Visual Exploration of Cash Flow Chains

Jerzy Korczak, Walter Łuszczyk
Uniwersytet Ekonomiczny
ul. Komandorska 118/120
53-345 Wrocław, Poland
Email: jerzy.korczak@ue.wroc.pl

Abstract—**The paper proposes a new method for interactive visual exploration of the chains of financial transactions, assisting an analyst in the detection of money laundering operations. The method mainly concerns searching, displaying and annotating selected groups of transactions from a database. We show how one can programmatically and interactively reduce the volume of the chains surveyed and limit the analysis to the most suspicious transactions. In order to improve readability of the transaction graph, an evolution-based algorithm has been designed to optimize its visual representation. The system is verified on the real-life database of financial transactions. The experiments conducted have shown that allowing visual exploration, one can accelerate the search process and enrich the data analysis.**

I. Introduction

MONEY laundering denotes the financial operations aimed at putting into circulation money from illegal sources, mainly from criminal activities. Detection of the transactions involved in the process of money laundering is extremely difficult and complex. This follows on the one side from the large amount of data that must be examined and, on the other from the difficulty of distinguishing the ordinary transactions from suspicious ones. Many systems of recording and monitoring transactions, type AML (Anti-Money Laundering), use predetermined rules to detect suspicious transactions [1]; [6]; [10]; [11]. However, despite the commitment of huge resources, the effectiveness of current solutions is very low. In general, the percentage of verified transactions as indicated by the system of suspicious transactions, measured by the index of TPR (True Positive Rate), is very small. For example, one of the largest financial institutions in the Balkans, after the purchase and implementation of software - some of the newest and most expensive software against money laundering - scored TPR equal to 0.02%. This result may seem daunting, but, as practice shows, the majority of institutions start with this level of accuracy and, after years of fine-tuning the system and finding new patterns, TPR may reach about 5% [4] Government bodies involved in the issuance of regulations recommend to developers that the system should reach TPR ranged from 4% to 7% [4]; [6].

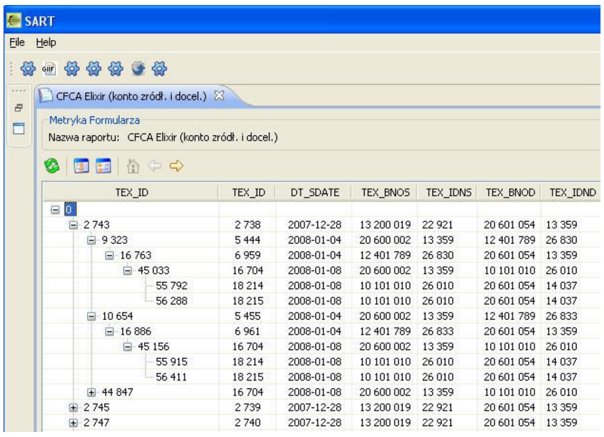
This work has been performed in the framework of a research project on methods of mining anti-money laundering data, conducted by the Center for Intelligent Information Technology in Management of Wrocław University of Economics, Poland.

In this paper, we propose an interactive method for visual data mining, assisting an analyst in the detection of illegal transactions. Support consists mainly of visualization and annotation of some of the operations in a graph showing the money flow chains between accounts.

In real-world applications, the number of accounts and transactions is huge; therefore even a partial graph visualization poses problems of feasibility, readability and interpretation of financial operations. In order to assist the analyst to interactive interrogation of the database, we have introduced the functionality of defining SQL queries and graphical operations on the graph of transactions resulting in a significant reduction in the search space. We have also provided a number of editing and graph visualization features, including heuristics to visualize sequences of transactions, graphical aggregation of transactions corresponding to the same account, and the transaction graph optimization by evolutionary algorithm.

The fundamental characteristic which determines the ease of interpretation and pattern matching operations is the representation of the transactions. Typically in financial information systems, transactions are represented in a tabular form as shown in fig. 1, a report of program cash flow chains between accounts taken from the SART system (System for Analysis and Registration of Transactions, developed by TETA SA company) [12]. We see that the presentation of a large number of transactions is barely legible. The report contains information on only a few dozen transactions and yet these are difficult to display on a computer screen. An additional difficulty in visualizing the chain of transactions using the tables and data sheets is heterogeneity. For example, an account can have several incoming and dozens or even hundreds of outbound transactions. In such cases the transactions recorded in the traditional table are hardly legible and are cumbersome to handle because the analyst must often "scroll" the report.

The paper is organized as follows. The next section will show how one can interactively reduce the volume of the analyzed chains, namely how one can view in place of all the data only the most important transactions.



Nazwa raportu: CFCA Elixir (konto źródł. i docel.)

TEX_ID	TEX_ID	DT_SDATE	TEX_BNOS	TEX_IDNS	TEX_BNOD	TEX_IDND
0						
2 743	2 738	2007-12-28	13 200 019	22 921	20 601 054	13 359
9 323	5 444	2008-01-04	20 600 002	13 359	12 401 789	26 830
16 763	6 959	2008-01-04	12 401 789	26 830	20 601 054	13 359
45 033	16 704	2008-01-08	20 600 002	13 359	10 101 010	26 010
55 792	18 214	2008-01-08	10 101 010	26 010	20 601 054	14 037
56 288	18 215	2008-01-08	10 101 010	26 010	20 601 054	14 037
10 654	5 455	2008-01-04	20 600 002	13 359	12 401 789	26 833
16 886	6 961	2008-01-04	12 401 789	26 833	20 601 054	13 359
45 156	16 704	2008-01-08	20 600 002	13 359	10 101 010	26 010
55 915	18 214	2008-01-08	10 101 010	26 010	20 601 054	14 037
56 411	18 215	2008-01-08	10 101 010	26 010	20 601 054	14 037
44 847	16 704	2008-01-08	20 600 002	13 359	10 101 010	26 010
2 745	2 739	2007-12-28	13 200 019	22 921	20 601 054	13 359
2 747	2 740	2007-12-28	13 200 019	22 921	20 601 054	13 359

Fig. 1. Tabular representation of chains of transactions

Then we show a simple, intuitive way in which one can find more detailed information. We will present a solution that optimizes the visual representation of transaction chains enabling easier analysis of graphic schemes, accelerating the search process, and enriching the functionality of the system. For example, by marking the selected transactions as suspicious, one can try to learn the system to recognize patterns of money laundering operations.

More details about the database structures in AML systems may be found in [7], [8], [9].

II. REPRESENTATION OF TRANSACTION CHAINS

The main concern while designing a visualization algorithm of sequential operations is the complexity of the resulting graph. In the project, the main measure of assessment of the graph readability was the number of intersections of edges connecting vertices of the graph: the fewer the edges, the better the resulting graph. If the graph contains n accounts there could occur approximately $n^2/2$ edges between them [13]. Let us note that in practice some of the edges can be repeated. It follows that the complexity of such a graph is $O(n^2)$ [2], [3]. Fig.2 presents an example of a graphical representation of the transaction chains. The bold edges indicate that there are many transactions between linked accounts. Red color means that the edge, or at least one of multiple edges, represents a suspicious transaction.

To effectively minimize the number of cuts generated in the initial graph, the whole transformation process was divided into two stages. In the first the simple heuristics have been applied such as the aggregation of multiple edges, the clustering of accounts by type, the linking accounts with the same neighbor in the graph, and the displacement at the bottom of the graph the accounts that are related to transactions with one another account. The graph thus prepared leads to the second stage of the optimization process, which aims to reduce the number of edge intersections and thus improves the readability of the graph.

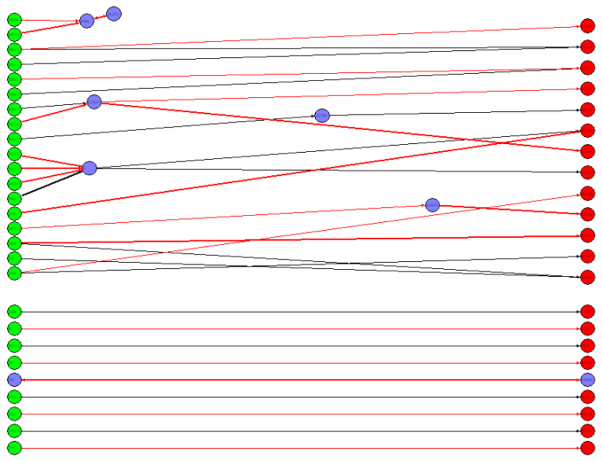


Fig. 2 Graph of transaction chains obtained by application

Let us now proceed to describe the evolutionary algorithm that minimizes the number of intersections of edges in the graph. The task of the algorithm is to find such repartition of vertices on the plane that the number of intersections of edges of the graph is as small as possible (see the specification of Algorithm 1).

Algorithm 1. Minimization of the number of edge intersections

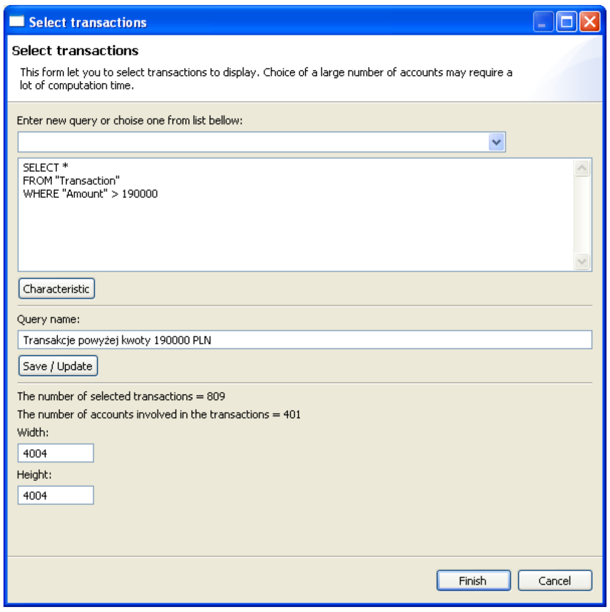
```
C = 5000;              // number of iterations without changes to
                          stop evolution
MutSrcPr = 0.45;       // probability of source account mutation
MutInterPr = 0.1;      // probability of intermediate account mutation
MutDstPr = 0.45;       // probability of target account mutation
Fmin = infinity;       // minimal number of intersections
F = infinity;          // minimal number of intersections of individual in
                          one trial
FTemp = infinity;      // number of intersections in currently mutated
                          individuals
IBest = NULL;          // the best individual
ITemp = NULL;          // temporary individual, similar for all trials
I = NULL;              // evolved individual in the trial
Imut = NULL;           // mutated individual in a given iteration
N = 3;                 // number of tested individuals
M = 3;                 // number of mutations of a given individual
FOR i = 1 TO N DO          // number of tested individuals

BEGIN
  ITemp = initiated new graph of transactions();
  FOR j = 1 TO M DO       // number of mutation of an individual
    I = ITemp;
    DO
    Imut = I;
      IF RAND() < MutSrcPr THEN Perform Imut mutation of source
         accounts;
      IF RAND() < MutInterPr THEN Perform Imut mutation of
         interm. accounts;
      IF RAND() < MutDstPr THEN Perform Imut mutation of target
         accounts;
      FTemp = compute a number of intersections in Imut;
    IF FTemp > F THEN
    BEGIN
      F = FTemp;
      I = Imut;
    END
    WHILE (F doesn't change in C last iterations)
    IF F < Fmin THEN
    BEGIN
      Fmin = F;
      IBest = I;
    END
  END
END
```

To explain the algorithm let us start by describing the process of evolution. The outer loop FOR algorithm indicates how many individuals will be considered in the initial population. Since the algorithm does not use crossover operator, the evolution of population is equivalent to the evolution of each of the individuals separately. The inner loop FOR controls the number of mutation attempts on the same individual. The algorithm may fall into a local minimum of the objective function, therefore repeating the evolution several times from the same starting point greatly reduces these cases.

The main part of the evolutionary algorithm is the most nested DO-WHILE loop. Mutations are performed until the number of iterations reaches *C* or the solution is not improving. We have proposed three types of mutation: displacement of source, intermediate and target nodes. Displacement of the source nodes changes the headings of two randomly selected groups of vertices (there may be a group containing one vertex). Here we use the heuristics that when the order of accounts in the group does not change, it is therefore permissible to move only the entire group. A similar mutation relates to the target accounts. Mutation of intermediate nodes is drawing the new position in the permitted area between the source and target accounts.

The second important feature concerns the user interaction related to the definition of the scope of data investigation. This is done in two ways. The first solution is the ability to filter transactions that are selected from a database by SQL query. The second way is a direct manipulation on the displayed graph of chains of transactions. For large databases, sequential viewing and analysis of all transactions would be very time-consuming or even impossible. To partially overcome these difficulties, the two phase commit was implemented, namely a display of the transactions characteristics that meet the search criteria; and acceptance by the analyst to draw the graph.

In the example in fig.3 the analyst launched a simple query searching for the transactions with the amount of more than 190 thousand zł. From the computing, *Characteristics*, we see that there are 809 such transactions involving 401 source, intermediate, and target accounts, and the area calculated for visualization is 4004 x 4004 pixels. This form lets the analyst select whether to view the graph, or modify the query to reduce or extend the set of transactions. The graph is displayed after optimization using an evolutionary algorithm


Select transactions

Select transactions

This form let you to select transactions to display. Choice of a large number of accounts may require a lot of computation time.

Enter new query or choise one from list bellow:

```
SELECT *
FROM "Transaction"
WHERE "Amount" > 190000
```

Query name:

Transakcje powyżej kwoty 190000 PLN

The number of selected transactions = 809

The number of accounts involved in the transactions = 401

Width:

4004

Height:

4004

Fig. 3 Two-phase commit - the appearance of the window after query execution

In some cases the analyst may optionally re-arrange the graph manually. An example of the moving operation of the account ID 5169 is shown in fig.4.

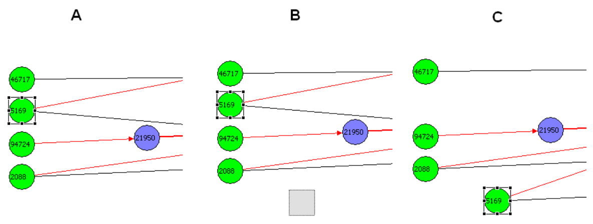


Fig.4. Moving a node: A - selection, B - dragging, C - a new position

Another operation is to remove vertices of the graph. This may happen when, after SQL filtering, the graph contains either too many accounts or accounts which have proved during the analysis to be uninteresting. To obtain a more readable graph, the analyst may then remove them manually. Let us note that the accounts are not removed physically from the database, but only from the currently displayed diagram. The steps of account deletion are shown in fig. 5.

An important operation is the interactive marking of transactions as suspicious. With this functionality, an analyst or pattern recognition program may mark the transaction as suspicious [cf. TABLE I]. Information about the suspicious transaction is stored in a database and can be used in subsequent studies.

In the system many other useful features have been designed, such as memorizing a query filtering transactions, zooming graph, *Undo* and *Redo* functions, and saving a trace

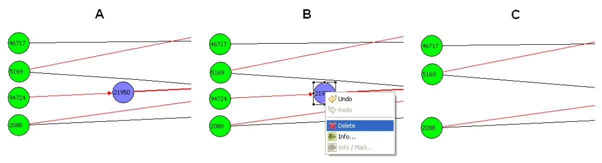


Fig.5. Remove a node: A - the initial situation, B – selection from the menu, *Delete*, C - after removal of the top figure

operations log. A detailed description of these functions along with examples is given in [9].

III. DESCRIPTION OF EXPERIMENTS

This section contains descriptions of two experiments, namely: 1) advanced retrieval of database transactions, and 2) optimization of the graph using an evolutionary algorithm. In the experiments, the real-life data obtained from one of the Polish banks are stored in a relational database PostgreSQL 8.3.

The experimental database consists of three tables: *Transaction, Query* and *Trace*. The first one contains the input data in the form of transaction descriptions (TABLE I). TransId column is a unique key serving to identify unambiguously the transaction. The next two columns contain information about the accounts involved in the transaction: SrcAcctId is the identifier of the account from which money is transferred, and DstAccId indicates a debited account. In this example, identifiers are represented by small integers, but in real applications, the account identifier may be the IBAN or NRB. The Date and Amount columns contain the date of execution and the transfer amount, expressed in PLN. The Suspected column indicates whether the transaction is suspicious in terms of money laundering, and may be modified by the system.

TABLE I. EXAMPLE OF TRANSACTION DESCRIPTIONS

TransId	SrcAccId	DstAccId	Date	Amount	Sus-pected
677	21561	22924	01-02	73 095,67	
678	21561	22924	01-02	88 142,15	
2888	24756	16579	01-02	67 684,03	
730	558	22971	01-02	66 495,43	
1821	19202	23934	01-02	57 669,16	
1823	18677	22839	01-02	80 000,00	
1933	2135	24037	01-02	180 000,00	
1925	21561	24032	01-02	74 657,08	
1926	21561	24033	01-02	76 910,61	
1927	21561	24033	01-02	109 671,77	
1928	21561	24032	01-02	134 746,56	

Other tables are *Query* and *Trace*. The first of these tables holds query requests through which data about the transactions are loaded. The second, the *Trace* table, stores the user actions on the displayed graph of chains of transactions. Data for application can be obtained from any bank's financial system which offers the functionality of transaction registration and thus at least partially satisfies the requirements of the legislature. If the data have a different structure than that shown in TABLE I, any tool of the class ETL (Extract, Transform, Load) can be used for importing data.

The system runs on PC computers, currently on the two leading operating systems, Windows and UNIX, using the popular database management system Postgres. To meet the requirement to run on different systems, the system was implemented using Java environment and a library Eclipse RCP GEF for graphics editing. Eclipse RCP (called the Eclipse Rich Client Platform) is a library which gives rise to the creation of rich graphical interfaces. To implement the system a so-called fat client architecture (called thick client) has been used. The program is executed directly on the user's computer called the client, and the data is stored on the server side.

Experiment 1. Advanced database retrieval – queries

In this experiment, we assume the role of the banking analyst who analyzes the context of transactions to detect money laundering operations.

Suppose that an analyst is interested in transactions having the most frequent accounts, that is, accounts which participate in the largest number of transactions during a specified period. For example, assume that the concerns are related to transactions in January 2011, involving a top 10 the most frequent accounts in the database. This task can be written in the form of the following SQL query:

```
SELECT *
FROM "Transaction"
WHERE ("Date" BETWEEN '2011-01-01' AND '2011-01-31')
        AND ("SrcAccId" IN
(
  SELECT "AccountId"
  FROM
  (
  SELECT "SrcAccId" AS "AccountId" FROM "Transaction"
     UNION ALL
     SELECT "DstAccId" AS "AccountId" FROM "Transaction"
  )
  GROUP BY "AccountId"
  ORDER BY COUNT(*) DESC
  LIMIT 10
 )
 OR "DstAccId" IN
 (
  SELECT "AccId"
  FROM
  (
     SELECT "SrcAccId" AS "AccountId" FROM "Transaction"
     UNION ALL
     SELECT "DstAccId" AS "AccountId" FROM "Transaction"
  )
  GROUP BY "AccountId"
  ORDER BY COUNT(*) DESC
  LIMIT 10
   ) )
```

Following the query definition, the system shows the characteristics of the resulting graph (Fig. 6), where the number of accounts in the graph is 140 and the number of transactions 306. The analyst can visualize the transactions chains because the volume seems feasible to display. This experiment also illustrates an advantage over traditional reports. Data having 140 accounts and 306 transactions could be displayed on one page. However, this visualization does not show all relevant data, such as account numbers, amounts and dates of transactions; but the analyst can easily identify the main streams of cash flow. If necessary, by using the graph zooming, or by clicking on particular objects, he can very quickly get more information. The resulting graph is the subject of further research.

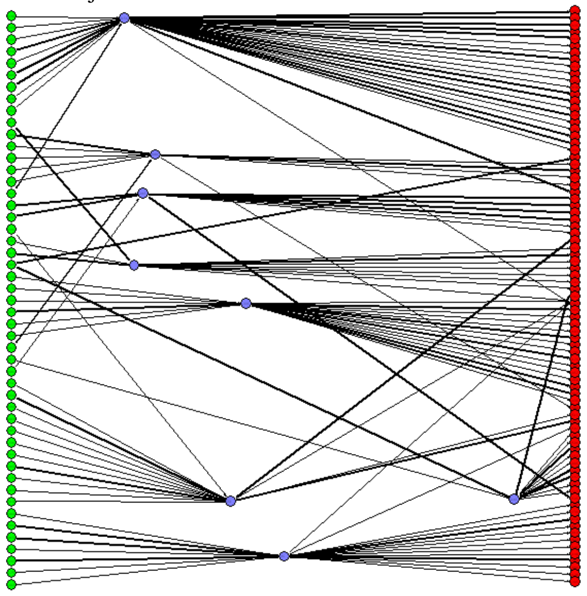


Fig.6. Graph showing the selected transactions by the query

This experiment shows how the analyst can apply his previous experience in searching money laundering transactions and the possibilities of visual exploration of cash flow chains. The program interface is in fact very simple, but just to define a SQL query may require some skills and experience in using this language. The solution to this question could be collaboration of the analyst with an SQL programmer who may predefine the specified queries or templates allowing the formulation of queries by QBE. However, the great advantage is the richness of language features that allow us to define any transaction filtering operation.

Experiment 2. Genetic optimization of graph of transactions

In this experiment, we are going to assess the quality of an evolutionary algorithm used to plot the graph [5]. The goal is to answer the question whether the parameters - the population size and number of mutations (N and M) - significantly affect the result obtained. In other words, we are interested whether with limited computing power it is more profitable to run a lot more shorter evolutions or fewer but longer ones.

Computing of the number of intersections in the graph has the complexity $O(m^2)$, where m is the number of edges in the graph [3], [13]. Thus, the computational complexity of the algorithm is $O(m^2 * N * M * C')$, where the parameter C' is dependent on the parameter C. From the definition of the parameter C, the estimation of C' is very difficult, since it is not known how many times one can carry out the innermost loop of the algorithm.

The algorithm has been tested on the previous database with 140 accounts and 306 transactions. In the experiment, the parameter $C = 100$, mutation probability of the source and destination nodes are 50%, and 90% of intermediate vertices. TABLE II shows the number of intersections obtained in each of the nine trials.

TABLE II. RESULTS OF GRAPH OPTIMIZATION

Number of individuals	*Number of mutation trials*	*Number of intersections*
1	1	277
	2	426
	3	258
2	1	229
	2	304
	3	373
3	1	506
	2	353
	3	334

The best result is the 229 intersections, and this is less than half the size of the worst. The standard deviation of the results is 87.0. The experimental data showed that the number of iterations (C) and population size (N) affected the value of the evaluation function. Fig. 7 shows the decreasing value of the objective function for individual evolution. After conducting many experiments we can conclude that it is better to run several times short mutations of many individuals rather than focus on a few individuals and increase the number of iterations.

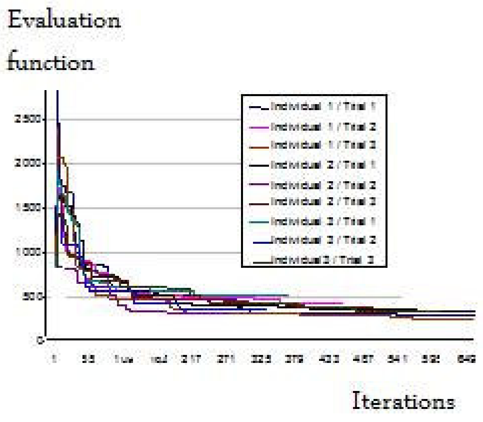


Fig.7. Curves of evaluation functions

IV. Conclusions

This study presents the visual exploration of cash flow chains, which has not previously been addressed in existing systems. The greatest difficulty in designing and implementing the system was to find a way to reduce the size of the transaction chains graph. This process has been divided into two stages. First, the heuristics have been applied to deploy accounts by type, then grouping accounts with the same neighbor in the graph, and placing at the bottom of the graph accounts that are related with only one other account. In order to reduce the visual complexity of the multiple edges, the edges were aggregated into a single edge (displayed in bold).

The graph visualization was optimized by an evolutionary algorithm whose task was to find an organization of vertices and edges with minimal number of intersections. Experiments have shown that the number of intersections in the graph has been reduced significantly, which improved its readability.

It is noteworthy that the system offers other interesting features such as logging user operations performed on the graph. The register of user actions can be used to work further on machine learning that would try to imitate the work of the analyst.

The proposed solution can be easily adapted into existing systems of transactions analysis and contribute to the development of new methods for exploration of complex data structures.

References

[1] Actimize, http://www.actimize.com, date of access: 2010-08-23.

[2] V. Bryant, "*Aspects of combinatorics. A Wide-ranging Introduction*", Cambridge Univ. Press, 1993.]

[3] T.H. Cormen, C.E. Leiserson, R. Rivest, C. Stein, "*Introduction to Algorithms*", MIT Press, 2009.

[4] D. S. Demetis, Artificial non-intelligence and anti-money lundering, London School of Economics, London, 2009.

[5] D. E. Goldberg, "*Genetic Algorithms in Serach, Optimization, and Machine Learning*", Addison-Wesley, 1989.

[6] E. Iwonin, City Handlowy, Implementation of AML directive into Polish legal system - City Handlowy experience, in Advanced Information Technologies for Management AITM 2009 Wrocław University of Economics, Research Papers no. 85, Wrocław 2009.

[7] J. Korczak, W. Marchelski, B. Oleszkiewicz, A New Technogical Approach to Money Laundering Discovery using Analytical SQL server, in Advanced Information Technologies for Management – AITM 2008, J. Korczak, H. Dudycz and M. Dyczkowski (eds.) Research Papers no 35, Wroclaw University of Economics, 2008, pp. 80-104.

[8] J. Korczak, B. Oleszkiewicz, Modelling of Data Warehouse Dimensions for AML Systems, in Advanced Information Technologies for Management – AITM 2009,.J. Korczak, H. Dudycz and M. Dyczkowski (eds.) Research Papers no 85, Wroclaw University of Economics, 2009, pp. 146-159.

[9] W. Łuszczyk, Wizualna eksploracja łańcuchów transakcji bankowych, Praca magisterska (MSc thesis), Uniwersytet Ekonomiczny, Wrocław, 2010.

[10] Norkom, http://www.norkom.com, date of access: 2010-08-23

[11] Ocean Technology,http://www.oceantechnology.com.vn /front/?page= solution&sid=27, date of access: 2010-08-23

[12] TETA S.A., http://www.teta.com.pl/71578.php, date of access: 2010-08-23.

[13] R.J. Wilson, „*Introduction to Graph Theory*", Addison-Wesley, 1996.