ξ_{ijk} are also 0-1 integers. Finally, the unknown variables ζ_i are integers in range $0 \leq \zeta_i \leq B_i$.

Let us consider the representation of the ζ_i in the binary notation.

$$\zeta_i = \sum_{l=0}^{L} h_{il} * 2^l$$

The representation of any number in the binary notation uses only zeros and ones. Therefore each h_{il} is 0-1 integer, L is non-negative integer and is well below the length of the Turing machine code of the entire BCP problem submitted for processing.

Therefore, we may re-write the problem (6) in the form:

$$\begin{cases} 0 \leq x_{ijk} \\ 0 \leq \xi_{ijk} \\ 0 \leq h_{il} \leq 1 \\ x_{ijk} + \xi_{ijk} = 1 \\ \sum_{jk} a_{ijk} * x_{ijk} - \sum_{pq} a_{piq} * x_{piq} + \sum_{l=0}^{L} h_{il} * 2^l = B_i \end{cases} \qquad (8)$$
$$x_{ijk} - integer, h_{il} - integer$$
$$F \to max$$

After the transformation all the unknowns are 0-1 integers. All other values are integers.

This problem is 0-1 integer programming which is known to be NP-complete (Richard Manning Karp (1972). "Reducibility Among Combinatorial Problems". In R. E. Miller; J. W. Thatcher; J. D. Bohlinger (eds.). Complexity of Computer Computations. New York: Plenum. pp. 85–103)

Despite the fact that the problem is of a well-known class, no known algorithm is capable of solving it to optimality under the restricted time and computer memory (2020). It exhibits the exponentially growing computation complexity as the number of variables increases.

2.3. Inventions and Tricks to Overcome the Computation Complexity

2.3.1. “Second-best” Solution Instead of the Optimum

Given the fact that the number of payments in CBR payment system is 500 000 (half a million) per hour, even the rough estimate will prove the optimal solution be impossible to find within reasonable time. In order to attack the problem further, we shall have to forget about global optimum and to be content with so-called “second-best” solution. The term “second-best” means that the solution we seek to find, is (i) feasible to the restrictions of problem (6); (ii) provably close to the global optimum of (6) in terms of the Objective function; (iii) unrestrictedly far from the global optimum of

(6) in terms of Euclidian distance.

2.3.2. Linear Program Relaxation

In order to seek the "second-best" solution it is quite natural to start form taking away the integrality restriction for the decision variables x_{ijk}.

After that the problem is re-formulated with no requirement for the variable x_{ijk} to hold integer values:

$$\begin{cases} 0 \le x_{ijk} \\ 0 \le \xi_{ijk} \\ 0 \le \zeta_i \\ x_{ijk} + \xi_{ijk} = 1 \\ \sum_{jk} a_{ijk} * x_{ijk} - \sum_{pq} a_{piq} * x_{piq} + \zeta_i = B_i \end{cases} \quad (9)$$

$$F \to max$$

This is nothing more than a linear program.

It has $(n + 2m)$ variables and $(n + m)$ restrictions, where m – is the quantity of all payment instructions submitted for processing.

The effective solving algorithms for the problems of the kind are well described in the literature. The most known is the simplex-method. It takes $(2n + 3m)^3$ atomic arithmetic operations to solve the problem to optimality in our case. Other polynomial estimates for simplex method in arbitrary matrixes can be found in [12] and [13].

But even dramatically reduced computation complexity of the problem requires substantial computational power which might be beyond available capacity. There are two main reasons for that, and both could be effectively overcome.

2.3.3. Reducing the Quantity of Variables in Linear Program

The first reason of high computation complexity is that the number of variables in (9) is high. To reduce this number we may insert one new variable y_{ij} instead of multiple variables x_{ijk}. We can do so since we no longer are bound by the integrality restriction for variables x_{ijk}. We denote the sum $\sum_k a_{ijk}$ as c_{ij}.

$$c_{ij} = \sum_k a_{ijk}$$

Since every x_{ijk} cannot exceed 1, the following observation is true

$$\sum_k a_{ijk} * x_{ijk} \le \sum_k a_{ijk} = c_{ij}$$

Therefore, unless $\sum_k a_{ijk} = 0$, the formal definition for the variable y_{ij} will be

$$y_{ij} = \frac{\sum_k a_{ijk} * x_{ijk}}{\sum_k a_{ijk}}$$

If $\sum_k a_{ijk} = 0$, then there is no payment instruction that tries to debit account number i and to credit the account number j. In this case the variable y_{ij} will not be introduced.

It immediately follows from the definition, that

$$0 \le y_{ij} \le 1$$

With the help of above mentioned substitution, the problem (9) will take the form

$$\begin{cases} 0 \le y_{ij} \\ 0 \le \psi_{ij} \\ 0 \le \zeta_i \\ y_{ij} + \psi_{ij} = 1 \\ \sum_j c_{ij} * y_{ij} - \sum_p c_{pi} * y_{pi} + \zeta_i = B_i \end{cases} \quad (10)$$

$$G \to max$$

Where the Objective function G inherits the properties of Objective function F and is naturally defined as

$$G = \sum_{ij} c_{ij} * y_{ij} \quad (11)$$

The ancillary non-negative variables ψ_{ij} are introduced in order to ensure that inequality $y_{ij} \le 1$ transforms into equation $y_{ij} + \psi_{ij} = 1$.

The point where

$$\begin{cases} y_{ij} = 0 \\ \psi_{ij} = 1 \\ \zeta_i = B_i \end{cases} \quad (12)$$

will serve as the initial feasible basic solution.

Thus the quantity of variables has been reduced by one order of magnitude.

Once we know the solution to (10), we will derive the respective non-integer variables x_{ijk} by means of the following simple routine, which should be run for every valid[1] pair of $\{i, j\}$:

```
*******
BEGIN
Assign all $x_{ijk} := 0$
For each $s$ in range $\{1, k\}$
    (Try) $x_{ijs} := 1$
    If $\sum_k (a_{ijk} * x_{ijk}) > c_{ij} * y_{ij}$
        Then (redeem) $x_{ijs} := 1 - \frac{\sum_k (a_{ijk} * x_{ijk}) - c_{ij} * y_{ij}}{a_{ijs}}$
        Exit routine
    Else (leave) $x_{ijs} := 1$
    End if
Next s
END.
*******
```

This routine sequentially tries each x_{ijk} to be equal 1, until such assignment results in violation of the leading inequality $\sum_k a_{ijk} * x_{ijk} \le \sum_k a_{ijk}$. Ahead of the violation, all x_{ijk} variables get the value = 1. Once the violation has occurred, the respective x_{ijk} variable gets non-integer value somewhere in between zero and one, and the remaining x_{ijk} variables get value = 0.

1 valid pair of $\{i, j\}$ means $\sum_k a_{ijk} > 0$.

2.3.4. Switching to Dual Linear Program

The second reason that this linear problem (10) requires substantial computational power is that the initial basic solution (12) is very "far" from the optimal one. Both in terms of Objective function and in terms of Euclidian distance. The calculation procedure may take too long to complete, whilst the result will be the optimal solution of the linear program, not the original integer linear program. To deal with this second difficulty we consider the dual linear program.

Before this discussion, it is worth making a little step back from the formalism of linear problem (10) and taking away the ancillary variables. Formalism (10) is good for computer implementation, but the theoretical discussion goes better without them.

Thus, we are a little bit back to

$$\begin{cases} 0 \le y_{ij} \\ y_{ij} \le 1 \\ \sum_j c_{ij} * y_{ij} - \sum_p c_{pi} * y_{pi} \le B_i \end{cases} \qquad (13)$$
$$G \to max$$

2.3.5. Dual Linear Program and It's Interpretation in Terms of Network Flow

The idea is to solve the dual linear program instead of the direct one. We are not aware of any other research with regard to "bank clearing problem" where the authors implement the concept of linear duality, ahead of our initial publication in 2015. In this sense the result found in that work was not known before. The concept of linear duality is well described in the available literature (e.g., [14, 15, 16]). With the "bank clearing problem" the switch to duality yields substantial computation benefits to be described below.

The following observation will provide the reader with the high-level understanding of these benefits. Assume that the initial ("direct") linear problem (13) for BCP is formulated in the way "Start from the point where nothing is settled. Settle as much as possible under resources restriction", the dual linear problem is formulated in the way "Start from the point where everything is settled, even if the balances of some accounts will go to infeasible negative area. Bring the accounts to the consistent state (non-negative balance) whilst rejecting as less as inevitable". The computational benefit comes from the fact that in the real-world instances of the BCP there are few accounts that are in non-consistent state (their balances are initially negative) at the starting point of dual linear problem. In other words, it is hard to imagine that even under the most severe liquidity shortage there will be more than a handful of banks unable to cover their net debit positions. The vast majority will have non-negative net debit positions. Therefore the amount of computational work in dual linear problem of BCP is nearly one order of magnitude less, compared to the computational work in direct linear problem.

Now we shall transform the linear program (13) into a dual one. (The reader is addressed to [14, 15, 16] for details).

According to the theory of linear duality, each variable of the "direct" linear program has the associated restriction in the dual linear program. Let us take the particular variable y_{ij}. We consider the associated column in matrix (13) which generates the corresponding restriction in the dual linear program. This column has only three non-zero elements.

The first non-zero element comes with (c_{ij}) coefficient, derived from $\sum_j c_{ij} * y_{ij} - \sum_p c_{pi} * y_{pi} \le B_i$ inequality (associated with index i of the account-balance restrictions in the "direct" linear program).

The second non-zero element appears in $\sum_p c_{ip} * y_{ip} - \sum_j c_{ij} * y_{ij} \le B_j$ inequality (associated with index j of the account-balance restrictions in the "direct" linear program), and is equal to $(-c_{ij})$.

Finally, the last non-zero element (equals 1) appears in $y_{ij} \le 1$ inequality (associated with the $\{(i,j)\}$ pair of indexes of the "y_{ij} upper-bound" restrictions in the "direct" linear program).

Since the respective coefficient in the Objective function G equals (c_{ij}), the inequality of the dual linear problem will have the form

$$c_{ij} * z_i - c_{ij} * z_j + 1 * f_{ij} \ge c_{ij} \qquad (14)$$

where z_i, z_j, f_{ij} are the dual variables. Each of them is non-negative.

It is now important to draw attention to the function z_i. This function is defined for every node (that is, for every account) and may be interpreted as a price function in the residual network of the BCP. Before proceeding to residual network of the BCP it is important to define the BCP network.

2.4. The Properties of the BCP Network and BCP Residual Network

2.4.1. Description of the BCP Network

We shall start with the definition of the flow in the BCP network.

The BCP network is defined as directed graph, where every node corresponds to the respective account.

The ordered pair (i,j) of the nodes in the BCP network is connected by the edge, which has two parameters: (a) the capacity, and (b) the cost.

The function *flow* over the edge (i,j) is feasible, if

$$0 \le flow(i,j) \le capacity(i,j)$$

The cost of the edge (i,j) in the BCP network is constant and equals 1.

The term $capacity(i,j)$ will serve as interpretation of the aggregate value of payment instructions which try to debit the account i and credit the account j. The term $flow(i,j)$ will serve as interpretation of the aggregate value of rejected payments in the direction from the account i to the account j.

The cost of the entire flow in the BCP network is

$$cost(flow) = \sum_{i,j} flow(i,j) * cost(i,j) = \sum_{i,j} flow(i,j)$$

We shall define the intensity of flow over the node i as follows:

$$Intensity(i) = \sum_q flow(i,q) - \sum_p flow(p,i)$$

The intensity of the node may be thought of as an indicator how much does the particular node inject into or consume from the rest of network.

We shall refer to the node i as Debtor if this node has strictly positive intensity, and we shall refer to the node i as Creditor if this node has strictly negative intensity. If the intensity of the node is exactly zero, then this node is balanced and will be neither the Debtor nor the Creditor.

To make the proof of the below statements easier, the two additional nodes are introduced into the BCP network.

The node S (stands for "source") and the node T (stands for "termination").

We shall connect the node S with every Debtor by the edge with capacity:

$$Capacity(S, Debtor) = Intensity(Debtor)$$

We shall connect every Creditor with the node T by the edge with capacity:

$$Capacity(\, Creditor, T) = -Intensity(Creditor)$$

The cost of the new edges $(S, Debtor)$ and $(\, Creditor, T)$ in the BCP network will also be equal to 1.

We shall extend the definition of the flow in the BCP network by defining the capacity and the flow over the edges $(S, Debtor)$ $(Creditor, T)$.

The flow over the edge $(S, Debtor)$ will be:

$$flow(S, Debtor) = capacity(S, Debtor) = Intensity(Debtor)$$

where the value of $Intensity(Debtor)$ was calculated before the introduction of the nodes S and T.

The capacity and the flow over the edge $(Creditor, T)$ will be:

$$flow(Creditor, T) = capacity(Creditor, T) = -Intensity(Creditor)$$

where the value of $Intensity(Creditor)$ was calculated before the introduction of the nodes S and T.

It is easy to see that the extended flow in the BCP network leaves every node (except the nodes S and T) balanced.

It is also easy to see that $Intensity(S) = -Intensity(T)$

It is, therefore, reasonable to define the intensity of the entire flow in the extended BCP network as follows:

$$Intensity(flow) = Intensity(S)$$

From now on, by referring to BCP network we shall mean the extended BCP network.

We are now ready to describe the BCP residual network.

2.4.2. Description of the BCP Residual Network

The BCP residual network is defined as directed graph, where every node corresponds to the respective account, in full analogy with the BCP network. (Two extra nodes S and T from the extension are also included from the very beginning).

As opposed to BCP network, the ordered pair (i,j) of the nodes in the BCP residual network is connected by the edge, which has three parameters: (a) the available direct capacity, (b) the available reverse capacity, and (c) the cost.

The available direct capacity in the BCP residual network is defined as the difference between the capacity in the BCP network and the flow over the edge in the BCP network.

$$available\ direct\ capacity\ (i,j) = capacity(i,j) - flow(i,j)$$

The available reverse capacity in the BCP residual network is defined as the flow over the counter-directed edge in the BCP network.

$$available\ reverse\ capacity\ (i,j) = flow(j,i)$$

We shall define the cost of the edge (i,j) in the BCP residual network as follows:

$$cost(i,j) = \begin{cases} -1, & if\ available\ reverse\ capacity\ (i,j) > 0 \\ 1 & \begin{pmatrix} if\ available\ reverse\ capacity\ (i,j) = 0\ and \\ available\ direct\ capacity\ (i,j) > 0 \end{pmatrix} \\ infinity & \begin{pmatrix} if\ available\ reverse\ capacity\ (i,j) = 0\ and \\ available\ direct\ capacity\ (i,j) = 0 \end{pmatrix} \end{cases}$$

The term "infinity" is actually used to represent any value which is above the cost of any feasible chain along the edges with non-zero capacity (both direct or reverse) in the BCP residual network. The value 3*n is well suited for that role, for the reason that even if the chain passes through every node in the BCP residual network over the edges with zero reverse capacity and non-zero direct capacity, the cost of that chain will be (n+1) at most. Passing through the edges with zero available capacity is prohibited.

2.4.3. The Maximum Flow in the BCP Network and in the BCP Residual Network

Definition of the Max flow in BCP network

The feasible function *flow* with the maximum possible intensity in the extended BCP network will be Max *flow* function.

Properties of the BCP residual network

Claim 1.

Any feasible flow in the non-extended BCP network is a Max flow in the BCP extended network with the source S and termination T.

The proof immediately follows from the definition of the BCP extended network. It is sufficient to point out that the source node S is connected to the rest of the BCP extended network by the edges with the exhausted capacities. And so does the node T. Therefore it is impossible to transfer more from the node S to the node T compared to what is transferred by the known function *flow*.

2.4.4. The Maximum Flow Minimum Cost in the BCP Network

Claim 2.

The Max flow function *flow* in the BCP extended network is also Max flow Min cost if and only if there are no negative cost cycles in the BCP residual network.

Part 1 (if there is a negative cost cycle in the BCP residual network, then the function *flow* is not Min cost)

Consider the negative cost cycle in the BCP residual network. Since it's cost is negative, every edge in the respective cycle in the extended BCP network has either positive available reverse capacity, or positive available direct capacity. The reason is that otherwise (if both direct and reverse capacities are zeros) the cost of the respective edge in the BCP residual network would be infinity, and the cost of the entire cycle will be infinity as well.

Every node in the cycle of the BCP residual network may be transited by the edges in the four possible combinations regarding their costs. The first option is that we arrive to the node via the edge with the $cost = -1$ and leave the node via the edge with the $cost = -1$. The second option is that we arrive to the node via the edge with the $cost = -1$ and leave the node via the edge with the $cost = +1$. The third option is that we arrive to the node via the edge with the $cost = +1$ and leave the node via the edge with the $cost = -1$. And finally, the fourth option is that we arrive to the node via the edge with the $cost = +1$ and leave the node via the edge with the $cost = +1$.

Now we shall show how to modify the *flow* in the extended BCP network, so that it's intensity remains the same, but the cost is decreased.

Each edge *(u,v)* of the cycle produces the flow alternation opportunity according to the rule:

$$\Delta f_{u,v} = \begin{cases} available\ direct\ capacity(u,v);\ if\ cost(u,v) = 1 \\ available\ reverse\ capacity(u,v);\ if\ cost(u,v) = -1 \end{cases}$$

Note, that $\Delta f_{u,v}$ is strictly positive value ($\Delta f_{u,v} > 0$).

We shall denote $\Delta f = \min_{\{(u,v)\}} \Delta f_{u,v}$. That is, we compare all flow alternation opportunities along the edges and select the least one. Note, that Δf is also a strictly positive value ($\Delta f > 0$).

Let us produce the recipe to the *flow* alternation in the BCP extended network:

If the edge *(u,v)* of the cycle has the $cost(u,v) = -1$ in the BCP residual network, we shall decrease the flow over the counter-directed edge *(v,u)* in the extended BCP network by the value Δf. This alternation is possible, since:

$$\Delta f \leq \Delta f_{u,v} = available\ reverse\ capacity(u,v) = flow(v,u)$$

If the edge *(u,v)* of the cycle has the $cost(u,v) = +1$ in the BCP residual network, we shall increase the flow over the same edge *(u,v)* in the extended BCP network by the value Δf. This alternation is possible, since:

$$\Delta f \leq \Delta f_{u,v} = available\ direct\ capacity(u,v)$$

Now we shall consider every transition option (of the four possible) in the BCP residual network and discuss how the balance of the respective node is affected by the suggested flow alternation. Assume that we interested to observe the balance on the node *v* in the cycle and we arrive there from node *u* and leave towards node *w*.

In the first option where we arrived via the edge *(u,v)* with the $cost = -1$ and leaved this node via the edge *(v,w)* with the $cost = -1$, we have decreased both the inbound flow over the edge *(w,v)* and the outbound flow over the edge *(v,u)* by the same value Δf. Thus, the balance of the node *v* is not affected by the change.

In the second option where we arrived via the edge *(u,v)* with the $cost = -1$ and leaved this node via the edge *(v,w)* with the $cost = +1$, we have increased the outbound flow over the edge *(v,w)* and decreased the outbound flow over the edge *(v,u)* by the same value Δf. Thus, the balance of the node *v* is not affected by the change.

In the third option where we arrived via the edge *(u,v)* with the $cost = +1$ and leaved this node via the edge *(v,w)* with the $cost = -1$, we have decreased the inbound flow over the edge *(w,v)* and increased the inbound flow over the edge *(v,u)* by the same value Δf. Thus, the balance of the node *v* is not affected by the change.

Finally, in the fourth option where we arrived via the edge *(u,v)* with the $cost = +1$ and leaved this node via the edge *(v,w)* with the $cost = +1$, we have increased both the outbound flow over the edge *(v,w)* and the inbound flow over the edge *(u,v)* by the same value Δf. Thus, the balance of the node *v* is not affected by the change.

Therefore, we may conclude that the suggested flow alternation is feasible and does not change the intensity of the flow in the extended BCP network. However, the cost of the alternated flow can be obtained from the cost of the initial flow by adding the value which is equal to Δf times the cost of the cycle, which is strictly negative. For that reason the initial flow cannot be Min cost.

Part 2 (if function *flow* is not Min cost then there is a negative cost cycle in the BCP residual network)

Assume that another function $flow^{\circledast}$ which is of the same intensity and is Min cost. Consider the flow which is the difference between the functions $flow^{\circledast}$ and *flow*. This difference is circulation, since it brings the intensity of every node to zero. Note, that the cost of this circulation is negative according to the BCP residual network. Since every circulation can be entirely decomposed into cycles, then the cost of at least one these cycles has to be negative.

2.4.5. The Price Function

Now we are ready to define the price function in the BCP residual network. This function can be used to evaluate the optimality (to verify Min cost property) of the non-extended BCP network.

Consider the ordered pair of nodes (j, i), which has strictly positive capacity (either direct or reverse) from the node j to the node i in the BCP residual network. For the BCP residual network we call the price function z feasible if for every edge (j, i) with strictly positive capacity (from the node j to the node i) it holds:

$$z(i) - z(j) + cost(j, i) \geq 0 \tag{15}$$

Claim 3.

The BCP residual network has feasible price function z if and only if it contains no negative cost cycles.

Part 1 (If there is a feasible price function in the BCP residual network, then there are no negative cost cycles)

Assume that there is a feasible price function z in the BCP residual network. Let us evaluate the cost of any given cycle in the BCP residual network. The cost of the cycle is a sum of all cycle edges:

$$\sum_{\{(u,v)\}\in cycle} cost(u, v)$$

We add and subtract the sum of prices of vertexes along the cycle:

$$\sum_{\{(u,v)\}\in cycle} cost(u, v) = \sum_{\{(u,v)\}\in cycle} cost(u, v) + \left(\sum_{v\in cycle} Z_v - \sum_{u\in cycle} Z_u\right)$$

Then, regroup

$$\sum_{\{(u,v)\}\in cycle} cost(u, v) = \sum_{\{(u,v)\}\in cycle} (cost(u, v) + Z_v - Z_u)$$

Use (15), and obtain

$$\sum_{\{(u,v)\}\in cycle} cost(u, v) \geq 0$$

That is, the cost of any given cycle in the BCP residual network is non-negative.

Part 2 (If there are no negative cost cycles in the BCP residual network then there is a feasible price function)

We are going to show that the shortest path distance (further: Spd) of any vertex u in the residual BCP network, where the cost of an edge is taken as it's length may be used to produce the price function. We define the function

$$\varphi(u) = -Spd(u)$$

Now we shall show that $\varphi(u)$ meets the definition requirement of the price function.

Let us start from the termination point T. Assign the $Spd(T) = 0$

Compute the shortest path distances for every reachable vertex using Bellman-Ford algorithm. It is possible, since there are no negative cost cycles. If the vertex is not reachable, it means that there is now flow through this node, and it's price may be taken arbitrary. We shall assign the price=0 in such cases.

According to the properties of the shortest path distance:

$$Spd(u) \leq Spd(v) + cost(v, u)$$

It easily yields:

$$Spd(v) - Spd(u) + cost(v, u) \geq 0$$

$$-\varphi(v) + \varphi(u) + cost(v, u) \geq 0$$

$$\varphi(u) - \varphi(v) + cost(v, u) \geq 0$$

Which is exactly the definition of the price function (15)

Once the BCP residual network does not contain negative-cost cycles – the respective flow of the BCP network has the minimum cost. Thus, the existence of the feasible price function over the BCP residual network is the criteria for the minimum cost of the flow in the BCP network. Remember that the existence of that function is a precondition of any feasible solution to the dual linear program (14). That is, every feasible solution to the dual linear program produces the MinCost property of the flow in BCP network.

Please note, that (15) and (14) are equivalent. Below is the short proof of that.

Claim 4. (15) and (14) are equivalent statements

Since $cost(j, i) \geq -1$, we may introduce the value $g_{ij} = cost(j, i) + 1 \geq 0$

$$z(i) - z(j) + cost(j, i) \geq 0$$

$$z(i) - z(j) + cost(j, i) + 1 - 1 \geq 0$$

$$z(i) - z(j) + (cost(j, i) + 1) - 1 \geq 0$$

$$z(i) - z(j) + g_{ij} - 1 \geq 0$$

$$z(i) - z(j) + g_{ij} - 1 \geq 0$$

$$z(i) - z(j) + g_{ij} \geq 1$$

$$c_{ij} * z(i) - c_{ij} * z(j) + c_{ij} * g_{ij} \geq c_{ij}$$

Since both the unknown variable f_{ij} and the product $c_{ij} * g_{ij}$ are non-negative values, we are under no burden to define $f_{ij} = c_{ij} * g_{ij}$. This equation should be treated as guidance how to obtain the value of the dual variable f_{ij} based on the knowledge of the $cost(j, i)$, and vice versa (how to obtain the value of the $cost(j, i)$ based on the knowledge of the dual variable f_{ij}).

This immediately yields:

$$c_{ij} * z(i) - c_{ij} * z(j) + f_{ij} \geq c_{ij}$$

The Objective function of the dual linear problem brings the intensity of the flow in the BCP network to the maximum. That is, the entire dual linear program to the BCP is

MaxFlow MinCost problem.

2.5. Some Consequences and Complementary Results

Based on the MaxFlow MinCost interpretation for the BCP residual network we may group the accounts according to their respective prices.

The picture below illustrates the plausible outcome of that procedure:

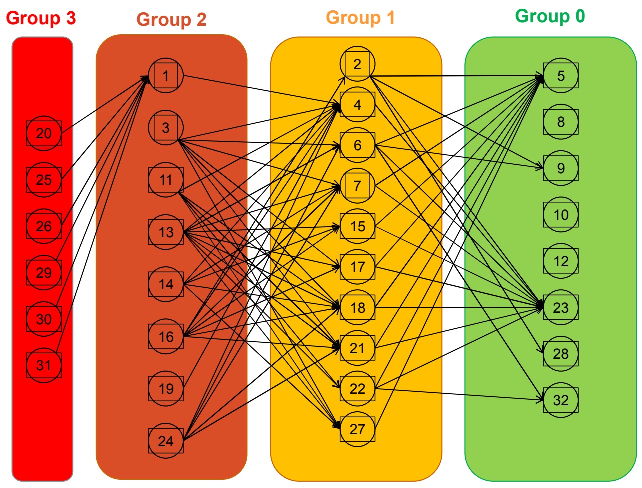

Figure 2. *The plausible outcome of that procedure.*

The higher the number of the group, the more valuable is the liquidity, that may be injected onto the account from this group. For example, if we consider the task to decrease the value of rejected payments, the first candidate to receive the injection of liquidity will be the account from the left-most group.

Consider the injection of liquidity into the account from the group zero. It is useless from the perspective of decreasing the value of the rejected payments. The reason is simple: there are no rejected payments originating from the accounts in this group.

Consider the injection of liquidity into the account from the group three. It triggers a well-predicted chain of cascading settlement passing through each of the groups:

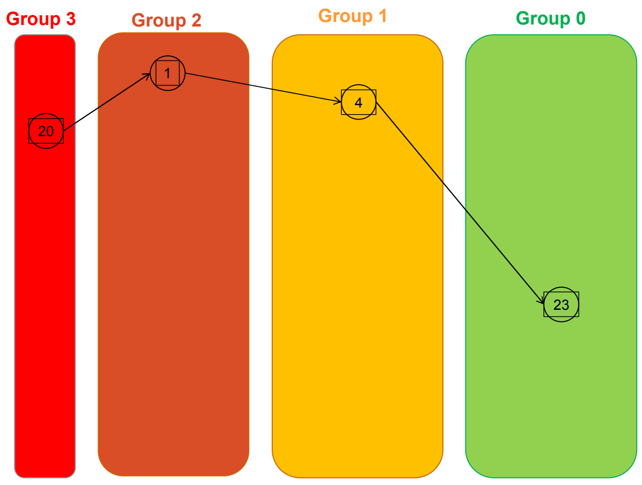

Figure 3. *The cascading settelement triggered by liquidity injection.*

Thus, every injected penny triggers the settlement process of the total value equal to the product of the rejected sum and the number of the group.

As a general rule, the accounts from the left-most group (in our case – the Third) contribute the most to the Gridlock severity, whilst the injected liquidity into the accounts from the left-most group contribute the most to the Gridlock resolution.

2.6. Meeting the Integrality Restriction

Up to this point of the BCP resolution, our strategy was ultimately optimal. That is, no other strategy may deliver better results.

However at this point we need to recall the integrality restrictions of the (6).

The proposed approach we follow is a heuristic, and therefore it is in no way an optimal strategy from the global perspective of the entire BCP, and there are known artificial instances of the BCP where the suggested approach fails to produce non-trivial result despite there exists a feasible non-trivial solution. Second, this is not the only possible heuristic that can deliver the results of the comparable quality during the comparable time of calculations. And finally, it is possible to design better heuristics for the problem. The most likely directions of the investigations would be within the areas: of integer linear programming, of NP-hard problems and of quantum computing.

The integer step of the algorithm requires that it is run in a sequence of iterations, and the graphical structure of the optimal flow (the dual problem of continuous linear program relaxation) becomes a dynamic parameter. This structure is changed on each iteration to meet the amendments, that might be produced due to the deviation from the optimal flow (on one hand) and come as a result reduction of the intensity of the flow (on the other hand).

At the start of the iteration the existing structure is analyzed to pick up the next suitable edge with strictly positive flow.

In picking the suitable edge it is essential to take precautions that there will be no need to pick this edge again at the later iterations of the integer step. Or, at least, there will be little chances to do so.

The heuristic suggests that it reasonable to select the next suitable edge from the edges, that link the left-most group (the group with highest price) with the immediately adjacent group (the group with highest price minus one). Indeed, if the precedence was arranged in apparently the opposite way, then it would mean that we would first process the flow that terminates in the group with the zero price (the cheapest nodes). But this flow is seriously dependent on the flow in the left side of the structure (more expensive nodes) which lacks flexibility. If subsequently we failed to find exact solution of the knapsack on the edge in the left (i.e. “expensive”) side of the picture, this would mean that the intensity of the flow terminating in the group with the zero price level would change, producing the need to repeat the respective knapsack search at the cheapest nodes.

Once the suitable edge is selected, the collection of the individual payments is retrieved from the database, where the ordered pair of accounts corresponds to the selected edge and to the direction of the flow. The collection is sorted in ascending manner for the payment value.

The knapsack packing subroutine (solving Sub-Set Sum problem) is called to identify those payments, that taken as a group, give the aggregate value not less than the flow along the edge, but at the same time as low as possible. Ideal situation would be, if the aggregate value of the identified group of payments is equal to the flow intensity along the edge. It might sound strange and counterintuitive, but the ideal situation is a frequent event with really high probability. Observe that the knapsack problem has the obvious solution when the intensity along the edge achieves the capacity restriction: in this case all individual payments along the edge should be selected. There are other frequent situations, when the knapsack is solved to optimality as well.

If the knapsack subroutine returns the ideal situation, then the current edge is removed from the structure and the intensity of the flow is decreased, but there is no further need to re-structure the remaining flow and we proceed with the next iteration. Otherwise the flow function and the respective structure are fine-tuned to respond to the deviation.

The iteration is complete.

Once the structure is empty (only one group of the accounts can be identified), there is no need to perform the next iteration and the entire algorithm is complete. The feasible solution to BCP is obtained. The practice and common sense judgement suggest that this solution is reasonably close to the optimal solution of the real-world instances of the BCP.

3. Result

The pivotal question is: how far is the “second-best” solution from the global optimum of the BCP? Below we bring some illumination on this question.

Assume that vector x^* is the optimal solution to BCP. We do not know this vector and it is little chance to find it as a result of the computational procedures.

However we can obtain the vector x_0 which is the optimal solution to the linear program relaxation of the BCP. This vector is non-integer, but it produces the upper bound for the Objective function of the BCP. That is,

$$F(x_0) \geq F(x^*)$$

Assume that the solution we find as a result of the heuristic algorithm is x_1. It is feasible to all BCP restrictions including integrality restriction. For that reason x_1 produces the lower bound for the Objective function of the BCP. That is,

$$F(x_0) \geq F(x^*) \geq F(x_1)$$

Every time the BCP is being solved, the upper and the lower boundaries can be easily obtained. The practice and common sense judgement suggest that the real-world instances of the BCP are prone to demonstrate the following inequality:

$$F(x_0) \leq 1{,}03 * F(x_1)$$

That is, the “second-best” solution is located within very narrow interval in terms of the Objective function. Therefore, no other computational strategy can demonstrate substantially better (more than few percentage points of the Objective function) results than that we have described above in this article.

4. Conclusion

Before this publication there was little hope to attack the Bank Clearing Problem for optimality. However the pragmatic approach illustrated here clearly shows that BCP may be successfully attacked within the well affordable time for the “second-best” solution which is hardly distinguishable from the global optimum in terms of the Objective function.

This is achieved by the proof of the equivalence of (a) the dual linear program of the LP relaxation of the BCP and (b) the MaxFlow-MinCost in the BCP residual network.

Under the liquidity shortage, the price structure of the MaxFlow-MinCost clearly pinpoints the sub-collection of the accounts which are the most efficient targets for liquidity injections from the National Central bank. The liquidity injections into the most “expensive” accounts produce the highest effect, whilst the liquidity injections into the accounts with zero price level are useless.

The Real Time Gross Settlement Systems with the liquidity saving mechanism based on the proposed algorithm are prone to demonstrate unparalleled liquidity efficiency and are safe from falling into gridlocks even under the most severe liquidity shortages ever experienced in the banking history.

References

[1] Rivandra K. Ahuja, Thomas L. Magnanti, James B. Orlin “Network Flows: Theory, Algorithms, and Applications”.

[2] Ford, L.R.; Fulkerson, D.R. (1956). "Maximal flow through a network". Canadian Journal of Mathematics. 8: 399-404. doi: 10.4153/CJM-1956-045-5

[3] Thomas Cormen, Charles Leiserson, Roland Rivest “Introduction to Algorithms”, p. 536-573.

[4] Christofides N., Graph theory: an algorithmic approach. 1975, London: Academic Press ISBN 0121743500 (ISBN 13: 9780121743505).

[5] Michael M. Güntzer, Dieter Jungnickel, Matthias Leclerc; Efficient algorithms for the clearing of interbank payments; European Journal of Operational Research 106 (1998) p. 212-219.

[6] Marco Galbiati Kimmo Soramäki. “Liquidity-saving mechanisms and bank behavior” Bank of England Working Paper № 400, July 2010.

[7] Harry Leinonen (ed.) “Simulation studies of liquidity needs, risks and efficiency in payment networks”. Proceedings from the Bank of Finland Payment and Settlement System Seminars 2005–2006.

[8] David Karger. Massachusetts Institute of Technology. Advanced Algorithms. Lecture 16: 10/11/2006. Minimum cost maximum flow, Minimum cost circulation, Cost/Capacity scaling.

[9] A. V. Goldberg and R. E. Tarjan. Finding minimum-cost circulations by canceling negative cycles. J. Assoc. Comput. Mach., 36 (4): 873-886, 1989.

[10] ´E. Tardos. A strongly polynomial minimum cost circulation algorithm. Combinatorica, 5 (3): 247-255, 1985.

[11] Shafransky Y. M., Doudkin A. A. An optimization algorithm for the clearing of interbank payments. European Journal of Operational Research, 2006, Vol. 171, 743-749.

[12] Alexander Schrijver, Theory of Linear and Integer Programming. John Wiley & sons, 1998, ISBN 0-471-98232-6 (mathematical)

[13] The simplex algorithm takes on average D steps for a cube. Borgwardt Karl-Heinz The simplex method: A probabilistic analysis. — Berlin: Springer-Verlag, 1987. — Vol. 1. — P. xii+268. — ISBN 3-540-17096-0.

[14] Kantorovich L.V., Mathematical methods for production process planning and management // Leningrad State University Press, 1939 (in Russian language).

[15] Gass Saul I., Linear Programming (Methods and Applications) Applied Science Department, International Business Machines Inc., Graduate School, U.S. Department of Agriculture, McGraw-Hill Book Company Inc., 1958.

[16] Muravyov V.I., Method of regular improvement with the alternating-size basis for the linear programing. — Collection "Operational Research and Statistical Model Methods". — Leningrad: Leningrad State University Press, 1972 (in Russian language).

[17] David Karger, 6.854 Advanced Algorithms, Minimum cost maximum flow, Minimum cost circulation, Cost/Capacity scaling, Lecture 16: 10/11/2006

[18] E. Tardos. A strongly polynomial minimum cost circulation algorithm. Combinatorica, 5 (3): 247-255, 1985.