Data Mining, Knowledge Acquisition

DATABASE SERVER USAGE IN THE SOCIAL NETWORKS ANALYSIS

Katarzyna Harężlak

***Abstract**: In the field of computer science, topics regarding analysis of social networks grows more and more popular. It stems from the fact that the knowledge hidden in connections between people might enable us to answer many questions concerning various areas of life. When analyzing social networks, we are usually interested in such information as – network structure, how it is managed and what is the scheme of flow of goods or information. When starting a study of processes taking place in social networks it can be noticed that a lot of data subject to analysis is stored in structures of databases. Thus, database server's capabilities in terms of social network examining and assigning roles to its members are worth analysing. The aim of this paper is to present studies regarding using a database serve to gather data on social networks and mechanisms it provides to specify the type of connections between network elements.*

One of the tasks of social network analysis is a graphic visualization of connections between various objects existing in the field of interest of a conducted research. The most convenient representation of such is a graph. Ensuring a possibility to gather data for a permanent storage and at the same time effectively operate on them required designing appropriate data structures. It was decided, that the relational database meets the demands. The basic structure of the proposed data model covers vital graph elements along with a description of their qualities.

Analysis of social networks presented as graphs poses a challenge when it consists of a huge number of nodes and connections between them. One of the most important functions for presenting a social network is to make it possible to locate and visualize only fragments of the whole graph in order to make analyses more effective. In the research, the mechanism for limiting set of nodes (creating sub-graphs, called paths for the purpose of this research) and for aggregating elements of a graph were prepared. Moreover, problem of graph search was also taken into consideration. The two algorithms, BFS and Dijkstra's, were implemented as a database procedures. Effectiveness of the operation of the first of algorithms was compared with its implementation in the. NET environment as well.

***Keywords**: social network analysis, database server, graphs, shortest paths.*

***ACM Classification Keywords**: H. Information Systems H.3 INFORMATION STORAGE AND RETRIEVAL H.3.3 Information Search and Retrieval*

Introduction

In the field of computer science, topics regarding analysis of social networks grows more and more popular. It stems from the fact that the knowledge hidden in connections between people might enable us to answer many

questions concerning various areas of life [Chau, 2006, Giran, 2002, Razmerita, 2009]. When analyzing social networks, we are usually interested in such information as – network structure, how it is managed and what is the scheme of flow of goods or information [Newman, 2004]. Regardless of size and purpose of such network, there can always be made a distinction between the core and the periphery of the network, depicting the division of power, influence or status inside the network [Williams, 2001]. A more in-depth analysis of tasks performed in a social network leads to a more precise specification of roles and their assignment to particular person. It is used to describe the behaviour of a node in relation with the rest of a network as a whole. Among the roles, we can enumerate Organisers, Isolators, Communicators, Guards, Expanders, Supervisors, Connectors, Soldiers, Recruits, Neutral [Piekaj, 2007] and Ambassadors, Big Fish, Bridge [Scripps, 2007]. In order to specify a role of a particular node in a network many measurements have been created, the most popular of which are degree, closeness and betweenness [Guangming, 2009, Shaikh, 2006].

Degree centrality is a measure describing the number of connections of a particular node with other nodes in a network. High value of this measure may indicate that the person connected with such a node may be a leader of a group.

Betwenness centrality is described by a number of shortest paths between each pair of nodes crossing a particular node, which indicates that this node plays a vital role in transmitting information – they can thus be thought of as broker or gatekeeper.

Closeness centrality is a measure describing the distance between a particular node and other nodes. It is computed using a number of shortest paths connected with this node. High value of this measure also can indicate leaders of a group.

When starting a study of processes taking place in social networks it can be noticed that a lot of data subject to analysis is stored in structures of databases. Thus, database server's capabilities in terms of social network examining and assigning roles to its members are worth analysing. The aim of this paper is to present studies regarding using a database serve to gather data on social networks and mechanisms it provides to specify the type of connections between network elements.

The representation of a social network

One of the tasks of social network analysis is a graphic visualization of connections between various objects existing in the field of interest of a conducted research. The most convenient representation of such is a graph G, with a weight function G=(V,E:w), where V stands for a set of nodes, E is a set of connections, w is a function mapping each connection (u,v) € E, to a weight w_{uv}, which shows the strength of connection between u and v [Yang, 2006]. Each node represents a person. Connection between nodes represents a type of a relation between two people. These graphs, due to the multiple analysis, should have a feature of durability, which means that a once-built graph should be kept for a renewed study.

Graph data model

Ensuring a possibility to gather data for a permanent storage and at the same time effectively operate on them required designing appropriate data structures. It was decided, that the relational database meets the demands. The basic structure of the proposed data model covers vital graph elements along with a description of their qualities. In this structure, the graph itself is an superior element. Graphs created by the user may concern various domains (billings, criminal network, connections between companies' managers), and within the domain various groups of objects represented by nodes (person, car, phone number). The last ones are usually described by a name, but, depending on the domain of the graph, their set of features may be different. For example, different figures are used to describe people and phone calls. The situation is very similar when

it comes to edges. There are two basic features, the direction of the connection and its weight, defining four types of graphs (undirected and unweighted, directed, weighted, directed and weighted).The other features are dependent on the type of graph. Taking this dependencies into account, a special model of data was elaborated, ensuring flexibility in determining a graph structure.

Graph paths and their versions

Analysis of social networks presented as graphs poses a challenge when it consists of a huge number of nodes and connections between them [Buja, 2008]. The main problem faced by software visualising such networks is selecting appropriate methods of presenting them, ensuring proper deployment of nodes in a two or three-dimensional scope [Xu, 2005]. By proper, guaranteeing similar length of all edges and a minimal number of edges crossings is meant [Kulmar, 1994]. Furthermore, one of the most important functions for presenting a social network, is to make it possible to locate and visualize only fragments of the whole graph in order to make analyses more effective. Such fragments can be obtained by [Harezlak, 2010]:

- zooming in on a neighbourhood of a particular node,
- limiting set of nodes (creating sub-graphs, called paths for the purpose of this research),
- aggregating elements of a graph.

Some of these operations can be performed by an analytical application and some part of the functionality can be transferred to a database server. In the research, the mechanisms for the two last were prepared. Along with this work came the need to extend the graph with elements enabling keeping track of changes in graph structure resulting from an aggregation of its nodes. This operation entails an aggregation of edges connecting the aforementioned elements. This is why two tables **superiorNodes** and **superiorEdges** where added to the existing objects to enable collecting Master-Slave relationships. Their task is to gather all information on merged nodes or edges and new graph elements created on their basis. Thereby, in the object representing graph elements a special attribute, **AggregationIndex**, was added. It indicates on which level of graph development the element was created (positive value of the index) or aggregated (negative value of the index). Let us analyze the graph presented in the figure 1.

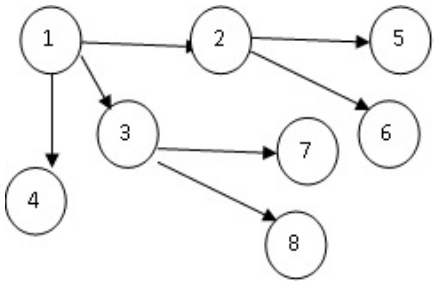


Fig. 1. A sample graph

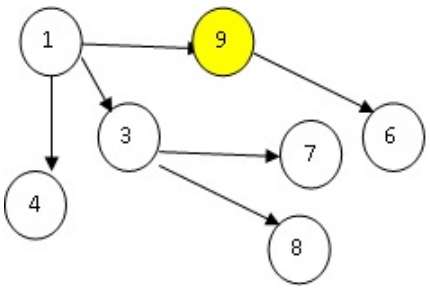


Fig. 2. A sample graph with integrated nodes

When a graph is being created, index attribute of all nodes is set to 1. In case when a user decides to aggregate nodes 2 and 5, a new node - number 9 – appears, and nodes 2 and 5 are no longer visible to the user, as presented in the figure 2.

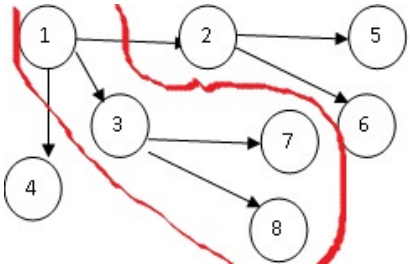
The described operations causes a change of index values for nodes 2 and 5 – it is set to -2 (elements were deleted in the second version of the graph), and node number 9 is added to a database with the aggregation index set to 2 (the element was created in the second version of the graph). The actions being performed when deleting edges 1-2,2-5 and 2-6, and creating 1-9 and 9-6 are very similar.

During the further research, a new functionality, allowing user for fragmentary analysis of a graph, was introduced as well. This way, idea of a graph **path** and its **version** was also proposed. A path is any sub-graph analyzed by a user at a given moment (figure 3).

With this subgraph a user can aggregate nodes and edges, creating consecutive versions of a path. Elements belonging to a version of a particular path have their own state, which equals the aggregation index from the above-mentioned solution. The state, if positive, corresponds to the version in which the element was created. If negative, it indicates that the element is deleted and which version of the path it was aggregated in.

In accordance with the data model proposed for performing this kind of operations, a full graph is at the same time its first path. Thus, the path presented in figure 3 is numbered 2, with a version number 1.

Described mechanisms are included in special database procedures, which input data consists of:



- graph identifier,
- path identifier,
- current version of the path,
- number and identifiers of aggregated nodes.

Fig. 3. A sample graph with a marked path

The effect of the procedure execution is creating another version of a chosen path of a particular graph and elements connected with it. It consists of a logical removal of the aggregated elements by changing their status and generating new objects instead, marking the version of their creation.

Experiments

The algorithms operating on a graph nodes implemented in the database server were tested using a sample data set. For the purpose of the test, data was imported to a model representing a graph. People became nodes, and connections between them edges. The import was performed taking directed connections into account. A database server chosen for the research was MS SQL Server 2008. The sample application developed during research was used for graphic presentation of the graph.

Proposed algorithms were tested in many variants of paths for the graph consisting of 1269744 nodes 5196852 edges. One of variants concerned five nodes with 107934, 110241, 142458, 142846, 132206 identifiers (marked in the figure 4 with graphic symbols OSO-01331206, OSO-01339508, OSO-01330596, OSO-01339610 OSO-01336904). They represent multi-level tree structure. Nodes on the last level of this structures (OSO-01331206, OSO-01339508) do not have any connections. The task of the node 132206 (OSO-01336904) is to connect a newly created node and the rest of the graph. Before aggregation it is connected to only one node from the aggregated structure. In such case, running the procedure resulted in (table 1):

Table 1. Nodes table

Node_ID	AggregationIndex
12231349	2
107934	-2
110241	-2
132206	1
142458	-2
142846	-2

- creation of one new node identified by 12231349 with aggregation index equal to 1,
- change of this index for aggregated nodes.

Table 2. SuperiorNodes table

SuperiorNode_ID	subordinateNode_ID
12231349	107934
12231349	110241
12231349	142458
12231349	142846

In the **SuperiorNodes** table four records can be found this time. Their aggregation brought to "life" a new node – 12231349 (table 2)

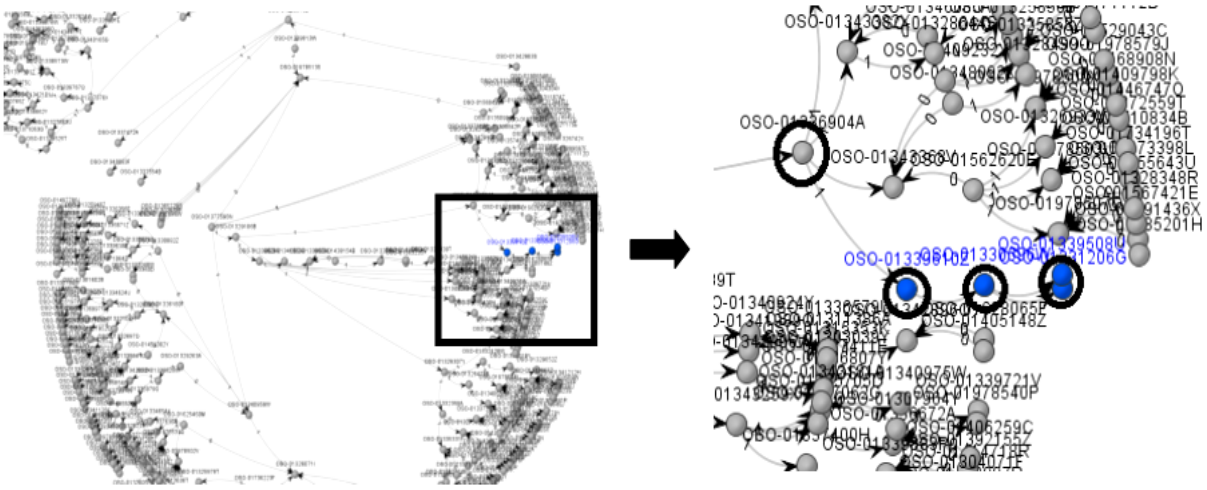


Fig. 4. Graphical representation of a network used in the first variant of tests

In the next presented step of testing the algorithms for nodes aggregation, a two-level sample of data was chosen, connected with the rest of the graph on both sides, characterized by various directions of connections. It consists of people with identifiers 131882, 132468, 156302 102566 (in the figure 5 OSO-01342822, OSO-01336979, OSO-01336825 OSO-01329263 respectively).

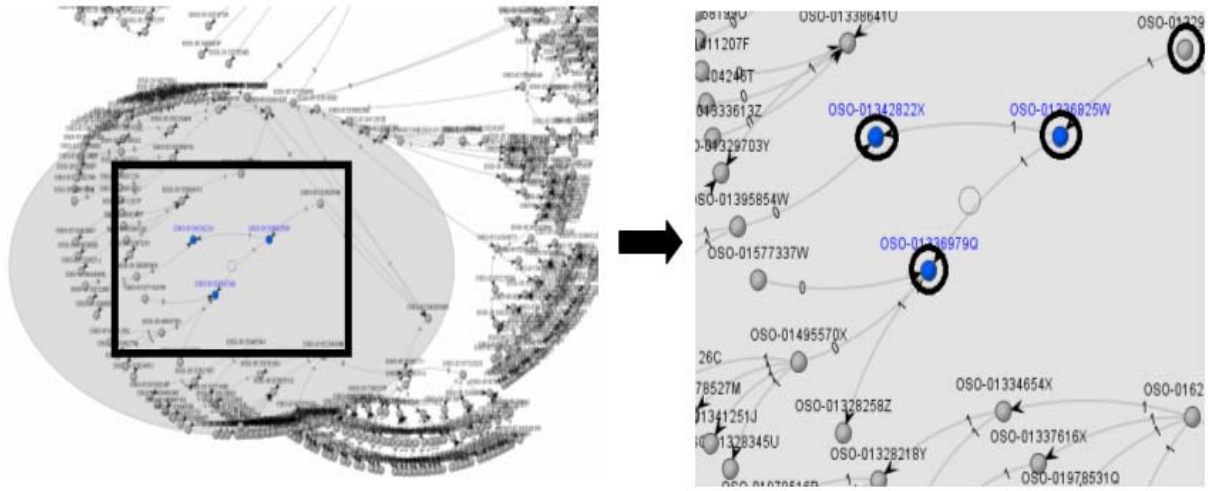


Figure 5. Graphical representation of a network used in the second variant of tests

Table 3. Nodes table

Node_ID	AggregationIndex
12231350	2
102566	1
131882	-2
132468	-2
156302	-2

Table 4. SuperiorNodes table

SuperiorNode_ID	SubordinateNode_ID
12231350	131882
12231350	132468
12231350	156302

The way in which the algorithm acted, in terms of operations on nodes, was identical with the cases described earlier. The aggregated nodes, 131882, 132468, 156302 are merged into one node 12231350 (table 4). Maintaining the index of aggregation is also conducted in a similar way, which is shown in the table containing data on nodes (table 3)

Table 5. SuperiorEdges table

SuperiorEdge_ID	subordinateEdge_ID
3052	2756
3053	2757
3054	2520
3055	2758
3056	2755

The aggregation of edges was conducted in a different way. The aggregated structure was connected with the rest of the graph by five edges, each ending in a different node. It caused the creation of five new edges, starting in these nodes and ending in the newly created one, preserving the primary binding direction of connection. The edges located in the middle of the aggregated structure had no influence on the number of new elements. Five new records were added to the table **SuperiorEdges** (table 5) and to the table representing data on graph edges (table 6). In the latter, aggregation index and weight of an edge were appropriately modified. In all the cases, their weight was inherited after removed edges.

Table 6. Edges table

EdgeID	Edge_A_ID	Edge_B_ID	Weight	AgregationIndex	Direction
2520	102566	131882	1	-2	1
2669	131882	156302	1	-2	1
2670	131882	132468	1	-2	1
2755	156302	401585	1	-2	0
2756	132468	98979	1	-2	1
2757	132468	1281444	1	-2	0
2758	132468	1863045	1	-2	0
3052	12231350	98979	1	2	1
3053	12231350	1281444	1	2	0
3054	102566	12231350	1	2	1
3055	12231350	1863045	1	2	0
3056	12231350	401585	1	2	0

Operations on graphs representing a social network

A graph search or in other words passing through a graph is an activity consisting of visiting (in some systematic, regular way) all graph nodes in order to gather relevant information [Cormen, 2001].There can be indicated two

most popular algorithms used for this purpose – BFS (breadth-first search) and DFS (depth-first search). Despite the fact that the analysis of database servers' mechanisms in terms of implementing both algorithms proved it to be theoretically possible, using the second one turned out to be impossible in practice. The cause of this was the necessity to use recursive queries, which were incapable of computing such a huge amount of data, encountered in social analysis. Not surprisingly, a decision was made to implement the BFS algorithm.

BFS algorithm is one of the simplest algorithms for searching graphs. The edges of the graph are systematically looked into, in order to visit every node accessible from the source node s. At the same time, distance (smallest number of edges) from source to each node is calculated. The effect of acting of the algorithm is also a breadth-first search tree from the s root to all accessible nodes. A path in the tree between s and node v is the shortest path between these two nodes in the graph. In order to make implementation of the algorithm, as a database procedure, possible, data model was extended with additional tables storing achieved tree and calculated paths.

In many cases of graph analysis searching the graph is performed in order to find the shortest path between two particular nodes. Among many available algorithms in this scope, Dijkstra's algorithms was chosen to be implemented [Cormen, 2001].

Dijkstra's algorithm determines the shortest paths starting in a particular node. It is performed for every node in the graph and requires weights of edges of a graph to be known beforehand. In the first step of the algorithm, distance between source node and each of the rest of the nodes is set to infinity. Next, during the process, the queue of neighbours of the currently analyzed node is created. For this purpose a special database table was created. Consecutive steps of the algorithm compare the distance between elements of the queue and the source node (which in the beginning equals infinity) with the weight of the edge connecting the current node and the analyzed one from the queue. The weight is obtained from the graph representing the social network. If the sum of this weight and distance from the current node to the source is smaller, it replaces the previous distance. The next current node is always selected acquisitively – meaning that it is always the unvisited node, to which the distance from the source is shortest. The algorithm effects in creating a set of shortest paths from the source node to each node in the graph. These paths are stored in the table of a graph data model. Thus, they can be easily accessed if necessary.

Experiments

The chosen algorithms were tested with usage of the aforementioned graph. In the first step, BFS algorithm was used to check the consistency of the graph. Due to the fact that the graph consisted of huge number of elements, call of the database procedure was parameterized in terms of numbers of analyzed edges and neighbours.

Subgraphs, which consistency was proven, became the input data for the procedure implementing the second algorithm. The obtained result for sample procedure call is presented in the table 7.

Table 7. Format of the shortest path from a sample node (11107)

Node_A_ID	Node_B_ID	distance from the starting node
111107	113781	1
111107	262461	1
113781	101261	2
113781	113280	2
101261	129513	3
129513	148320	4
129513	158543	4
129513	679012	4
148320	102566	5
148320	108960	5
148320	321729	5

The weight of every edge was set to 1.

The elaborated mechanisms were also tested in terms of efficiency. For this purpose, BFS algorithm was implemented on the .NET platform using C# language. The same tests, which were run for database procedures, were repeated for the program constructed in this way. Both procedures were run in the same environment. Then, execution times of both procedures (with the same input data) were compared. Obtained results are presented in the figure 6. Their analysis clearly indicates that database server procedure was more efficient in checking consistency of a graph than the .NET one. It is worth to mention that .NET procedure connected to database server to achieve input data and meet the requirement concerning permanency of obtained results.

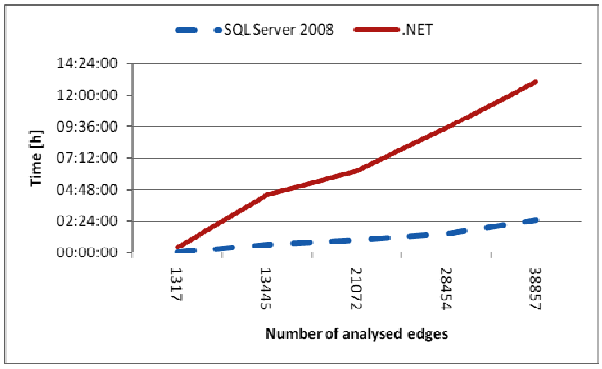


Fig. 6. Comparison of execution time of .NET and SQL Server 2008 procedures

Conclusion

The aim of the research presented in the article was to assess the possibility of using database server resources to support social network analysis. There were two reasons of taking such approach. First of them was connected with the fact that information about objects constituting a social network are usually stored in database tables. The second reason resulted from the need for permanent storage of data on the previously computed characteristics concerning structure of such network. Adopting databases for such purpose seems to be the right concept. Within the research, a special data model was elaborated. Its architecture is universal enough to make usage of this model in analysis of social networks from various domains. The model consists of two groups of objects. In the first one, data on network elements is stored while the second consists of objects used to determine its characteristics. The second one is also exploited by database procedures implementing algorithms elaborated during the research, allowing for a change of social network structure (creating subgraphs, aggregation nodes and edges) and its searching (BFS and Dijkstra's algorithms). In order to check the correctness of the created mechanisms, tests for a sample social network were executed. They proved the preliminary assumption to be true. Furthermore, additional efficiency studies were performed, comparing the effectiveness of the mechanisms implemented on the database server with the corresponding ones created in the .NET platform. These tests also confirmed the rightness of the direction of the research.

The results obtained in the research are highly encouraging to further activity. In the next steps, elaborating solutions, allowing for usage of database servers in order to determine the roles of objects creating a social network, is planned. Moreover, what is also planned to be used for this purpose is multi-agent systems.

Bibliography

[Buja, 2008] A.Buja, D.F.Swayne, M.Littman, N.Dean, H.Hofmann, L.Chen. Interactive Data Visualization with Multidimensional Scaling, Journal of Computational and Graphical Statistics, 2008, Vol 17 (2): pp. 444–472.

[Chau, 2006] M.Chau,J.Xu. A Framework for Locating and Analyzing Hate Groups in Blogs. PACIS 2006 Proceedings.

[Cormen, 2001] T.H.Cormen, C.E.Leiserson, R.L.Rivest, C.Stein. Introduction to Algorithms. The MIT Press, 2nd revised edition, September 2001.

[Giran, 2002] M.Girvan, M.E.J.Newman. Community structure in social and biological networks, Proceedings of the National Academy of Sciences, Vol. 99, No. 12, pp. 7821-7826, 2002

[Guangming, 2009] T.Guangming, T.Dengbiao, S.Ninghui. AParallel Algorithm for Computing Betweenness Centrality Parallel Processing, 2009. ICPP '09. International Conference on, 2009,pp. 340 - 347.

[Harezlak, 2010] K.Harężlak, M.Kozielski. The methods of criminal network analysis (Metody analizy sieci kryminalnych). STUDIA INFORMATICA. Volume 31, Number 2A (89), 2010

[Kulmar, 94] A.Kumar, R.H.Fowler. A spring modeling algorithm to position nodes of an undirected graph in three dimensions, Technical Report CS-94-7

[Newman, 2004] M.E.J. Newman. Detecting community structure in networks, Eur. Phys. J. B 38, 321–330 (2004).

[Piekaj, 2007] W.Piekaj , G.Skorek, A.Zygmunt, J.Koźlak Environment for identifying behavioral patterns using social networks (Środowisko do identyfikowania wzorców zachowań w oparciu o podejście sieci społecznych). Technologie Przetwarzania Danych, TPD 2007.

[Razmerita, 2009] L.Razmerita, R.Firantas New Generation of Social Networks Based on Semantic Web Technologies: the Importance of Social Data Portability, http://academic.research.microsoft.com/Publication/5968829/new-generation-of-social-networks-based-on-semantic-web-technologies-the-importance-of-social-data, 2009.

[Scripps, 2007] J.Scripps, P.N.Tan, and A.H.Esfahanian. Node roles and community structure in networks. In WebKDD/SNA-KDD '07: Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis, ACM, 2007, pp 26□35..

[Shaikh, 2006] M.A.Shaikh, J.Wang. Investigative Data Mining: Identifying Key Nodes In Terrorist Network. Multitopic Conference, 2006. INMIC '06. IEEE, pp 201 - 206.

[Williams, 2001] P.Williams. Transnational Criminal Networks. http://www.rand.org/pubs/monograph_reports/MR1382/MR1382.ch3.pdf., 2001.

[Xu, 2005] J.Xu, H.Chen. CrimeNet Explorer: A Framework for Criminal Network Knowledge Discovery, ACM Transactions on Information Systems (TOIS), 23(2), pp. 201-226, 2005.

[Yang, 2006] C.C.Yang, L.Nan, M.Sageman. Analyzing the Terrorist Social Networks with Visualization Tools, Intelligence and Security Informatics, Lecture Notes in Computer Science, Vol. 3975, 2006 pp.331-342.

Acknowledgements

Project financed from the funds for learning in years 2010 - 2012 by a research and development grant number O R00 0113 12. I am grateful to the copartner WASKO S. A company for letting me exploit the screenshots of the sample application used in my figures

Authors' Information

Katarzyna Harężlak *– Silesian University of Technology; e-mail: katarzyna.harezlak@polsl.pl*

Major Fields of Scientific Research: Distributed and Mobile Databases, Software Engineering, Agent Systems, Social Networks