System Supporting Money Laundering Detection

Rafał Dreżewski[a], Jan Sepielak[a], Wojciech Filipkowski[b]

[a] *AGH University of Science and Technology, Department of Computer Science, Kraków, Poland*
[b] *University of Białystok, Faculty of Law, Poland*

Abstract

Criminal analysis is a complex process involving information gathered from different sources, mainly of quantitative character, such as billings or bank account transactions, but also of qualitative character such as eyewitnesses' testimonies. Due to the massive nature of this information, operational or investigation activities can be vastly improved when supported by dedicated techniques and tools. The system supporting the police analyst in the process of detecting money laundering is presented in this paper. The main parts of the system are data importer and analyzing algorithms, such as transaction mining algorithm and frequent pattern mining algorithms. The results obtained with the use of these algorithms can be visualized, so that they can be easily explored by the police analyst. The transactions found can be treated as suspected operations. The frequent patterns found are mainly used to identify the roles of suspected entities. For the transaction mining algorithm a performance study is also presented.

Keywords: Money laundering, money transfer analysis, clustering, frequent patterns, bank statement import

1. Introduction

Money laundering consists in granting legitimacy to operations on properties, which are criminal offenses. This is done by making a series of transactions that are carried out both through the legal or the official financial system, and illegal or unofficial circulation of property assets.

Email address: drezew@agh.edu.pl (Rafał Dreżewski)

The most important fact is that these practices try to simulate a legal, normal turnover. Transactions are usually hidden in the vast volume of official financial market. Moreover, in these practices many characters are involved. There can be from a few to several hundred individuals or entities engaged in criminal economic activities. Each of them has usually more than one bank account. These facts make it difficult to analyze such activities in a traditional way, using simple tools like spreadsheets.

Money Laundering Detection System (MLDS) was designed and implemented to get over this issue. The main purpose of MLDS is to interactively or automatically analyze financial flows (money transfers within the bank system) in order to find money laundering operations, which are hidden in thousands of legal operations. MLDS is a module of the CAST/LINK system developed at the AGH University of Science and Technology in cooperation with Polish State Police to support the work of the police analyst.

The remainder of the paper is organized as follows. In section 2, we analyze the process of money laundering. In section 3, we present related works on money laundering detection. In section 4, we present the implemented system. We discuss our experimental results in Section 5 and conclude with final remarks in section 6.

2. Money Laundering Phenomena

Financial crimes are types of economic crimes which involve using instruments and institutions of the financial market for obtaining financial gains at the expense of other market players. Such activities are illegal because they constitute a violation of one of the foundations of a free-market economy, namely the confidence of market players that each of them shall observe the applicable rules, whether written or not[1]. Moreover, criminals often conduct transactions of little economic sense, since maximizing profits is not their priority. This violates the free-market rules by introducing an element of unpredictability into the market and consequently increases the investment risk.

Money laundering is an economic crime. It is strongly related to operations of organized criminal groups. It must be emphasized that it is difficult

[1]Compare (Górniok, 2000, p. 13), (Grabarczyk, 2002, p. 35ff.) and (Skorupka, 2007, p. 16)

to present one universally binding, or at least commonly accepted, definition of this phenomenon (Prengel, 2003, p. 95ff.) (see also Article 6 of the *United Nations Convention against Transnational Organized Crime* available in the *United Nations Convention Against Transnational Organized Crime and the Protocols Thereto*, 2004). For the purpose of this paper, we will define money laundering as the conduct of a number of activities that are mainly aimed at giving the appearance of legality to income originating from illegal acts. Criminals make efforts to introduce their illegally gained money into legal financial transactions. One could make a proposition that money laundering makes it possible to transfer financial assets from the so-called gray market and the criminal underground to the legal, official economy (van Duyne, 2003, p. 3ff.). This way, criminals can freely use such assets without the need to worry that the tax office will question their legal origins and that the law enforcement or administration of justice agencies will seize and forfeit them. There is also a secondary purpose of money laundering. It is to conceal the identity of persons who coordinate activities from agencies of law enforcement and administration of justice in order to escape prosecution.

It should be emphasized that money laundering is not a uniform phenomenon; on the contrary, it has multiple forms and variations. They depend on economic, political, geographic, and social factors. What can be considered to be a common characteristic of all the methods used by criminals is the presence of numerous transactions among numerous entities—natural or legal persons and their bank accounts—in a fairly short amount of time. Criminals strive to hide their actions in a huge number of cash and non-cash transactions performed daily on the financial market. One of the basic methods is to mix legal income with illegal income. The purpose is to make it difficult to separate the two types of income earned by one or many entities cooperating with criminal groups.

The literature mentions many other methods of money laundering. Their typologies are elaborated on the basis of information drawn from criminal cases, analysis conducted by financial intelligence units (Wójcik, 2007, p. 249ff.), and analysis of the risk that specific instruments or institutions may be used for money laundering[2].

In light of the above, law enforcement and subsequently administration of justice agencies must answer the following questions:

[2]Compare (Filipkowski, 2004, p. 61ff.) and (Wójcik, 2007, p. 241ff.)

- How can a huge number of records of transactions constituting evidence in a criminal case be analyzed?
- How can one pick those related to money laundering?

Both of these questions pertain to finding the accounting trace that will make it possible to determine the route followed by the financial assets from their illegal origins, through a network of entities and bank accounts, all the way to their present location. This determination is immensely important for successful operational work and criminal cases. It is one of the conditions for effectively prosecuting perpetrators. The literature uses the terms "criminal analysis", "criminal intelligence analysis" and, more specifically, "financial flow analysis" or "transaction pattern analysis" to describe this process (Michna, 2004, p. 71), (Paynich and Hill, 2010, p. 9).

There is no doubt that new technologies are more and more extensively used by agencies of law enforcement and the administration of justice, for example compare (Tomaszewski, 2008, p. 155). In this context, one may wonder how important criminal intelligence analysis is in operational work. Its basic advantage is that it makes it possible to arrange pieces of data collected in a case and to show relations between them. At the same time, it shows directions for additional activities at the operational work stage or investigation, such as obtaining another bank account record or extending investigation to another person.

To an extent, criminal intelligence analysis serves a similar purpose in criminal cases. Technicalities do not change. However, there are doubts as to whether financial analysis can be used as stand-alone evidence or only as information about evidence concerning, for example, bank statements (as recent research in this field indicates (Chlebowicz and Filipkowski, 2011, p. 145ff)). The word "criminal", as opposed to "criminalistic" or "forensic" is commonly used in reference to the analysis. Thus, can it be described as an expert opinion or forensic analysis? At this time, Poland has neither relevant laws, nor a uniform doctrine, nor a uniform practice on presenting results of analysis as evidence (Wójcik, 2004, p. 374ff.), (Chlebowicz and Filipkowski, 2011, pp. 163–165). To an extent, this is a *terra incognita*.

It is only natural to look for new solutions and to use methods based on artificial intelligence and agent systems to increase the speed and efficiency of analysis like that. It should be emphasized that calculations performed by such systems go beyond those using a simple spreadsheet. New methods of numerical data analysis have been tried and tested by the scientific

community. Second, it cannot be denied that analyzing large amounts of data and searching for transactions that should be of interest to agencies of law enforcement and the administration of justice, frequently goes beyond perception and analytic capabilities of a single person performing work manually, on paper. Such an analysis would very likely include human errors. Consequently, its value could be questioned in a criminal proceeding. This is why there have been efforts to employ computers and appropriate software to replace people in performing this task. Since such systems have not been in use before, many controversies and issues are to be solved by lawyers. Nevertheless, only recently there have been efforts in Poland aimed at testing their suitability for the use in criminal cases or, more broadly, in the activities of law enforcement and the administration of justice agencies.

In view of the above, one should answer the question of whether criminal intelligence analysis should be considered as the so-called scientific research evidence in criminal cases. This pertains especially to the analysis performed using advanced scientific methods.

According to Tadeusz Tomaszewski it should meet the following criteria (Tomaszewski, 2008, p. 163):

1. Verification of correctness of the adopted scientific principles and credibility of research methods used by the analyst.
2. Verification of correctness of research techniques—types of measures and materials used.
3. Verification of the analyst's qualifications and their adequacy for the research conducted.

As we can see, only the first item pertains to theoretical, general questions. The other two pertain to the evaluation of evidence in a specific case. The first item requires evaluating the integrity and reliability of the method used with regard to repeatability of the results obtained in identical conditions. The second item requires verifying the correctness of the very research method. Additionally, the results obtained should be reproduced by independent experts in the future using the same evidence. The third item focuses on the accuracy of measurements and calculations, which can be verified using statistical tests aimed at determining the percentage of erroneous indications. It can also be achieved by ensuring that accepted professional standards and techniques are applied. Researchers as well as practitioners, who work in this field, must refine and elaborate procedures and methods of

criminal intelligence analysis so that the effect of their work can be used as stand-alone evidence in criminal cases.

It should be pointed out that there are differences in legal and forensic science approach to the criminal intelligence analysis and the use of its results. Practice shows its widespread use by law enforcement, since it brings enormous benefits to the detection of crime, particularly in financial investigations. Legal problems should not hinder researchers in the constant improvement of available tools.

3. Related Work and Contribution

There is a wide variety of commercial anti-money laundering (AML) software. However, its implementation details are a well-guarded secret. Nevertheless, there are many publications proposing how implementation can be done.

In (Zhang et al., 2003) a new methodology for Link Discovery based on Correlation Analysis to address money laundering detection problem was proposed. The authors developed a Hierarchical Composition Based correlation analysis along timeline to generate the Money Laundering Crimes (MLC) group model. The time axis of timelines was "discretized" into time instances. Each node in the timelines recorded part of the financial transaction history. They projected all transactions to the timeline axis by accumulating transaction frequency to form a histogram. Clustering of transactional activities based on histogram segmentation was carried out. To detect suspicious patterns, local and global correlation analyses were performed.

In (Kingdon, 2004) support vector machine (SVM) was applied. The approach identified unusual behavior and detected potential money-laundering situations. A matrix with massive dimensionality was created to deal with heterogeneous data. Due to the size of matrix, the system had performance issue.

In (Jun, 2006) a system based on outlier detection among financial transaction was developed. A fast distance estimation was introduced to optimize the computation of deviation from the tested data point to the reference dataset. In (Zhu, 2006a), (Zhu, 2006b) outliers were sought among transactions from customers in the same industry. Accounts from the obtained transactions were suspicious.

In (Gao et al., 2006) intelligent agent technology was applied. Agents were distributed in financial organizations or departments involved in AML. They

communicated with each other through the Internet. The system consisted of five kinds of agents: user agent, data collecting agents, monitoring agents, behavior diagnostic agent and reporting agent. The agents performed analysis, monitoring and diagnosed suspicious transactions. When a suspicious transaction was encountered then a potential alert was sent to appropriate personnel. Multi-agent neural network, text mining, genetic algorithms, velocity analysis and case-based reasoning were applied in (Qifeng et al., 2007).

In (Wang and Yang, 2007) a method how to evaluate the potential money laundering risk of each customer was proposed. To establish whether the risk was low, medium or high a decision tree was built with some rules proposed by the authors.

A radial basis function (RBF) neural network was applied in (Lv et al., 2008). APC-III clustering algorithm was used for determining parameters of radial basis function in a hidden layer, and recursive least square (RLS) algorithm was leveraged to update weights of connections between hidden layer and output layer. The RBF neural network was trained to determine whether transaction data was illegal or not.

Sequence matching based algorithm to identify suspicious transactions was proposed in (Liu et al., 2008). The authors used classical Euclidean similarity distance to define similarity between different sequences. A threshold-based method was applied to classify normal and suspicious sequences.

In (Le-Khac et al., 2009a) two parameters of the model were introduced: proportion between the redemption value and the subscription value conditional on time (Δ_1), and proportion between a specific redemption value and the total value of the investors' shares conditional on time (Δ_2). For each value of Δ_1 and Δ_2 K-mean classification technique was applied. A back-propagation based neural network was used to determine suspicious degree of transactions. The results were stored in the knowledge-base. In (Le-Khac et al., 2009b) some improvements were added. Transactions were still clustered by using the center-based clustering algorithm. Transactions were divided into two groups: individual and corporate. A novel heuristic approach was used to find initial centers of clusters. The modification allowed to improve performance.

Clustering system with outlier detection was also proposed in (Gao, 2009). In (Luell, 2010) in a given graph, connected sub-graphs were identified in order to perform clustering. To find out whether the grouped sub-graphs are suspicious, pattern matching algorithms were proposed. The approach allowed to detect *transaction chains* and *smurfing* activities.

In (Raza and Haider, 2011) an approach called Suspicious Activity Reporting using Dynamic Bayesian Network (SARDBN) was presented. It employed a combination of clustering and dynamic Bayesian network (DBN) to identify anomalies in sequence of transactions. SARDBN identifies abnormalities in sequence of transactions. To detect anomalies Anomaly Index using Rank and Entropy (AIRE) was developed. The computed index was compared with a pre-defined threshold to mark the transaction as normal or suspicious.

Varieties of investigations have been done on information extraction from textual files. Hammer et al. (1997) proposed a fully manual system to process textual documents. The user had to code wrapper (a procedure which translated content) for each document type, the system interpreted the wrapper and executed provided instructions. There were many learn-based approaches proposed in order to extract information (Chawathe et al., 1994), (Grumbach and Mecca, 1999), (Garofalakis et al., 2000), (Chang and Lui, 2001). Many authors elaborated systems which tried to induct wrapper to extract required information (Kushmerick et al., 1997), (Hsu and Dung, 1998), (Muslea et al., 1999), (Kuhlins and Tredwell, 2002). In (Black et al., 2004), (Spiliopoulou et al., 2004) and (Wong, 2009) ontology learning systems were used to identify entities in documents. Nekvasil (2007) described an approach in which wrapper was induced from ontologies.

In the approach presented in this paper a rule-based system was implemented to import bank statements because the domain is closed (bank statements). Moreover, such a method is easier to interpret and develop. The applied approach is similar to the system described in (Hammer et al., 1997). Both allow to construct trees of instructions to perform. However, the approach proposed in this paper is more general. Output values can be described with regular expressions. Instructions to carry out are presented explicitly in a tree. The user does not have to code the template (which is a counterpart of the wrapper), but he/she can configure it in rich UI controls by using a mouse.

The main contributions of this work are:

1. Application of frequent pattern mining algorithms to detect money laundering, when the following methods were used: *distributive boxes* and *collective boxes*.
2. Proposal of detecting suspicious clusters by recognizing laundering methods and roles of the offender.

3. Proposal of bank statement importer, which is not bound to any file format.

4. Money Laundering Detection System

Money Laundering Detection System (MLDS) is part (module) of the system supporting work of the police analyst (CAST/LINK), which is being developed at the AGH University of Science and Technology in cooperation with the Polish State Police.

The main purpose of the system presented in this paper (MLDS) is to automatically analyze financial flows in order to detect money laundering processes. The system carries out the analysis by:

- Clustering—clusters are sets consisting of money transfers which fulfill specified criteria such as: flow condition, gathering amounts of money to a single account, minimum set size. Cluster elements can be treated as suspected operations utilized in money laundering.
- Mining for frequent sets and sequences in clusters that are found—it is helpful to identify potential money laundering and the roles of entities involved in this activity.
- Data visualization—obtained clusters and frequent patterns can be visualized in schema and timeline diagrams. It allows the police analyst to review and interpret retrieved results. Visualization facilitates the user to explore provided data in order to find suspected money flows and indirect relationship between suspected offenders.

4.1. Architecture of the System

Figure 1 shows diagram of the system architecture. There are the following modules in the system:

- *Import module*—reads of money transfer data from bank statements into the system. On the output it provides bank statements data in the internal system format.
- *Clustering module*—analyzes imported data using clustering algorithm. On the output it provides clusters which are input of frequent pattern mining algorithms.

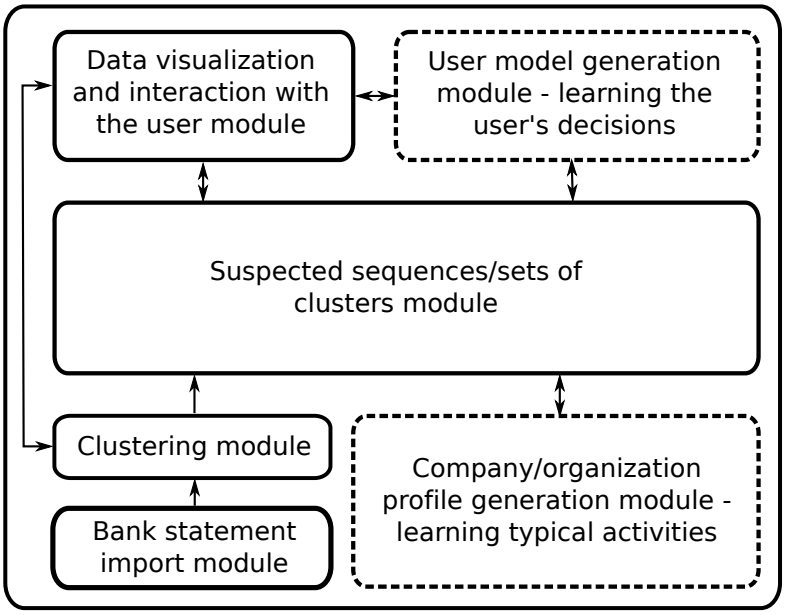

Figure 1: Architecture of the system used in money flow analysis

- *Suspected sequences/sets of clusters module*—analyzes the clusters with the use of frequent pattern mining algorithms. On the output it provides frequent patterns that are found (sets or sequences).

- *Data visualization and interaction with the user module*—visualizes the results on dedicated schema and timeline diagrams.

- *User model generation module*—learns from decisions made by the user interacting with the system. This module deals with the user interacting with the system (police analyst) and learns its decisions during the analysis of suspected sequences of transfers. Then the constructed user profile may be used by the system to additionally filter suspected sequences on the basis of the previous user's decisions. On the output it provides a model with the user's decisions.

- *Company/organization profile generation module*—automatic generation of profiles of suspected companies/organizations typical activities, including titles of money transfers, time/date of making money transfers, amount of money being usually transferred, etc. This module automatically learns, based on bank statements, typical financial operations

of the suspected company/organization. On the basis of such data, the profile of the company may be created and used during further filtering of suspected sequences of money transfers. On the output it provides a profile of typical activities of suspected companies/organizations.

The two last modules which are marked in Figure 1 with the dotted line, have not been implemented yet, but they are to be implemented in the future. These modules are to support the automation of suspected financial operations detection, which may provide valuable input for the human decision maker to draw his attention to selected transactions, or even generate money laundering warnings when needed.

Figure 2 shows data flow between implemented algorithms. At the beginning the system analyzes data coming from the importer of bank statements. The first stage of the analysis relies upon clustering. The obtained clusters are passed on to the frequent pattern mining algorithms. The retrieved results are a subject for visualization in schema and timeline diagrams. Visualized data is presented to the police analyst.

4.2. Supplying data to analyzing modules

Input data is provided in bank statement files. Files are electronic and textual. Different banks provide files in different formats. In order to import data from these files, the importer of bank statements was implemented. The importer does not perform OCR analysis. When it is necessary to import hard copied statements, then external tools have to be used to provide textual and electronic file to the module.

There is a wide variety of accounting software, both commercial and free, which import textual files with financial data. However, they accept only predetermined file types like OFX, QFX, QIF, IFX, XBRL, OFC, CVS, etc. The implemented system is not bound to any file format. The user constructs templates which allow to import files of formats unknown at the implementation stage.

The import process is divided into three stages:

- Header analysis which provides the following data: name and address of the bank account holder for whom a bank statement was created, currency of the account, and the account number.

- Money transfer separation which separates the bank statement into independent fragments. This step is preparatory for the next stage.

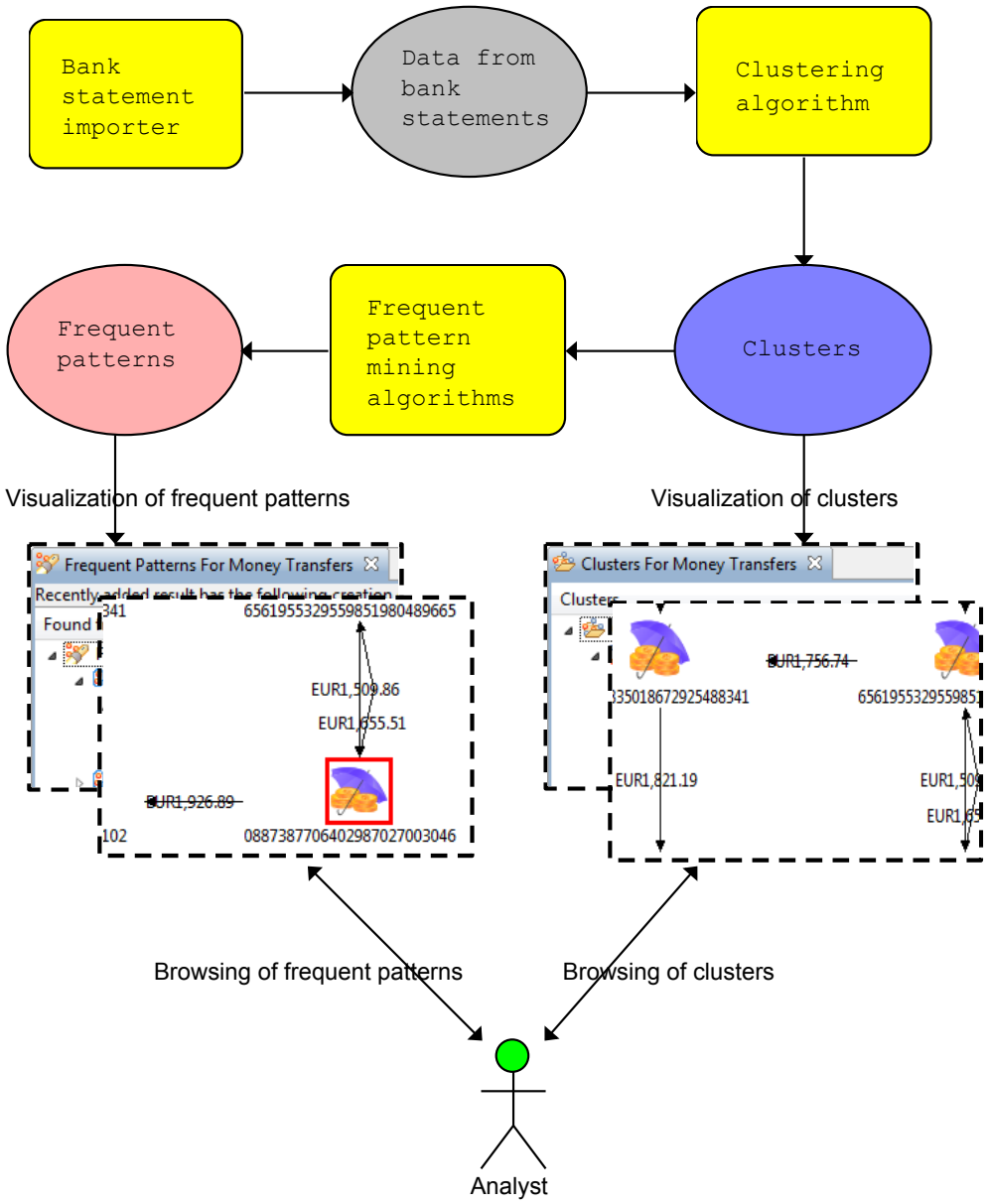

Figure 2: Schema of data flow in the system

- Money transfer data extraction which allows to obtain data related to money transfers such as operation ID, operation name, title, balance, date, account number, name and address of the holder, transfer direc-

tion (incoming, outgoing).

The importer work is based on templates. Templates specify what activity should be carried out to parse an input file. Parser actions are basic elements of templates. They determine what the importer should carry out with the input file. Parser actions work on data in the form of lines or columns. Using parser actions trees are built. Each template consists of three trees. They are: (1) tree used to parse the header of bank statement, (2) tree used to separate money transfers, (3) tree used to parse separated money transfers.

The parser actions can be divided, with regard to getting and returning data, into four categories:

- actions getting lines and returning lines;
- actions getting lines and returning columns;
- actions getting columns and returning columns;
- actions getting columns and returning lines.

What is more, some parser actions provide entries wrapping recognized portions of text. The entries are the most important outcome of the import process because they represent result data.

There were 13 parser's actions implemented in the system:

- *Column Filter*—gets columns and returns columns. On each column the following operations are performed. Column lines are selected using line indicators (line indicators supply line index using regular expressions or integer number entered by the user.) Groups from the specified regular expressions are used to remove parts of the obtained lines. All the specified regular expressions are applied to each line. An output of one expression is an input of the next one.
- *Column Line Selector*—gets columns and returns columns. The action iterates over columns and selects lines from columns. To determine which lines have to be selected, line indicators are used. To preserve the same line count in each column only, indexed line indicators are allowed.
- *Column Selector*—gets columns and returns lines. Represents an action which selects a single column from input columns. The column is selected using a specified index.

- *Column Separator*—gets lines and returns columns. Represents an action which allows to separate columns. It requires that input boundaries of columns which are to be used indicate where each column starts and where it ends.

- *Delimited Values*—gets lines and returns lines. Represents an action which extracts entries from a sequence of delimited strings. The entry type is recognized using a string which precedes the entry value. When the entry value is recognized, then a filter can be applied to it to remove some undesired characters.

- *Line Filter*—gets lines and returns lines. It selects lines from an input list of lines using line indicators. It uses groups from specified regular expressions to remove parts from the obtained lines. All the specified regular expressions are applied to each line. An output of one expression is an input of the next one.

- *Line Removal*—gets lines and returns lines. It represents a parser action which removes required lines from an input list of lines. There are two ways to specify which lines should be removed: (1) using a line indicator—allows to remove only a single line, (2) using a regular expression—removes a given number of lines.

- *Line Selector*—gets lines and returns lines. Selects lines from input lines. Line indicators are used to determine which lines should be selected. The lines outside of the indicated region are removed. Start and end indicated lines are left.

- *Multiple Line Matcher*—gets lines and returns lines. The action works on a range of lines indicated by line indicators. Entries supplied by this action come from objects describing the line content. If there is more than one such object, then the action tries to infer on which lines each object should work.

- *Region Removal*—gets lines and returns lines. The action removes regions consisting of lines from the input lines. The regions to remove are indicated using the line indicators. The number of regions to be removed can also be specified.

- *Simple Column Separator*—gets lines and returns lines. Allows to divide lines into columns. The columns must be separated by using whitespace characters. To separate columns only one parameter is used—the minimum count of characters separating the columns. In relation to the *Column Separator* action, this action is much simpler in the configuration, but provides narrower possibilities and is less flexible.

- *Single Line Matcher*—gets lines and returns lines. When a line indicator is specified, then the action works on the indicated line. In another case, the action iterates over the lines and when entries can be retrieved from a given line, then the iteration stops. To obtain the entries, a regular expression is used. Each group from such an expression can be treated as an entry value.

- *Statement Slicer*—gets lines and returns lines. The action allows to separate regions representing money transfers, in input lines. An acceptance condition for the regions can be specified. If it is not fulfilled, then the entire region is thrown away. To find the regions, two kinds of arrays with regular expressions are used: the first to find the beginning of the region, and the second to find the ending of the region. The expressions in the arrays are connected with the logical disjunction.

Parser actions simplify the system architecture and its extensibility. The former is correct due to the fact that the system consists of simple and isolated actions. The latter appears because only some simple parser actions should be implemented to extend importer functionality, leaving other actions untouched.

Regular expressions are a very important element in the implementation of parser actions. Thanks to the implemented GUI, the user does not need to enter manually regular expressions. Created GUI allows to select required elements in lists and the system will build regular expression automatically.

An implemented preview is an important element of GUI. It allows to display data which comes from:

- execution of a single parser action,

- execution of a single path from a tree with parser actions,

- execution of entire templates.

Preview shows the following data:

- lines of columns returned by actions,
- separated lines representing money transfers,
- recognized bank statement elements (e.g., account numbers, amounts, dates, etc.).

Preview indicates which fragments of the file were recognized by parser actions. It is also helpful during the construction of the action tree, because it allows to find out which data is got and returned by parser actions. Hence it is an auxiliary tool in the bug tracking.

4.3. Implemented algorithms

The system contains the following most important algorithms used for analyzing money flows: the clustering algorithm and frequent pattern mining algorithms.

4.3.1. Clustering

The clustering algorithm builds cyclic, directed and connected graphs in which nodes represent money transfers. An edge is created between two nodes when the target account from the transfer of the first node is the same as the source account from the transfer of the second node. For instance, let a cluster contain n_1 nodes representing money transfers with target accounts a_1 and n_2 nodes representing money transfers with source accounts a_2. When a new money transfer is to be added with the source account a_1 and the target account a_2, then the total number of edges to create equals $n_1 + n_2$.

The amount from money flows is between the minimum and/or maximum values. These values are specified by the user. Prior to clustering, amounts from the money transfers are unified to a single currency, because money flows can be made in different currencies. Exchange rates used in this step are daily and market. They are downloaded automatically from a given URL address.

There also exists the possibility that clusters may begin only from money flows which cause "zeroing" of the source account. The account is treated as empty only when balance is not greater than zero. When relatively inactive balance is left on the account, then the account is not treated as empty.

The constructed clusters must fit in time window whose size is entered by the user.

When a constructed graph is to be accepted three conditions are taken into account: flow condition, gathering amounts of money into a single account, minimum cluster size. Flow condition allows to cut off graph nodes which do not fulfill the condition. It considers commission received by entities involved in money laundering. Gathering amounts of money into a single account allows to detect *gatekeepers*, i.e. final, target accounts.

The flow condition algorithm is elaborated by the authors. In a nutshell, it leaves money transfers in the graph, when such a possibility exists. The algorithm analyzes trees into which graph with money transfers in nodes was split. On the input, it receives a tree root (node without offspring). In the first step, offspring of the root are analyzed. When the amount from the root is greater than the sum of amounts from children, then children are removed and the algorithm is stopped. In other cases, for each child, the minimum amount is computed, which will assure that the entire amount from the root will spread. For each child the algorithm is called recursively. The child becomes a root, but the amount of this root is not the amount from the money flow of the child, but the minimum computed earlier.

The clustering algorithm gets on input parameters allowing to find money transfers, which can be treated as suspected operations utilized in money laundering.

The next step of the analysis—frequent pattern mining—can be used to establish methods applied by offenders. It allows to determine with a higher probability that clusters that are found contain money transfers utilized in the illicit activity.

4.3.2. Frequent pattern mining

To mine frequent sets in the presented system, three algorithms can be used: *FP-Growth*, *FPClose* and *FPMax*.

Algorithm *FP-Growth* (Han et al., 2000) is used to find all possible frequent itemsets. It uses FP-tree which stores database in compressed form, and thereby memory usage is reduced. What is more, it diminishes the number of scans of database in the search process.

FP-tree is a rooted and acyclic graph with labeled nodes. The root has the *null* label, whereas other nodes represent one-element frequent itemsets. Labels in nodes, except the root, contain one-element itemset and a number of clusters which support the itemset. When the FP-tree is explored all frequent itemsets are found. The exploration process is based on observation, that for each one-element frequent itemset α all frequent supersets of α are

represented in FP-tree by paths which contain node(s) α.

FP-Growth neither generates nor tests candidate itemsets. It utilizes *divide-and-conquer* methodology dividing tasks into smaller ones.

Algorithm *FPClose* (Grahne and Zhu, 2003) is based on *FP-Growth* algorithm and allows to find closed frequent sets (frequent pattern t is closed if none of t's proper super-pattern has the same support as t has).

When *FPClose* finds a new itemset, the following operations should be performed in order to check whether it is closed (Wang et al., 2003):

- superset-checking—checks if this new frequent itemset is a superset of some closed itemset candidate with the same support that has already been found,
- subset-checking—checks if the newly found itemset is a subset of a closed itemset candidate with the same support that has already been found.

Because *FPClose* works under divide-and-conquer and depth-first-search framework, to determine whether a frequent itemset is closed, it checks if the newly found itemset is a subset of a closed itemset with the same support that has already been found. If such a superset does not exist then the itemset found is closed (Wang et al., 2003).

All the frequent closed itemsets are stored in FP-tree in order to reduce memory usage and to speed up subset-checking. If the current itemset S_c can be subsumed by another closed itemset S_a that has already been found, they must have the following relationships: (1) S_c and S_a have the same support; (2) The length of S_c is less than that of S_a; and (3) all the items in S_c should be contained in S_a (Wang et al., 2003).

To speed up subset-checking, a two-level hash table is used. One level uses ID of the last item in S_c as a hash key, another uses support of S_c as a hash key, and the result tree nodes falling into the same bucket are linked together. Each tree node in FP-tree has the length of the path from this node to the root node (Wang et al., 2003).

Algorithm *FPMax* (Grahne and Zhu, 2005) is also based on *FP-Growth* algorithm and is used to mine maximal frequent sets (frequent pattern t is maximal if none of the t's proper super-pattern is frequent).

The algorithm is a modification of *FPClose* algorithm. FP-tree is also used to store intermediate results. In order to determine whether a frequent itemset is maximal, it checks if the newly found itemset is a subset of a

maximal itemset that has already been found. In order to determine whether itemset S_c is a subset of itemset S_a two conditions have to be met: (1) the length of S_c must be less than that of S_a; (2) all S_c elements must be contained in S_a. If S_c has no superset, it is maximal.

To mine frequent sequences, three algorithms were implemented: *Sequence Miner*, *BIDE* i *BIDEMax*.

Algorithm *Sequence Miner* is similar to *PrefixSpan* (Pei et al., 2001) algorithm and allows to mine all of the frequent sequences. The algorithm was elaborated by J. Wang and J. Han (Wang and Han, 2004). It does not contain any search space pruning method. To speed up support computing for the sequences that are found, pseudo projection is applied. Unlike physical projection, a set of pointers is recorded, one for each projected sequence, pointing at the starting position in the corresponding projected sequence (Wang and Han, 2004).

The general idea of the algorithm relies on examining only prefixes of sequences. Prefixes grow using sets of its locally frequent items.

Algorithm 1 shows *Sequence Miner* algorithm. Procedure *FrequentSequences* is called recursively. Non-empty prefix is added to the result set (line 7). The projected database S_p_SDB is scanned once to find the locally frequent items (line 9). Locally frequent item i is chosen to grow S_p to create a new prefix S_p^i (line 14). S_p_SDB is scanned once again in order to build a new pseudo-projected database for each new prefix S_p^i (line 15). As it can be seen, frequent sequences are found according to the depth-first paradigm.

List of abbreviations for Algorithm 1:

- SDB—an input sequence database,
- min_sup—a minimum support threshold,
- FS—the complete set of frequent sequences,
- S_p_SDB—a projected sequence database,
- S_p—a prefix sequence,
- LF_S_p—locally frequent items (items frequent in the current recursive step),
- S_p^i—a prefix sequence with item of index i,

Algorithm 1: Schema of *SequenceMiner* algorithm (Wang and Han, 2004)

```
1  SequenceMiner(SDB, min_sup, FS)
   Data: SDB — an input sequence database, min_sup — a minimum support
         threshold
   Result: the complete set of frequent sequences, FS
2  FS = φ;
3  FrequentSequences(SDB, φ, min_sup, FS);
4  return FS;
5  FrequentSequences(S_p_SDB, S_p, min_sup, FS)
   Data: S_p_SDB — a projected sequence database, S_p — a prefix sequence,
         min_sup — a minimum support
   Result: the current set of frequent sequences, FS
6  if S_p is non-empty then
7  |  FS = FS ∪ S_p;
8  end
9  LF_S_p = locally frequent items(S_p_SDB, S_p, min_sup);
10 if LF_S_p is empty then
11 |  return;
12 end
13 foreach locally frequent item i do
14 |  S_p^i = S_p ∪ i;
15 |  S_p^i_SDB = pseudo projected database(S_p^i, S_p_SDB);
16 |  FrequentSequences(S_p^i_SDB, S_p^i, min_sup, FS);
17 end
```

- S_p^i_SDB—a projected sequence database for a prefix sequence with item of index i.

Algorithm *BIDE* (Wang and Han, 2004) (elaborated by J. Wang and J. Han) can be used to mine closed frequent sequences. It can be created from *Sequence Miner* by adding, among others, checking whether a sequence has forward and backward extension.

This checking schema is called *BI-Directional Extension checking*. According to the definition of a frequent closed sequence, if a n-sequence, $S = e_1 e_2 \dots e_n$, is non-closed, there must exist at least one event, e', which can be used to extend sequence S to get a new sequence, S', which has the same support. Sequence S can be extended in three ways:

1. $S' = e_1 e_2 \dots e_n e'$ and support of S' is equal to support of S;
2. $\exists i \ (1 \leq i \leq n)$, $S' = e_1 e_2 \dots e_i e' e_{i+1} \dots e_n$ and support of S' is equal to support of S and
3. $S' = e' e_1 e_2 \dots e_n$ and support of S' is equal to support of S.

In the first case, event e' occurs after event e_n, e' was called a *forward-extension event* and S' a *forward sequence* with respect to S. While in the second and third cases, event e' occurs before event e_n, e' was called a *backward-extension event* and S' a *backward-extension sequence* with respect to S (Wang and Han, 2004).

BIDE checks whether a sequence has a forward and a backward extension. If it has no such extensions then it is closed.

This algorithm avoids the curse of the *candidate maintenance-and-test* paradigm, prunes the search space more deeply and checks the pattern closure in a more efficient way, while consuming much less memory in contrast to the previously developed closed pattern mining algorithms. It does not need to maintain a set of historic closed patterns, therefore it scales very well with the number of frequent closed patterns.

BIDE adopts a strict depth-first search order and can output the frequent closed patterns in an on-line fashion (Wang and Han, 2004).

Algorithm *BIDEMax* was elaborated by the authors. It was created by modifying *BIDE* algorithm, and allows to mine maximal frequent sequences. In contrast to *BIDE*, the most important modification is that support is not taken into account when extensions are tested.

5. Results

Using a described version of the system, a series of experiments were carried out. The aim of these experiments was to test the elaborated and implemented clustering algorithm and to check the correctness of the created visualization method of the analysis result. The calculations were performed using money transfers provided by the implemented test data generator.

The implemented generator creates 100 random accounts and then connects them creating money transfers. The accounts for money transfers are selected at random. As the result, multiple directed, cyclic and connected graphs are created, which contain bank accounts in nodes. The edges represent money transfers.

All experiments were performed with the use of the following values of parameters:

- The number of money transfers—2000. This value guarantees that measured results do not vary and computational time is not too long.
- The number of different accounts used to create money transfers—100. When the value is too low then no clusters are found. When the value is too high then computations are slow. 100 accounts ensure that a reasonable number of clusters is mined.
- The number of money transfers made in a single day—50. This parameter is related to the size of the time window. In order to provide the same number of clusters when this parameter is smaller, the time window has to be larger. And when the number of money transfers per day is greater then the time window should be smaller. The used parameter values allow to provide stable results in proper time.
- The size of time window—1 day. It is related to the above parameter.
- The minimum number of money transfers which should gather into a single account so that a cluster can be accepted—1. This value causes that the gathering of amount is not checked. When this parameter is greater than 1 then experiments show that memory usage almost does not change, and the higher value of this parameter the longer work time. It is not really surprising, so, for simplicity reasons, when other parameters are examined then this parameter has value 1.

All experiments were carried out on the machine with one Intel Pentium M 740 processor and 2 GB of RAM.

The algorithm was run 10 times for each parameter value coming from the established range, and average results were computed.

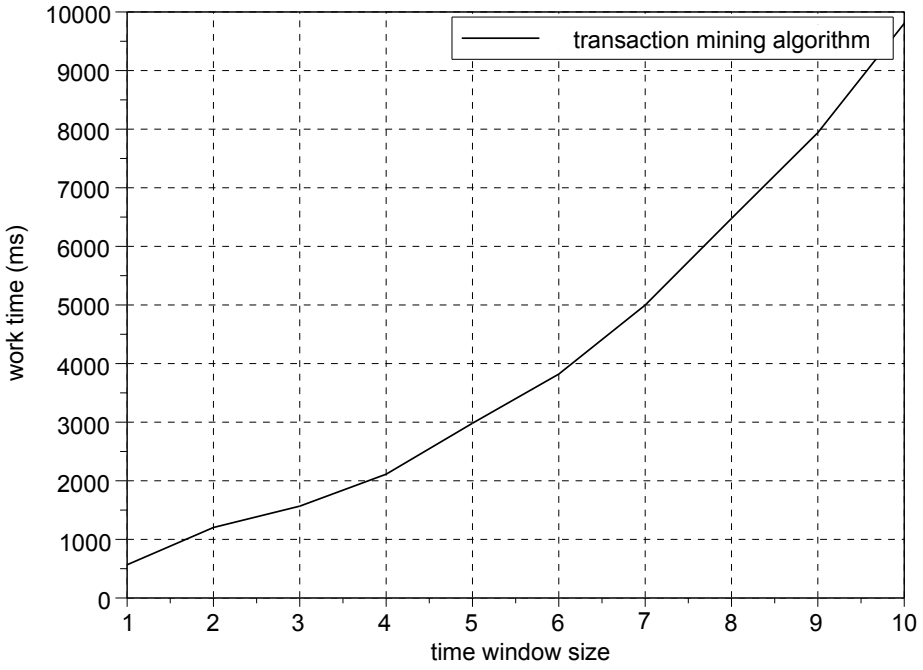

Figure 3: The work time of the clustering algorithm depending on the size of time window (average results from 10 runs of the algorithm)

Figure 3 shows work time of the clustering algorithm when the size of time window was changed. It shows that the greater size of time window the longer work time. The presented plot is similar to a power function, because work time grows faster and faster.

Figure 4 shows work time of the clustering algorithm when the number of input money flows was changed. In this case the number of money transfers per day was constant, because the period from which flows came is proportionally wider. The figure shows that when the number of money transfers grew then work time was also longer.

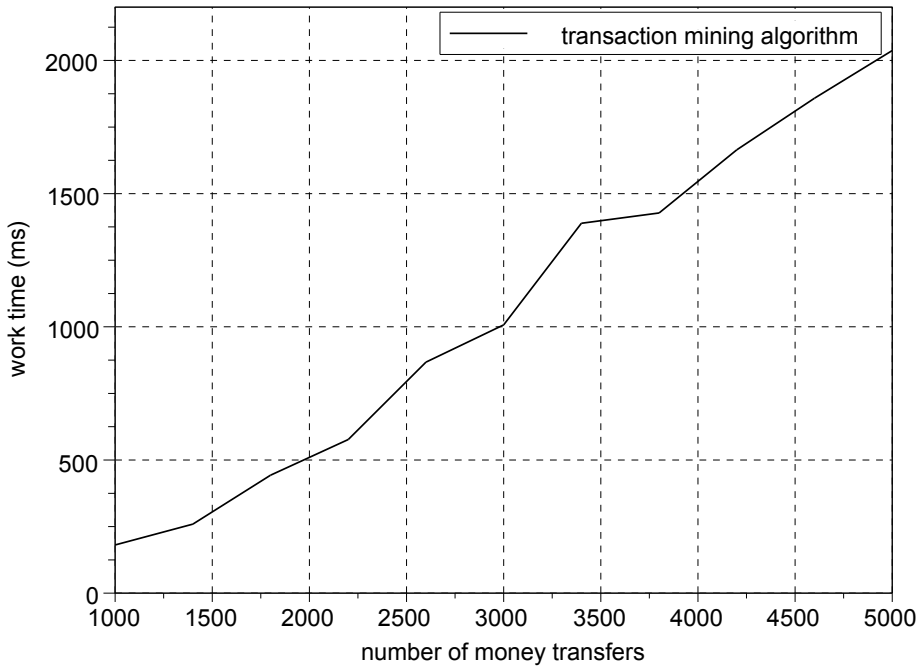

Figure 4: The work time of the clustering algorithm depending on the number of input money transfers (average results from 10 runs of the algorithm)

The influence on work time of two other parameters was also measured: the minimum number of money transfers which should gather into a single account, and the number of different accounts used to create money transfers. However, unambiguous influence was not observed.

To visualize the result data, two views are used: a view with clusters and a view with frequent patterns. Moreover, the clusters and frequent patterns that are found can be presented in the schema and timeline diagrams.

Figure 5 shows a view with clusters, where each cluster is a graph. The cluster is presented as a list of graph nodes. Each tree node may have children. The tree with nodes illustrates between which accounts money was transferred. In addition, columns, which are displayed on the right side of the tree, allow the analyst to quickly access the detailed data about each transfer.

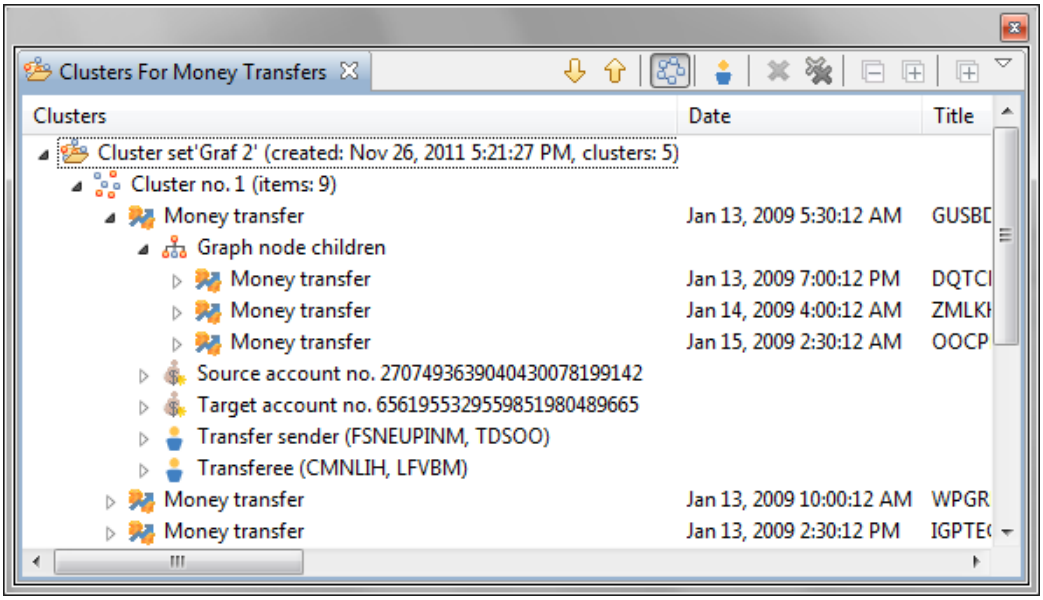

Figure 5: View with the clusters found

Figure 5 shows a cluster named *Cluster no. 1*. The first node, which has three children, is expanded.

Figure 6 shows a cluster named *Cluster no. 1* in the schema diagram, whereas Figure 7 shows the same cluster in the timeline diagram. Figure 6 shows between which accounts money was sent. What is more, numbers on relations indicate how much money was sent between the accounts. Such diagrams are intended to illustrate a structure of money flows, and especially relations between the accounts, while timeline diagrams for clusters focus on order of made transfers. Due to the fact that the clusters that are found contain suspected and associated sequences of flows, diagrams with such clusters can be treated by the analyst as suspected groups of operations.

Figure 8 shows a view with frequent patterns. It displays a set named *Frequent set no. 1* with two elements in the form of source bank accounts. The elements represent accounts from which money was frequently sent. Frequent sets are grouped into collections with the same size. For each frequent pattern its support is displayed. The frequent items that are found can be highlighted in the currently opened diagram (see Figure 9). An item from the frequent set named *Frequent set no. 1* was highlighted in the schema

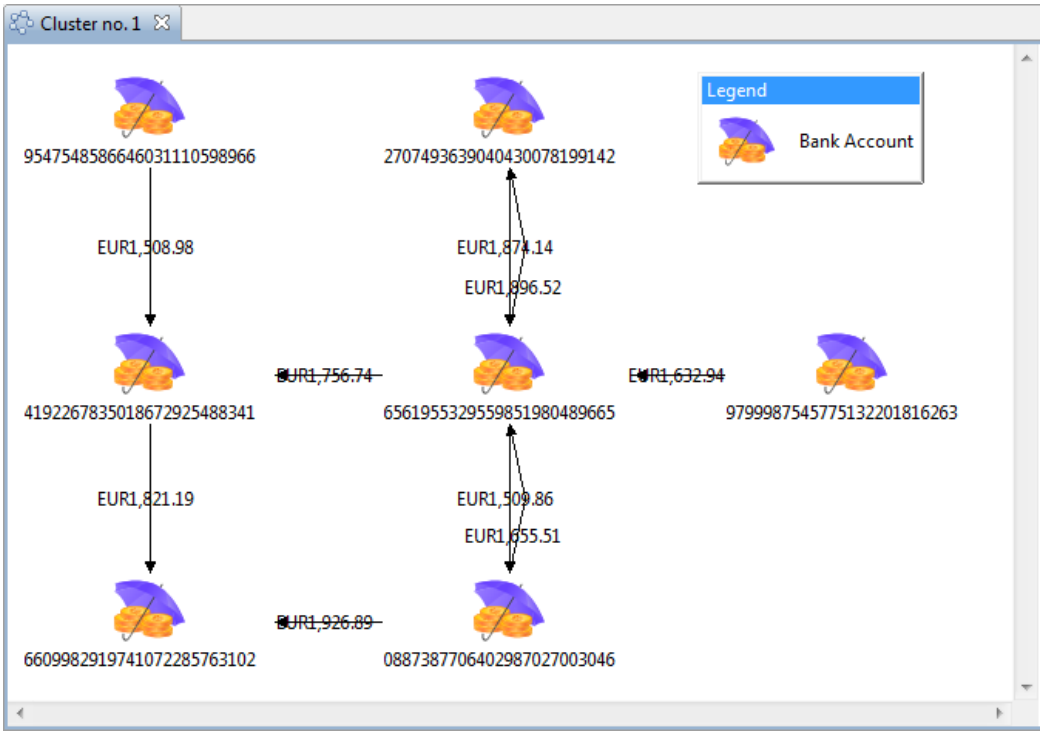

Figure 6: Schema diagram for the exemplary cluster

diagram with a cluster named *Cluster no. 1*.

The visualization allows for further analysis which is made by the user. The police analyst can interact with charts. He/she can add new nodes, remove or layout the existing ones. The nodes can be tagged, highlighted, searched out, etc.

Using charts the user can find suspicious sequences of transfers as well as identify (in the schema and timeline diagrams) the roles of individual persons who participate in money laundering. The frequent pattern mining between the source and target accounts has a significant meaning, because it allows to identify *distributive boxes* and *collective boxes* respectively.

6. Summary and Future Work

In this paper, a system supporting money laundering detection was presented. In the system, an importer of bank statements was implemented.

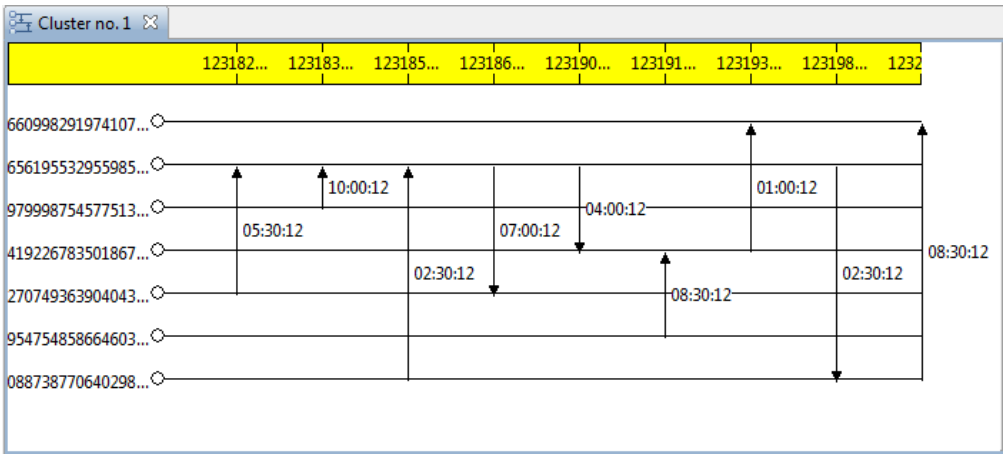

Figure 7: timeline diagram for an exemplary cluster

Imported data are used by the clustering algorithm, which is the basis for further analysis of money transfers. The clusters that are found have a graph structure. The algorithm takes account of such criteria as money flow condition, gathering amounts of money into a single account and commission received by entities involved in the money laundering process. Six important algorithms which mine frequent sets and sequences were implemented. These algorithms can provide all of the frequent patterns; closed or even maximal ones. The next possibility is the visualization of mined data in designed views and schema and timeline diagrams.

The system allows to build clusters which have a graph structure. Finding graphs, where there is at least one account with a large quantity of incoming transfers allows to find clusters with *gatekeepers*. The frequent pattern mining algorithm allows to find entities which frequently or always appear in transfers. In the case of *occasional offenders*, the system can find graphs with money flows deducting commission from the transfer amount, which is received by such entities.

What is more, the clustering algorithm allows to consider only money transfers whose amounts are in a given range. The user can specify the min-

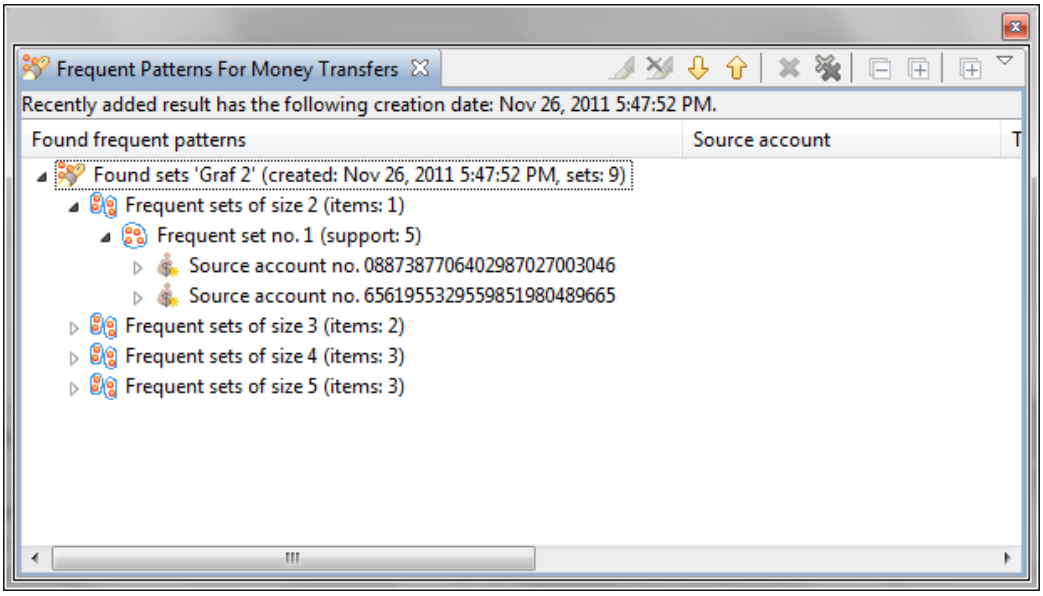

Figure 8: View with the frequent patterns found

imum and/or maximum amount. Hence, there is a possibility of considering only money transfers, for example, not greater than 15 000 €.

The time taken for the realization of money transfer is also taken into account. While creating a cluster, it is checked whether the duration time of the cluster is not greater than a given size of the time window. The place from where transfers were sent is not taken into consideration, due to the fact that this kind of data is not specified in bank statements.

The implemented system can be used successfully to detect money laundering when offenders use the following methods:

- Distributive boxes—the frequent set mining algorithms can be used to find accounts which frequently appear in clusters.
- Collective boxes—as in the case of *distributive boxes*, accounts which appear frequently in clusters can be found.
- Structuring—properly adjusting a range of amount from money transfers taken to find clusters, money transfers with small amounts can be considered.

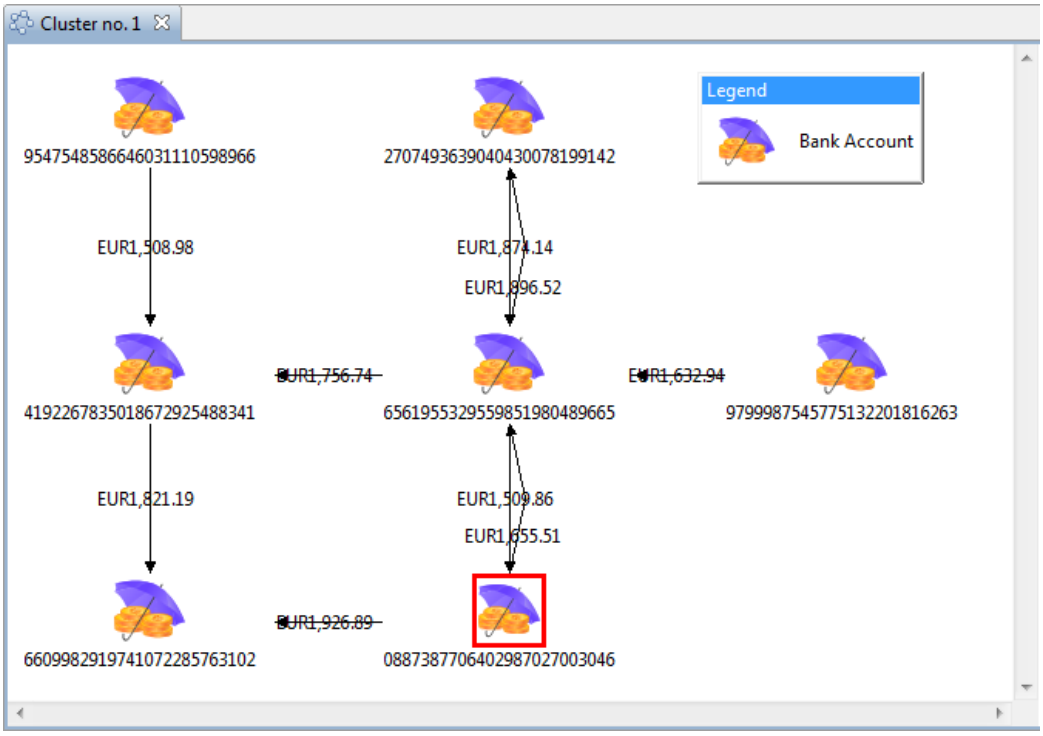

Figure 9: The highlighted element of frequent set

- Fast payment or payoff—properly adjusting the size of time window, fast payments and withdrawals can be found. When this kind of laundering is detected, finding money transfers which cause "zeroing" of a source account can be helpful.

Current achievements do not exhaust the analysis of money laundering. In the future, it is planned to implement the following functionalities of the presented system:

- Automatic generation of profiles of suspected companies/organizations typical activities, including titles of money transfers, time/date of making money transfers, amount of money being usually transferred, etc. (company/organization profile generation module).

- Learning from decisions made by the user interacting with the system (user model generation module).

Acknowledgments

The research presented in this paper was partially supported by the Polish National Center of Research and Development grant No. O ROB 0008 01.

References

Black WJ, Jowett S, Mavroudakis T, McNaught J, Theodoulidis B, Vasilakopoulos A, Zarri GP, Zervanou K. Ontology-enablement of a system for semantic annotation of digital documents. In: Handschuh S, editor. 4th International Workshop on Knowledge Markup and Semantic Annotation (SemAnnot 2004). 2004. p. 38–45.

Chang CH, Lui SC. IEPAD: information extraction based on pattern discovery. In: Proceedings of the 10th international conference on World Wide Web. New York, NY, USA; WWW '01; 2001. p. 681–8.

Chawathe S, Garcia-Molina H, Hammer J, Ireland K, Papakonstantinou Y, Ullman JD, Widom J. The TSIMMIS project: Integration of heterogeneous information sources. In: Proceedings of the 16th Meeting of the Information Processing Society of Japan. 1994. p. 7–18.

Chlebowicz P, Filipkowski W. Analiza kryminalna. Aspekty kryminalistyczne i prawnodowowe [Criminal Intelligence Analysis. Forensics and Procedural Aspects]. Warszawa: Wolters Kluwer, 2011. In Polish.

van Duyne PC. Foreword: Greasing the organisation of crime in europe. In: van Duyne PC, von Lampe K, Newell JL, editors. Criminal finances and organized crime in Europe. Nijmegen: Wolf Legal Publishers; 2003. p. 1–19.

Filipkowski W. Przeciwdziałanie przestępczości zorganizowanej w aspekcie finansowym [The financial aspects of counteracting organized crime]. Kraków: Zakamycze Kantor Wydawniczy, 2004. In Polish.

Gao S, Xu D, Wang H, Wang Y. Intelligent anti-money laundering system. In: Service Operations and Logistics, and Informatics. 2006. p. 851–6.

Gao Z. Application of cluster-based local outlier factor algorithm in anti-money laundering. In: Management and Service Science. 2009. p. 1–4.

Garofalakis M, Gionis A, Rastogi R, Seshadri S, Shim K. XTRACT: a system for extracting document type descriptors from xml documents. SIGMOD Rec 2000;29:165–76.

Grabarczyk G. Przestępczość gospodarcza na tle przemian ustrojowych w Polsce [Organized crime on the background of regime changes in Poland]. Toruń: TNOiK, 2002. In polish.

Grahne G, Zhu J. Efficiently using prefix-trees in mining frequent itemsets. In: Goethals B, Zaki MJ, editors. FIMI '03, Frequent Itemset Mining Implementations, Proceedings of the ICDM 2003 Workshop on Frequent Itemset Mining Implementations, 19 December 2003, Melbourne, Florida, USA. 2003. p. 123–32.

Grahne G, Zhu J. Fast algorithms for frequent itemset mining using fp-trees. IEEE Transactions on Knowledge and Data Engineering 2005;17(10):1347–62.

Grumbach S, Mecca G. In search of the lost schema. In: Proceedings of the 7th International Conference on Database Theory. London, UK: Springer-Verlag; ICDT '99; 1999. p. 314–31.

Górniok O. Przestępstwa gospodarcze, Rozdział XXXVI i XXXVII kodeksu karnego, Komentarz [Economic crimes, chapters XXXVI and XXXVI of the Penal Code, A commentary]. Warszawa: C. H. Beck, 2000. In polish.

Hammer J, Garcia-molina H, Cho J, Aranha R, Crespo A. Extracting semistructured information from the web. In: In Proceedings of the Workshop on Management of Semistructured Data. 1997. p. 18–25.

Han J, Pei J, Yin Y. Mining frequent patterns without candidate generation. In: Chen W, Naughton JF, Bernstein PA, editors. Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA. ACM; 2000. p. 1–12.

Hsu CN, Dung MT. Generating finite-state transducers for semi-structured data extraction from the Web. Inf Syst 1998;23:521–38.

Jun T. A peer dataset comparison outlier detection model applied to financial surveillance. In: Proceedings of the 18th International Conference on

Pattern Recognition - Volume 04. Washington, DC, USA: IEEE Computer Society; ICPR '06; 2006. p. 900–3.

Kingdon J. AI fights money laundering. IEEE Intelligent Systems 2004;19:87–9.

Kuhlins S, Tredwell R. Toolkits for generating wrappers. In: LNCS. Springer; 2002. p. 184–98.

Kushmerick N, Weld DS, Doorenbos RB. Wrapper induction for information extraction. In: IJCAI (1). 1997. p. 729–37.

Le-Khac N, Markos S, Kechadi MT. Towards a new data mining-based approach for anti money laundering in an international investment bank. In: International Conference on Digital Forensics and Cyber Crime (ICDF2C 2009). Albany, New York, USA: Springer Verlag; number 31 in LNICST; 2009a. p. 77–83.

Le-Khac NA, Markos S, Kechadi MT. A heuristics approach for fast detecting suspicious money laundering cases in an investment bank. In: Proceedings of the International Conference on Software and Knowledge Engineering (ICSKE 2009). Bangkok, Thailand; 2009b. .

Liu X, Zhang P, Zeng D. Sequence matching for suspicious activity detection in anti-money laundering. In: Proceedings of the IEEE ISI 2008 PAISI, PACCF, and SOCO international workshops on Intelligence and Security Informatics. PAISI, PACCF and SOCO '08; 2008. p. 50–61.

Luell J. Employee Fraud Detection under Real World Conditions. Ph.D. thesis; University of Zurich; 2010.

Lv LT, Ji N, Zhang JL. A RBF neural network model for anti-money laundering. In: ICWAPR '08 International Conference. 2008. p. 209–15.

Michna P. Wykorzystanie analizy kryminalnej w ściganiu przestępstw. Aspekty ekonomiczno-finansowe [Using criminal analysis to prosecute crimes. Economic and financial aspects]. In: Lelental S, Potakowski D, editors. Pozbawianie sprawców korzyści uzyskanych w wyniku przestępstwa [Depriving perpetrators the profits of their crimes]. Szczytno: Wydawnictwo Wyższej Szkoły Policji; 2004. p. 67–86. In Polish.

Muslea I, Minton S, Knoblock C. A hierarchical approach to wrapper induction. In: Proceedings of the third annual conference on Autonomous Agents. New York, NY, USA; AGENTS '99; 1999. p. 190–7.

Nekvasil M. The use of ontologies in wrapper induction. In: DATESO. volume 235; 2007. p. 132–6.

Paynich R, Hill B. Fundamentals of Crime Mapping. Sudbury: Jones and Bartlett Publishers, 2010.

Pei J, Han J, Mortazavi-Asl B, Pinto H, Chen Q, Dayal U, Hsu MC. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In: Proc. of the 17th International Conference Data Engineering. Los Alamitos, CA, USA: IEEE Computer Society; 2001. p. 215–24.

Prengel M. Środki zwalczania przestępczości prania pieniędzy w ujęciu prawnoporównawczym [Measures to fight money laundering crimes from the comparative law perspective]. Toruń: TNOiK, 2003. In Polish.

Qifeng Y, Bin F, Ping S, Coll. E. Study on anti-money laundering service system of online payment based on union-bank mode. In: Wireless Communications, Networking and Mobile Computing. 2007. p. 4991–4.

Raza S, Haider S. Suspicious activity reporting using dynamic bayesian networks. Procedia CS 2011;3:987–91.

Skorupka J. Prawo karne gospodarcze, Zarys wykładu [Penal economic law, a lecture outline]. 2nd ed. Warszawa: LexisNexis, 2007. In Polish.

Spiliopoulou M, Müller RM, Brunzel M, Rinaldi F, Hess M, Dowdall J, Black WJ, Theodoulidis B, McNaught J, Bernard L, Zarri GP, Orphanos G, King M, Persidis A. Coupling information extraction and data mining for ontology learning in PARMENIDES. In: Fluhr C, Grefenstette G, Croft WB, editors. RIAO. CID; 2004. p. 156–69.

Tomaszewski T. Dowód naukowy przed sądem [scientific evidence before the court]. In: Girdwoyń P, editor. Prawo wobec nowoczesnych technologii, Materiały z konferencji Wydziału Prawa i Administracji, Uniwersytetu Warszawskiego, która odbyła się 2 marca 2007 roku [Law in view of modern technologies, materials from the conference of the Faculty of

Law and Administration of the Warsaw University held on 2 March 2007]. Warszawa: Liber; 2008. .

United Nations Office on Drugs and Crime . United Nations Convention Against Transnational Organized Crime and the Protocols Thereto. New York: United Nations, 2004.

Wang J, Han J. Bide: Efficient mining of frequent closed sequences. In: Data Engineering, 2004. Proceedings. 20th International Conference on. Los Alamitos, CA, USA: IEEE Computer Society; 2004. p. 79–90.

Wang J, Han J, Pei J. Closet+: searching for the best strategies for mining frequent closed itemsets. In: KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining. New York, NY, USA: ACM Press; 2003. p. 236–45.

Wang S, Yang J. A money laundering risk evaluation method based on decision tree. In: Proceedings of the 2007 International Conference on Machine Learning and Cybernetics. Hong Kong; 2007. p. 283–6.

Wong W. Learning Lightweight Ontologies from Text across Different Domains using the Web as Background Knowledge. Ph.D. thesis; University of Western Australia; 2009.

Wójcik JW. Przeciwdziałanie praniu pieniędzy [Counteracting money laundering]. Kraków: Zakamycze Kantor Wydawniczy, 2004. In Polish.

Wójcik JW. Przeciwdziałanie finansowaniu terroryzmu [Counteracting the financing of terrorism]. Warszawa: Wolters Kluwer Polska – Oficyna, 2007. In Polish.

Zhang ZM, Salerno JJ, Yu PS. Applying data mining in investigating money laundering crimes. In: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining. New York, NY, USA; KDD '03; 2003. p. 747–52.

Zhu T. An outlier detection model based on cross datasets comparison for financial surveillance. In: Proceedings of the 2006 IEEE Asia-Pacific Conference on Services Computing. Washington, DC, USA: IEEE Computer Society; 2006a. p. 601–4.

Zhu T. Suspicious financial transaction detection based on empirical mode decomposition method. In: Proceedings of the 2006 IEEE Asia-Pacific Conference on Services Computing. Washington, DC, USA: IEEE Computer Society; 2006b. p. 300–4.