Multi-Agent Liquidity Risk Management in an Interbank Net Settlement System

Badiâa Hedjazi[1], Mohamed Ahmed-Nacer[2], Samir Aknine[3], and Karima Benatchba[4]

[1] Information Systems Division, CERIST Research Center, 5 Rue des Frères Aissou Ben Aknoun, Algiers, Algeria
[2] Information Systems Laboratory, USTHB University,BP 32 El Alia 16111 Bab Ezzouar, Algiers, Algeria
[3] LIRIS, Université Claude Bernard Lyon,LIRIS UMR 5205INSA de Lyon, Campus de la Doua,Bâtiment Blaise Pascal,20, Avenue Albert Einstein69621 VILLEURBANNE CEDEX France
[4] ESI, National High School of Computer Science, BP 68M OUED SMAR, 16309, El-Harrach, Algiers, Algeria
{badiaa.hedjazi,benatchba}@yahoo.fr, anacer@cerist.dz, samir.aknine@univ-lyon1.fr

Abstract. A net settlement system is a payment system between banks, where a large number of transactions are accumulated, usually waiting until the end of each day to be settled through payment instruments like: wire transfers, direct debits, cheques, ... These systems also provide clearing functions to reduce interbank payments but are sometimes exposed to liquidity risks. Monitoring, and optimizing the interbank exchanges through suitable tools is useful for the proper functioning of these systems. The goal is to add to these systems an intelligent software layer integrated with the existing system for the improvement of transactions processing and consequently avoid deadlock situations, deficiencies and improve system efficiency. We model and develop by multi-agent an intelligent tracking system of the interbank exchanged transactions to optimize payments settlement and minimize liquidity risks.

Keywords: payment system, net settlement system, multi-agent system, liquidity risk, classifier system.

1 Introduction

A net settlement system is a system that processes retail payment instruments: wire transfers, direct debits, cheques, bank cards... These systems also provide clearing functions to reduce the number of interbank payments and therefore cost and consumed time[1]. The payment obligations are divided into two types: High-value payments and low-value payments. Two complementary systems treat these payments: the first is called RTGS (Real Time Gross Settlement) dealing with large amounts and in real time, and the second called net settlement system or DNS (deferred net settlement) or retail payment system which handles retail payments (cheques, direct debits, bank cards, etc.) on a “deferred net” basis[1].So far, the net settlement system plays only the role of a router in a network formed by the participating banks. Multilateral netting results has a significant reduction in payment flows and liquidity needs, compared to the bilateral netting or the gross settlement, but in the event of the insolvency of a participant, all the underlying transactions settlement of the other participants would be blocked. This creates to non-defaulting participants a *liquidity risk* [2]. Revoking payments (exclusion of failed transactions) when settlement failure, should eliminate this risk. However, risk could remain if the non-defaulting participants do not suspend payments to their defaulting customers [2]. It is necessary to have an additional system to monitor and analyze each received transaction. We model and develop an intelligent tracking system integrated with the net settlement system to minimize the risk of insolvency and liquidity, avoid deadlocks and bypass certain failures by a multilateral optimization of settlement processes through a multi-agent system (MAS) where each participant (bank) is associated with an adaptive agent. In section 2 of this article, we present some optimization models of interbank payment systems. In section 3 we describe our multi-agent balance tracking system. In section 4 we discuss the implementation and our various experiments. In section 5 we conclude with a synthesis of our contributions and give some research perspectives.

2 Related Work

FIFO mechanism is applied by all payment systems, but this mechanism becomes an obstacle for the final high-value payments settlement. Several optimization algorithms have been proposed. The *bilateral optimization* examines the participants in pairs and settles transactions simultaneously for maximum possible value. Güntzer et al [4] proposed *Greedy algorithm* to optimize the netting amount value. This algorithm is optimal for a small number of payments. Renault and Pecceu [5] improved the *Greedy algorithms* and the *Multilateral (*where all participants and all payments are considered simultaneously) *Greedy Las Vegas*. Beyeler et al. [6] proposed the addition of another source of liquidity but this entails additional costs and constraints. All these works have made significant contributions to settlement optimization but their drawback is that they assume that the participating banks behavior remains unchanged and therefore no adaptation or improvement in their decision making process. However, we were inspired by these models, by promoting operations which reduce liquidity risks in the learning model of each bank. If a bank becomes debtor, then we promote operations decreasing the debit value (credit transactions such as cheques if remitting bank or wire transfers if receiving bank).The growing complexity of payment systems requires efficient structures like MAS to their study [7] but this work is devoted to study RTGS systems and not net settlement systems but we were inspired from it to build our system according to multi-agent approach.

3 Multi-Agent Balance Tracking System

Our system is designed for intelligent tracking and processing in an interbank clearing system using a multi-agent system and classifier systems [8] [9] for the reasoning model of the agents associated to banks. It works in collaboration with the net settlement system, to make reliable decisions on transactions and prevent risks.

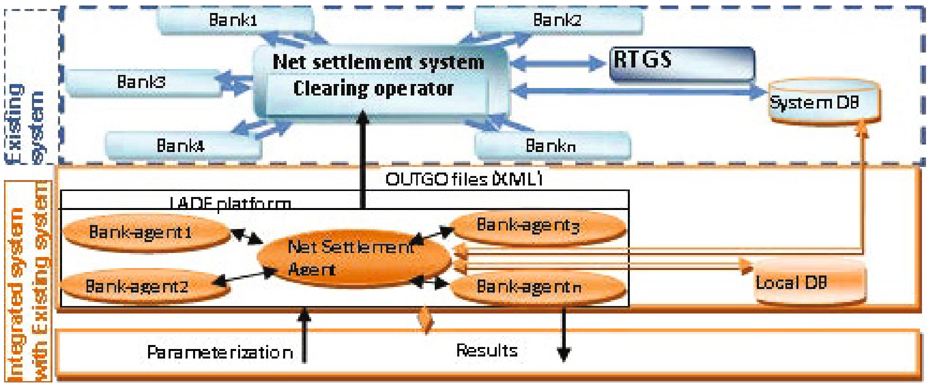

Fig. 1. System representation

Our system is integrated with the existing Net Settlement System. It is composed of a Net Settlement Agent, a set of Bank Agents (BAs) interacting between them, and two databases (DBs) (Fig. 1). *System DB* contains interbank transactions exploited by Net Settlement Agent for extracting data. The BAs exchange messages with each other and with the Net Settlement Agent. It's then a decentralized architecture.

3.1 Net Settlement Agent

Net Settlement Agent manages banks' transactions. This agent is reactive. It reacts to each received payment order and has four functions (Fig. 2): (1) Start or close a day. (2) Extract transactions from *System DB*. (3) Insert and update transactions into *Local DB*. (4) Generate OUTGO files of processed transactions in XML format.

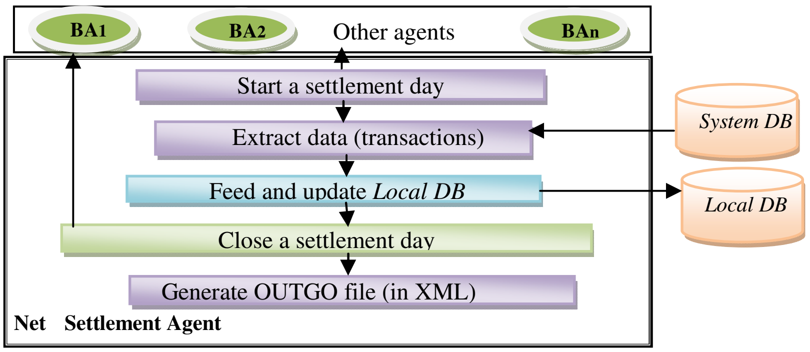

Fig. 2. Internal architecture of Net Settlement Agent

Net Settlement Agent feeds *Local DB* from *System DB* which has a specific format. Hence, the need of a working database for our system, *Local DB*. The Net Settlement agent is reactive. It reacts to each bank payment order reception.

At the beginning of a day, the Net Settlement Agent informs all the BAs of the new day and each start or close of a clearing session. It makes updates by decrementing settlement dates, giving status to balance previous states (creditor or debtor).

3.2 Banks Agents (BAs)

Each BA is associated with a participating bank. A BA is responsible of processing transactions related to it (when the remitting or receiving bank of the transaction corresponds to the BA). ABA is a cognitive agent designed to improve its behavior and profitability by learning and then based on a classifier system (CS) (Fig. 3).

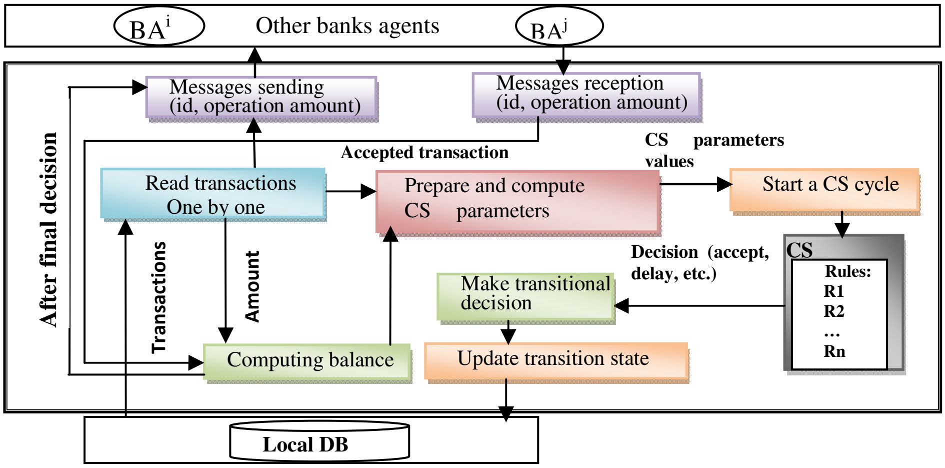
BA^i
Other banks agents
BA^j
Messages sending (id, operation amount)
Messages reception (id, operation amount)
Accepted transaction
CS parameters values
Read transactions One by one
Prepare and compute CS parameters
Start a CS cycle
After final decision
Transactions
Amount
Decision (accept, delay, etc.)
CS
Rules: R1 R2 … Rn
Make transitional decision
Computing balance
Update transition state
Local DB

Fig. 3. BA internal architecture

A BA performs the following functions:

(1) Read transactions from Local DB. (2) Prepare CS parameters (Bank State, Bank Type, Rejection Rate, etc.). (3) Start a CS Cycle including, compute Reward and make decision. (4) Compute balance after each transaction and update Local DB.

The BAs gradually adapt to their environment and learn from their past experiences with periodic evaluation of their rules. We choose then classifier systems to build their reasoning models. A classifier system is a set of rules determining agent behavior. It has a mechanism for evaluating its rules by rewarding those who produce more gains. The system starts with a random set of rules; others are generated periodically to expand the search space.

BA Classifier System (CS). A BA CS evaluates bank state, transaction type, bank type, bank threshold, calculates parameters and make decision. Each CS rule consists of 3 parts (**condition**: on 21 bits, **action**: on 2 bits, **fitness**: a real number) (Fig. 4).

Condition: it consists of two parts:
Case: corresponds to encountered cases by a bank during a session.
Parameters: contains seven parameters validating or not the *Case* part.

Action: shows the four possibilities of the agent action (decision):
(1)***Accept*****:** perform transaction. (2) ***Delay*****:** delay transaction to the next session. (3)***Reject*****:** Reject transaction but will be processed next day. (4) ***Cancel*****:** Cancel transaction, with the possibility of treatment in the next day.

Fitness: contains the strength of the rule and is a real in the interval [0, 1].

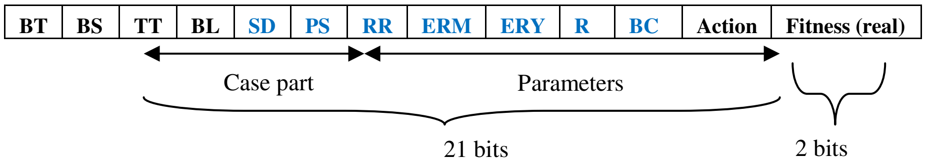

Fig. 4. CS rule representation

BT (Bank type): Coded on 1 bit. If *remitter bank* BT=*0* else (*receiver*) BT=1
BS (Bank state): When starting our system all BAs have 'creditor' state. This state changes after interbank exchanges. BS is calculated as in equation (1):

$$BS = C\text{-}D \tag{1}$$

where: C = ∑ amounts of credit payment instruments.
D = ∑ amounts of debit payment instruments.
BS is coded on 1 bit. If C-D>0 then the bank is *Creditor*: *0* else is *Debtor*: *1*.
TT (Transaction Type): Instrument type and coded on 4 bits as for example: *Cheque* coded *0000* and *wire transfer* coded *1001*.
BL (Bank limit): Coded on 2 bits (Table 1) and calculated as in (2)**:**

$$BL = (C\text{-}D)/X \tag{2}$$

where: C = ∑ amounts of credit payment instruments of the bank.
D = ∑amounts of debit payment instruments of the bank.
X =80% of the initial balance of the bank.
We are interested only by negative values of BL (the debtor banks).

Table 1. Binary representation of BL

Interval	Degree	Meaning	Code
[0, 0.4 [	mild	Delaying debit transactions	00
[0.4, 0.7 [	moderate	Sorting transactions to accept (Sort)	01
[0.7, 0.9 [	severe	Sending alarms to participants and Sort	10
>0.9	Very severe	Cancelling debit transactions and Sort	11

SD (Settlement Date): coded on 2 bits where D is the trading day and D1, D2 are the following days (D: *00*, D*1*: *01*, D2: *10*).
PS (Previous State): is the final state of the previous day after balance settlement, and is coded on 1 bit: (*Creditor*: *0*; *Debtor*: *1*).

RR (Rejection Rate): is the rate of rejected transactions by the number of transactions. It is coded on 2 bits and calculated as in equation (3):

$$\text{RR} = (\textit{Rejected}/\ (\textit{Processed} - \textit{Rejected})) \tag{3}$$

where: *Rejected:* number of rejected transactions.
Processed: number of all the transactions to be processed.
If **RR** in [0, 0.4 [then coded 00; If **RR** in [0.4, 0.7[then coded 01;
If **RR** in [0.7, 0.9[then coded 10; if **RR** in [0.9, 1] then coded 11;

ERM (Evolution rate per month): Coded on 2 bits, is amounts evolution rate between 2 consecutive months and calculated as in (4):

$$\text{ERM} = ((\text{N-M})/\ \text{M})*100 \tag{4}$$

where: N: the value of transactions of the current month.
M: the value of transactions for the previous month.
If **ERM** <0 then coded 00; If **ERM** in [0,50[then coded 01;
If **ERM** in [50,100[then coded 10; if **ERM** >=100 then coded 11;

ERY (Evolution rate per Year): coded on 2 bits; compares amounts of a month of current year with the same of previous year. Is calculated as in (5):

$$\text{ERY} = (\text{N}/\ \text{M})*100 \tag{5}$$

where: N: total amount of all transactions processed in the current month.
M: total amount of processed transactions of same month the previous year.
If **ERY** <0 then coded 00; If **ERY** in [0,50[then coded 01;
If **ERY** in [50,100[then coded 10; if **ERY** >=100 then coded 11;

R (Ratio): Coded on 2 bits, reflecting position of a payment instrument from all payment instruments, and calculated by the formula (6):

$$\text{R} = \text{A}/\ \text{T} \tag{6}$$

where : A: the total amount of the transactions by a payment instrument.
T: the total amount of the instruments processed by a bank.
If **Ratio** in [0, 0.4 [then coded 00; If **Ratio** in [0.4, 0.7[then coded 01;
If **Ratio** in [0.7, 0.9[then coded 10; if **Ratio** in [0.9, 1] then coded 11;

BC (Bank category): coded on 2 bits (*Large:00*; *Medium: 01; Small: 10*).

Action: Agent's decision; coded on 2 bits for *Accept*, *Delay*, *Reject*, *Cancel.* Condition parameters weights depend on Action value (Tables 2, 3, 4, 5):

Table 2. Parameters weights when action is « Accept »

Parameters	Weight	Significance
SD, BC	3	Important
PS, RR, ERM, ERY	2	Moderately important
R	1	Little importance

Table 3. Parameters weights when action is « Delay »

Parameters	Weight	Significance
SD, PS, RR, ERM, ERY, R	2	Moderately important
BC	3	Important

Table 4. Parameters weights when action is « Reject »

Parameters	Weight	Significance
SD, R	1	Little importance
PS, ERM, ERY	2	Moderately important
RR	4	Very important
BC	3	Important

Table 5. Parameters weights when action is « Cancel »

Parameters	Weight	Significance
SD, PS	1	Little importance
RR	4	Very important
ERM, ERY, R	2	Moderately important
BC	3	Important

For the « Case part » of the condition we have the following possibilities:
Where: *RC* is for receiving bank, *RM* for remitting, *C*: Creditor and *D*: debtor.
We consider: wire transfer as *Debit_instrument*;
card, direct debit, cheque, negotiable instrument as *Credit_instrument*.

1. **If** (BT = *RM &* TT = *Credit_instrument*) ***or*** (BT = *RC &* TT = *Debit_instrument*) ***then*** *accept*.
2. **If** (BT = *RM &* TT = *Debit_instrument &* EB = *C*) ***or*** (BT = *RC &* TT = *Credit_instrument &* BS = *C*) ***then accept.***
3. **If**(BT = *RM &* TT = *Debit_instrument &* EB = *D &* BL є [0,0.4[ ) ***or*** (BT = *RC &* TT = *Credit_instrument &* BS = *D &* BL є [0, 0,4[ ) ***then*** *delay*.
4. **If** (BT = *RM &* TT = *Debit_instrument* & EB = *D &* BL є [0.4, 0.7[ ) ***or*** (BT = RC & TT = Credit_instrument) & BS = D *&* BL є [0.4, 0.7[ ) ***then reject.***
5. **If** (BT = *RM* & TT = *Debit_instrument* & BS = D & (BL>0.7) ***or*** (BT = RC & TT = Credit_instrument & BS = D & (BL>0.7) ***then*** cancel.

Rules Reward: The *Case par*t of a classifier is the most important. We associate it with the weight *5/8* of the decision. The *Parameters part* classifier's condition affects less the decision and is associated with the weight *3/8*. If one of the two parts is rewarded then it is multiplied by 1and if it is less rewarded then it is multiplied by *1/4*.We limit the value of the reward (*RW*) in [0, 1] by dividing it by 2.

```
If action part of Cl (Classifier to evaluate) is « accept »
If (BT=RM & TT=Credit_instrument) or (BT=RC &
TT=Debit_instrument)
Then
RW=(5/8+3/8×[(3×SD)+(2×PS)+(2×RR)+(2×(1/ERM))+(2×(1/ERY))+
(3×BC)+(1×R)])/(2×Σweights)
1/ERM and 1/ERY is user to limit the value in [0, 1].
If (BT=RM & TT=Debit_instrument & BS=C) or (BT=RC & TT= Cre-
dit_instrument & BS=C)
Then
RW=(5/8+3/8×[(3×SD)+(2×PS)+(2×RR)+(2×(1/ERM))+(2×(1/ERY))+
(3×BC)+ (1×R)])/(2×Σweights)
```

```
Else
RW=(5/8×1/4+3/8×[(3×SD)+(2×PS)+(2×RR)+(2×(1/ERM))+(2×(1/ERY))+(3
×BC)+ (1×R)])/(2×Σweights)
If action part of Cl is « Delay »
If (BT=RM & TT=Debit_instrument & BS=D & BL∈[0,0.4[) or (BT=RC &
TT=Credit_instrument & BS=D & BL∈[0, 0.4[)
Then
RW=(5/8+3/8×[(2×SD)+(2×PS)+(2×RR)+(2×(1/ERM))+(2× (1/ERY))   +
(3×BC)+ (2×R)])/(2×Σweights)
Else
RW=(5/8×1/4+3/8 × [(2×SD)+(2×PS)+(2×RR)+(2×(1/ERM))+
(2×(1/ERY))+(3×BC)+(2×R)] ) /(2×Σweights)
If action part of Cl is « reject »
If (BT=RM & TT=Debit_instrument & BS=D & BL∈[0.4,0.7[) or BT=RC
& TT=Credit_instrument) & BS=D & BL∈[0.4, 0.7[)
Then
RW=(5/8+3/8×[(1×SD)+(2×PS)+(4×RR)+(2×(1/ERM))+ (2×(1/ERY))
+(3×BC)+(1×R)])/(2×Σweights)
Else
RW=(5/8×1/4+3/8×[(1×SD)+(2×PS)+(4×RR)+(2×(1/ERM))+(2×(1/ERY))
+(3×BC)+(1×R)])/(2×Σweights)
If action part of Cl is « Cancel »
If (BT=RM & TT=Debit_instrument & BS=D & (BL> 0.7) or (BT=RC &
TT=Credit_instrument) & BS=D & (BL>0.7))
Then  RW=(5/8+3/8×[(1×SD)+(1×PS)+(1×RR)+(2×(1/ERM))+(2×(1/ERY))+
(3×BC)+(2×R)])/(2×Σweights)
Else
RW=(5/8×1/4+3/8×[(1×SD)+(1×PS)+(1×RR)+(2×(1/ERM))+(2×(1/ERY))+
(3×BC)+(2×R)])/(2×Σweights)
```

3.3 Transactions Processing and Decision Making

A settlement day is composed of three clearing sessions.

1st session: This session deals with all the new arrival transactions on D-Day and the recovery and processing of already included transactions in *Local DB* of previous days (D-1, etc.) with one of these statements: (1) Transaction's transitional state: *delay*, *reject* or *cancel.* (2) Transitional state *accept*, but acceptance or rejection not confirmed by the receiving bank. (3) Final state *accept*, but their settlement dates not yet reached.

2nd session: Resumes only transactions with *delay* transitional state and processes the new transactions.

3rd session: The same as the 2nd one but updating *Local DB* before closing.

At the end of each session is generated an OUTGO file of processed transactions. Net Settlement Agent extracts all the transactions from *System DB* and copies them into *Local DB*. BA processes each transaction with the remitting bank is this BA.

If the transaction is not yet processed, then process it, generate a transitional decision (state) and update *Local DB*, pending the final state sent by the bank when transitional state is *accept*;

If transaction already processed, the bank final decision is needed. If transaction not rejected, then finished else removed and balance recalculated.

At the end of each clearing session, the system updates the final statement in *Local DB* and generates an OUTGO file in XML format.

Transactions such as *wire transfers* have negative impact on remitting banks than receiving ones. We therefore give decision priority to remitting banks. For the other payment instruments: *card*, *direct debit*, *cheque*, *negotiable instrument,* we give priority to receiving banks. If a *receiving BA* (or *remitting*) processes transactions and results to transitional state *accept* then it informs the *remitting BA* (or *receiving*) to update balances. If the final state is *reject* then the concerned BA updates its balance.

4 Implementation and Experiments

Our system is implemented with JADE multi-agent platform, JAVA, ORACLE DBMS, ART(Artificial Reasoning Toolkit) package for programming CS and DOM for generating OUTGO files. A BA or the Net Settlement Agent can execute several behaviors (specific tasks: ex. *New_settlement_day()*) concurrently (Fig.5).

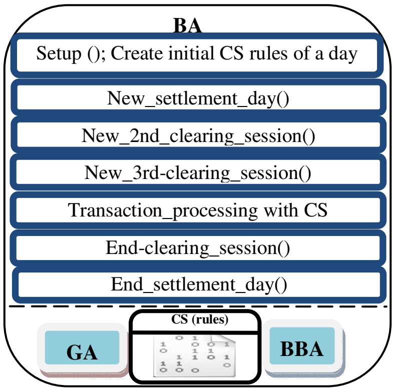
BA
Setup (); Create initial CS rules of a day
New_settlement_day()
New_2nd_clearing_session()
New_3rd-clearing_session()
Transaction_processing with CS
End-clearing_session()
End_settlement_day()
GA
CS (rules)
BBA

Fig. 5. A BA principal behaviors

At the end of each clearing session the Net Settlement agent generates an OUTGO file in XML format. Each processed transaction contains the decisions of its final or transitional states. These decisions are generated by their corresponding agents. For example the transaction *83* has the transitional decision *accept* and is generated by the *bank agent 3*(BA_3) corresponding to the remitting bank number *3*.

The results of our experiments allow us to judge the performance of our system compared to the current system. Our experiments are made with five banks (Bank 1, Bank 2, Bank 3, Bank 4, and Bank 5). The same operations initially with random amounts are considered for both types of experiments. The first type simulates the current system that accepts all operations and the second type that processes transactions by treating them with our multi-agent system. The graph in Fig.6 are calculated without prior treatment, that is to say, all transactions are accepted. Also, Banks often reach the limit value (BL). This puts them in situations of liquidity risks.

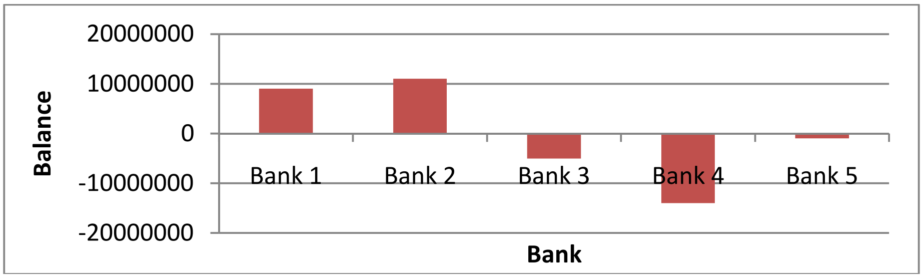

Fig. 6. Virtual balance evolution in the current system for 5 banks / Day

The graph in Fig.7 is generated at the end of a settlement day. It represents the virtual balance evolution achieved in each bank with our system.

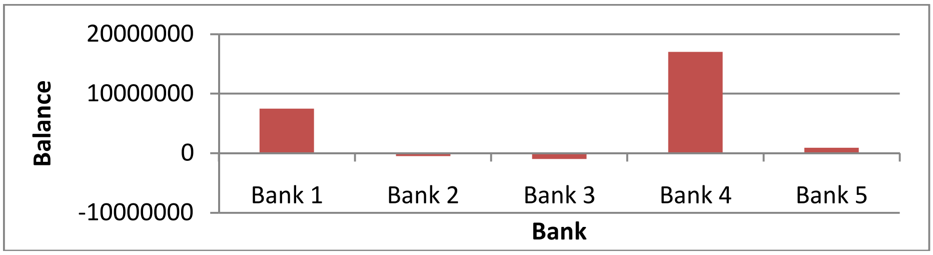

Fig. 7. Virtual balance evolution in our system for 5 banks / Day

The purpose of these two graphs is to compare the effectiveness of the old system compared to ours and to judge the performance of our system. On Fig. 6 and 7 we note that 3 banks (Bank 3, Bank 4, Bank 5) are in high amounts of debtor state (~ -14000000) with the existing system (current) against 2 banks that are in small debtor state amounts (~ -900 000) with our system. By doubling the number of banks in our simulations (10 banks) we noticed that our system is more efficient because all the banks find themselves in a creditor state at the end of the settlement day (Fig. 9). This is explained by the increased number of liquidity sources in the system. This is not the case with the current system even if the number of banks doubles (Fig. 8).

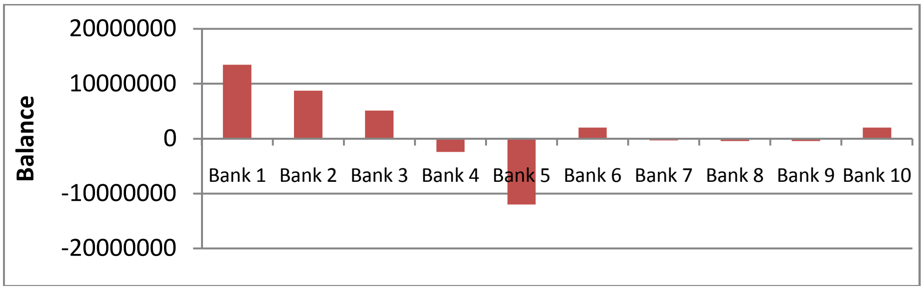

Fig. 8. Virtual balance evolution in the current system for 10 banks / Day

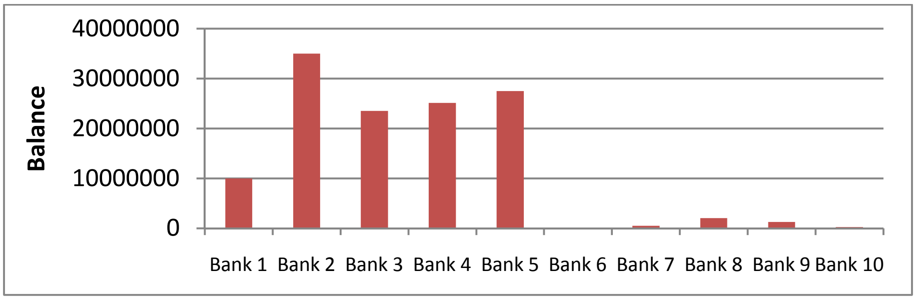

Fig. 9. Virtual balance evolution in our system for 10 banks / Day

Fig. 10 shows the evolution of virtual balances through ten consecutive settlement days. This graph tries to target vulnerabilities in the current system by showing the negative development of some banks, which can cause situations of insolvency, if the problem persists. The system does not matter the number of banks with a negative balance, and gives no warning. Therefore, the balances will evolve, even if it is in debtor state, and this will certainly worsen the financial situation of banks.

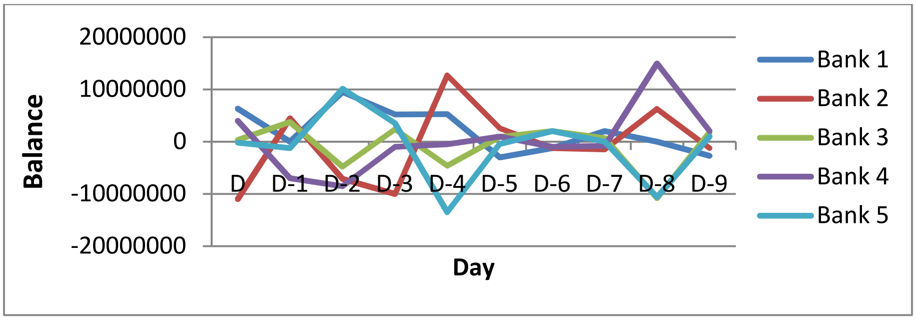

Fig. 10. Virtual balance evolution in the current system for 9 days

Fig. 11 shows the evolution of virtual balances through a period of ten settlement days. This graph shows the improvements made by our system to treatment of balances. In this graph, we note that our system has a positive impact on balances by reducing the margins between negative and positive balances and consequently maintaining stability of the clearing system.

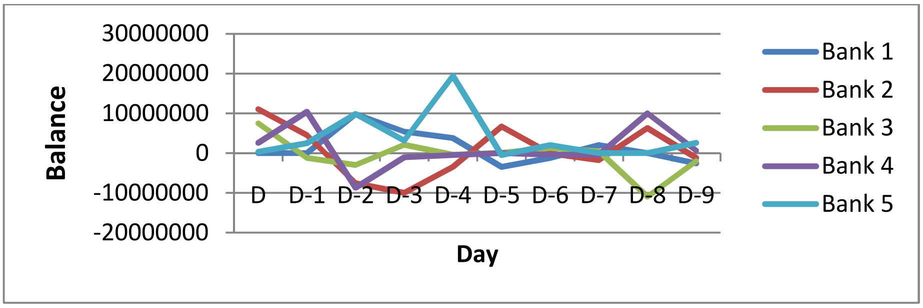

Fig. 11. Virtual balance evolution in our system for 9 days

Our system allows running the entire daily payment processing (three clearing sessions) in each settlement day. The results show that relatively to the existing system, our system allows a significant improvement in the bank balances evolution and significantly minimizes the number of times when banks are in debtor making the payment system less exposed to liquidity risk.

5 Conclusion

Our system is designed as a response to problems posed by the risks of participants failure or insolvency situations in the interbank clearing system, which requires the inversion of the clearing day (removal). This involves the recalculation of clearing balances of the non-defaulting participants. Improving balances management is by a multi-agent decision support system is necessary to reduce liquidity risks. Our current system makes the system more flexible and adaptive by detecting risky transactions and processing them by minimizing liquidity risks. The obtained results show that multi-agent models can be used to better manage the system and resolve payment system major problem which the liquidity risk. Our system fully meets the original goals but some improvements can make our system more flexible such as (1) creating direct interface with the RTGS system in order to manipulate actual balances and (2) adding predicting agents that probe the history data automatically and intelligently to forecast the future evolution of the system.

References

1. Taibi, A.: BDL Système de télécompensation, rapport de banque de la BDL (Mai 2006)
2. Banque de France, Revue de la stabilité financière (3) (Novembre 2003)
3. Atos Euronext/Diamis, Rapport de Conception. Principes Fonctionnels pour la mise en œuvre dansle système d'information, Groupe Atos Origin. Version 2.0. 18. (Juin 2004)
4. Güntzer, M., Jungnickel, D., Leclerc, M.: Efficient algorithms for the clearing of interbank payments. European Journal of Operational Research 106, 212–219 (1998)
5. Renault, F., Pecceu, J.-B.: From PNS to TARGET2: the cost of FIFO in RTGS payment system. In: Leinonen, H. (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, 320 p. (2006) ISBN 978-952-462-360-5
6. Beyeler, W., Bech, M., Glass, R., Soramäki, K.: Congestion and Cascades in Payment Systems. Physica A: Statistical Mechanics and its Applications 384(2), 693–718 (2007)
7. Galbiati, M., Soramaki, K.: An agent-based model of payment systems. Journal of Economic Dynamics and Control 35(6), 859–875 (2011)
8. Livolant, E.: Apprentissage Multi-Agents par Systèmes de Classeurs, Université de Caen, Laboratoire GREYC, DEA Intelligence Artificielle et Algorithmique (Septembre 2003)
9. Holland, J.H.: Genetic algorithms and classifier systems: Foundations and future directions. In: Refenstette, J.J. (ed.) Proceedings of the Second International Conference on Genetic Algorithms and Their Applications, pp. 82–89. Lawrence Erlbaum Associates, Hillsdale (1987)