The HiTS/ISAC Social Network Analysis Tool

Hirad Asadi, Christian Mårtenson, Pontus Svenson, Magnus Sköld
{hirasa,cmart,ponsve}@foi.se, skoeld@gmail.com
Swedish Defence Research Agency
Gullfossgatan 6, 164 90 Stockholm, Sweden

Abstract—**We describe the Social Network Analysis Tool (SNA-Tool) developed in the HiTS/ISAC (Highway to Security: Interoperability for Situation Awareness and Crisis Management) EU PASR (Preparatory Action for Security Research) project. The tool enables the construction of social networks from database sources using advanced filters on three different levels. The tool also allows visualization of the results for analysis purposes. The tool was demonstrated and evaluated in a scenario involving customs and police operators working to detect smuggling of dangerous substances and also for detecting and preventing a terrorist attack.**

keywords Social network analysis, SNA, predicate filtering, information fusion

I. Background

The continuous evolution of information technology brings new possibilities of studying human sociological interactions. These interactions can be described as relations between different actors. Data is the source of information and much of human activities are stored in different databases around the world.

In many cases we would like to examine how some data is related to other data and visualize the results. If a person exist in a car registry, can that data be combined with e.g. the person's health insurance registry or tax registry and so on? Can social network data be created based on relational data sources and then visualized?

Law-enforcement analysts in particular today need advanced computer support tools in order to work efficiently on the data associated with cases. They also need tools for interoperability and collaboration at a distance. In the EU PASR project HiTS/ISAC, one of the aims was to demonstrate how collaborative, distributed tools for social network analysis could be used to help police analysts to investigate an on-going criminal case and help the users achieve situation awareness.

The vision of HiTS/ISAC is a more secure Europe through prevention of terrorism and organized crime. Superior situation awareness and cross-border interoperability are key enablers, leading to new technical and operational methods to work, train and co-operate across Europe. Today, information in databases at law enforcement authorities is distributed across Europe. The information is not easily available to other authorities in Europe, especially not on-line.

The objective of HiTS/ISAC is to enable information analysis and data fusion from many different sources, through secure cross-border on-line group cooperation between authorities, in order to detect and provide early warnings for suspicious activities, be it communication between suspected criminals, or anomalous movement of persons, goods or money, etc. Some more details on the objectives and goals of HiTS/ISAC can be found in [1]. In this paper, we will describe the SNATool developed as part of the HiTS/ISAC project. The main research question that guided the development of the SNATool was to see if it was possible to design a tool for analysis and visualization of a filtered social network graph, where only those parts of the network that matched certain predicates were used.

II. Outline

In this paper we propose a tool (SNATool) that shows how social network data can be extracted, filtered on three different levels (using special filtering techniques) and later visualized. The focus will be on the last filtering level, which uses "predicates" for filtering. If we can represent human relational structures then we can perhaps understand how different social phenomenons occur. This knowledge can be adapted in such a way that in the future we can perhaps understand how terrorist networks occur and also where we can focus our counter-terrorism efforts.

This paper is outlined as follows. In section III we give an overview of the system architecture of the HiTS/ISAC SNATool. Sections IV and V describe the two components of the tool: the data extraction & filtering levels and the social network visualizing interface which is used for visualization and analysis. Finally, section VI gives a brief overview of the demonstration performed to evaluate HiTS/ISAC and section VII ends in a brief discussion.

III. System Architecture

The system architecture that we developed consists of different layers which are described in Figure 1 and Figure 2. The data layer constructs the network data from relational databases. Because databases contain vast amounts of data we have to filter out the majority of data some how. This is done through filtering on three different levels: the first level is by plain SQL, the second level is by a tool called Proximity [2] which uses a graphical querying method to detect patterns in data. The third level is by different node discovery algorithms which use "predicates", see section IV. Finally when we have a filtered graph (stored in the GraphML XML[1] [3] or

[1]eXtensible Markup Language

JUNG[2] [4] format) we can visualize and analyze it using the SNA visualization & analysis interface, see chapter V.

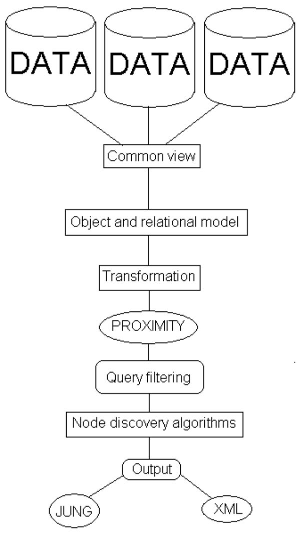


Figure 1. Data layer.

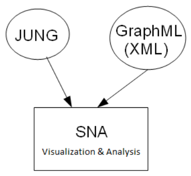


Figure 2. Visualization layer.

We would like to integrate different database sources into a single intermediate storage. This would allow us to process data easier without having to deal with multiple connections. There are different tools to overcome this problem, one of them is Denodo Virtual DataPort (DVDP) [5] which is considered in this paper. DVDP is a tool which allows us to integrate different database sources into a *common view*. This view basically allows us to see all the database sources as one database and therefore it is a practical tool for data integration.

IV. Data Extraction & Filtering

Creating the data structures and relations that define the networks is one of the most important stages in the analysis process: without the right data, analysis and visualization is meaningless.

Network data is used for creating graphs, but the process from relational data to network data is rather abstract. What data is important? What data should be extracted? How much data should be extracted? These are some questions that need to be answered in order to fully understand how network data can be constructed and finally visualized.

[2]Java Universal Network/Graph

A. From Relational Data to Network Graphs

Relational data stored in databases around the world are modeled in a special way in order to be meaningful. Many times relational data is stored in the second normalized form [6]. This form is showed in Table I, each person is identified by an "Id". Table II shows emails sent by persons from table I—both tables are normalized. Note that the identifying column does not generally need to be of numerical type in relational databases.

When data is stored in this structured way it is quite easy to transform it to a network graph since there almost exists an implicit network definition for nodes and edges within the tables. In this case each node could correspond to a row in Table I and each edge to a row in Table II, hence the result would be a network graph with 4 nodes and 4 edges, see Figure 3.

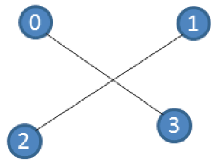


Figure 3. A network graph constructed using data from Table I & Table II.

The mappings shown so far are trivial, but how do we map tables that are not normalized? The easiest way is to normalize those tables and this procedure can e.g. be done with DVDP [5]. After normalization we can deal with tables as collections of objects which enables us to interpret some tables as "node-tables" and some as "edge-tables".

B. What data is important?

The data that is important is basically dependent on the scenario. If the scenario is to study how emails are sent back and forth between people, then we would e.g. be interested in data identifying each person and also data showing who sent an email to who. This is not something that is always trivial, for example, in the case of catching terrorists, individual data items might not be interesting, but rather the whole pattern that didn't seem to fit "normal". These patterns can be viewed as different relations between objects with different conditions. There are tools that help us create these patterns, one of them is Proximity [2] which can be used to create patterns that e.g. allow anomaly-detection, see Figure 4.

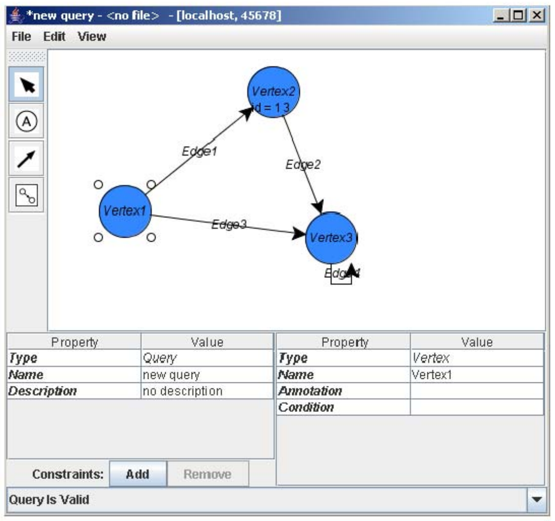
In order to gain information we must begin to classify data. Advanced automatic classification models [7] can be used to predict unknown data but in this paper the concept of classifying data is mainly defined by the user. After the correct data is classified and extracted we can produce different kinds of visualizations, especially between the same objects. This step is important because different visualizations give different information about the extracted data. Using different

Id	Name	Weight	Age	Gender
0	Mary	50	31	F
1	Alex	90	31	M
2	Fred	50	15	M
3	Hillary	55	31	F

Table I
SHOWS A PEOPLE TABLE WITH ID AS A UNIQUE IDENTIFIER.

From	To	Subject	Message	Sent
0	3	Hi!	Hello how are you...	2007-08-03 20:00:31
3	0	Re: Hi!	Im fine how are you...	2007-08-03 22:30:01
2	1	Meeting soon	Remember to schedule...	2007-04-03 12:10:53
1	2	Re: Meeting soon	Sorry busy, but I can...	2007-04-03 17:17:41

Table II
SHOWS A TABLE WITH SENT E-MAILS.



Property	Value
Type	Query
Name	new query
Description	no description

Property	Value
Type	Vertex
Name	Vertex1
Annotation	
Condition	

Query Is Valid

Figure 4. Proximity QGraph for creating patterns.

visualizations we could retrieve interesting information that could ease knowledge discovery.

C. How much data should be extracted?

How much data that should be extracted depends on the visualization task. Too much data would give much information, but the cost could be information overload, that is, a human would not easily be able to analyze the visualized network. But some times this does not have to be the case. We could extract large volumes of data and then apply smart analytic methods to create small views and visualize the views.

On the other hand, too little data would produce insufficient results, causing the analysis to be incomplete. Basically the line is drawn by the capabilities of the analysis tools used by the analyst, therefore each study case needs to be treated separately. A simple data extraction process for visualization purposes could be a question based approach [8], for example:

1) What knowledge from the information do I want to gain?
2) Can the information be visualized?
3) If the information can be visualized, which type of visualization should I use (network graphs, other charts and so on) in order for the visualization to be meaningful?
4) What sort of data instances can be visualized for the chosen type of visualization?
5) What are the limitations of this type of visualization?

D. Data Filtering

Working with social networks means dealing with enormous amounts of data. Different filtering mechanisms must be introduced in order to reduce the volume of data [9]. The proposed solution is to use three levels of filtering.

The first level of filtering is to use SQL queries which most databases support. This is a more raw type of querying comparing to the next filter levels. The second level of filtering is done with Proximity QGraph [2] to retrieve subgraph sets using different patterns. The third level which this paper focuses on is filtering using node discovery algorithms that implement so called "predicates" (predicate filtering) which narrow the search space. In short, when level one and two are completed we have a smaller graph (subgraph) which has been filtered from relational databases using SQL and "cleaned" for different patterns using Proximity QGraph.

E. Predicate Filtering

Many times when we study network data we would like to discover additional information, that is, we would like to discover new nodes that are somehow related to the existing nodes—the subgraph. There are different ways this can be done, in this paper we chose the BFS algorithm—Breadth First Search and modified it to support "predicates". Depending on the scenario and the data set, different search algorithms are optimal. We have chosen the BFS algorithm mainly for its simplicity and optimal performance on unweighted graphs.

The BFS algorithm has a time complexity of $O(|V| + |E|)$ [10], where $|V|$ is the vertex set and $|E|$ is the edge set. Algorithm 1 shows a general BFS algorithm [10].

Algorithm 1 BFS

```
1: Let $G$ be a graph
2: Q $\leftarrow$ QUEUE
3: $\forall\nu$ vertex, $\nu \in G$
4:     Paint $\nu$ white
5: Paint start vertex $\phi$ gray, $\phi \in G$
6: Q $\leftarrow$ Q + $\phi$
7: **WHILE** Q$\neq \emptyset$
8:     Q $\leftarrow$ Q - $\nu$
9:     $\forall\delta$ white neighbors of $\nu$
10:         Paint $\delta$ gray
11:         Q $\leftarrow$ Q + $\delta$
12:     Paint $\nu$ black
```

The BFS algorithm allows us to traverse through the entire network graph and find new nodes but there need to be conditions that must be fulfilled in order to expand the subgraph set or else the data set would grow enormous due to expansion. We call these conditions “predicates”, that is, a predicate—or rule—has to be fulfilled if a new undiscovered node is to be added to the subgraph set. Figure 5 shows an example of this, a subgraph and new nodes that can be discovered by some predicate.

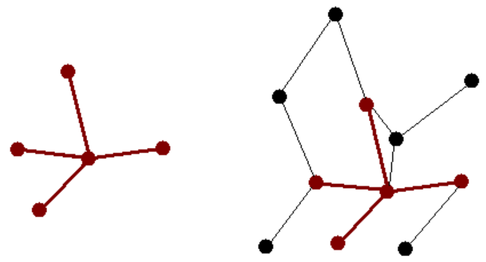


Figure 5. Shows a subgraph to the left and new nodes that can be discovered to the right.

There can be many types of predicates depending on the study, but in this paper we chose three predicates as an example. Before we mention them we explain the general outline of the new algorithm that is used to discover new nodes. This new algorithm is a modification of the BFS algorithm which was discussed earlier. Algorithm 2 shows the general outline of the new algorithm.
This new algorithm requires that we have two graphs G and S, $S \subset G$ (S has been generated on filtering-level two, by Proximity, which in turn received its input from filtering-level one, SQL against databases) and the goal is to expand S. G's vertices are painted white according to the BFS algorithm. We put all vertices of S in the queue and paint them gray since

Algorithm 2 Modified BFS

```
1: Let $G$ and $S$ be graphs, $S \subset G$
2: Let $\rho$ be a predicate
3: Q $\leftarrow$ QUEUE
4: $\forall\nu$ vertex, $\nu \in G$
5:     Paint $\nu$ white
6: $\forall\phi$ vertex, $\phi \in S$
7:     Paint $\phi$ gray
8:     Q $\leftarrow$ Q + $\phi$
9: **WHILE** Q $\neq \emptyset$
10:     Q $\leftarrow$ Q - $\nu$
11:     $\forall\delta$ neighbours of $\nu$
12:         IF $evaluate(\rho, \delta)$ AND $\nu$ is white
13:             Paint $\delta$ gray
14:             Q $\leftarrow$ Q + $\delta$
15:             $S \leftarrow S + \delta$
16:     Paint $\nu$ black
17: **RETURN** $S$
```

the S graph already exists and we want to expand it. Then we look for all nodes and try to evaluate the predicate on them. If the predicate evaluates true and the node is white, then the node is processed according to the BFS algorithm and is added to the subgraph set. This will make sure that the predicate criteria is fulfilled on each new node that we want to discover. Finally we return the expanded subgraph. For this paper three different examples of predicates were chosen for implementation:

- *Predicate 1 (“Links between the subgraph nodes”)*:
  This predicate returns true if there are edges between subgraph nodes which are not included in the default subgraph generated by Proximity.
- *Predicate 2 (“Distance from a subgraph node to a root graph node”)*:
  This predicate returns true if a node ν, where $\nu \notin S$ and $\nu \in G$ and ν is on a distance x from S. That is, if the distance from a node in S to ν is $\leq x$, $x \in \mathbb{Z}$ and $x \geq 0$. If x is 1, then all undiscovered nodes are valid if they are not present in S and are on a distance ≤ 1 from S.
- *Predicate 3 (“Links from a root graph node to the subgraph”)*:
  This predicate returns true if a node ν, where $\nu \notin S$ and $\nu \in G$ and ν has x links to nodes in S, $x \in \mathbb{Z}$ and $x \geq 0$. If x is 2, then all undiscovered nodes are valid if the number of links from ν to nodes in S are ≥ 2.

If the time complexity for a predicate is $O(1)$ then the time complexity of this new algorithm becomes $O(|V| - n + |E| - m)$, where n is the number of nodes and m the number of edges already included in the non-expanded subgraph. However in many cases time complexity should increase since the evaluate-method would require more processing.

The main strength of the modified BFS algorithm is that it follows the search standards of the BFS. This will insure us that each node that is put into the queue will be processed, therefore modification is rather trivial. The main weakness of the modified BFS algorithm is the evaluation part. This evaluation is dependent on the required input-data that needs to be collected in order for the predicates to evaluate. For a small number of predicates only little data needs to be collected, but for a large number of predicates we would perhaps have to gather a lot of data and do a lot of processing which would in turn slow down node discovery.

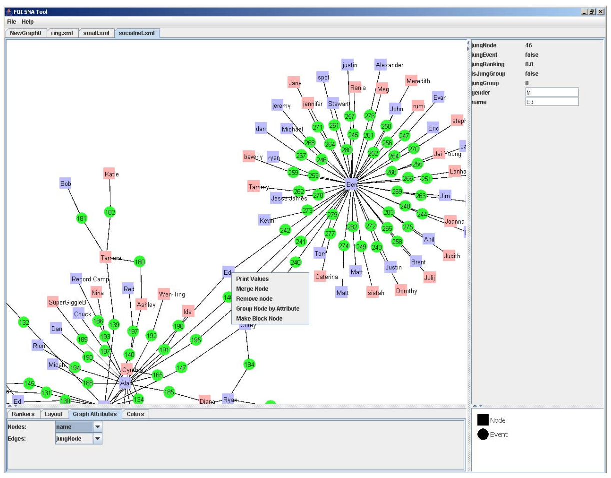
F. Network Graphs

Networks provide a great source of knowledge in understanding relations between different entities. The graph data that is finally produced should easily be exported to formats that can be interpreted by other programs. In this paper we consider two of those formats, JUNG [4] and GraphML [3].

We have chosen JUNG as an internal format for representation of graphs since it is widely used in many visualizing applications. We choose the GraphML format as an external format for graph representation for the same reasons, also this format is considered as a standard format for graphs.

V. SN Visualization Interface

In order to make use of social network analysis in the HiTS/ISAC demonstration, we also developed a visualization interface. This interface enables a user to visualize graphs and perform different network metrics in order gain further knowledge about the network. The GUI consists of tabbed panes for visualizing, analyzing and manipulating graphs. There is also a menu that is used to load and save graphs and generate random graphs. Each tab in the GUI consists of a visualization of the data graph and there are tool bars for manipulation of the graph, see Figure 6.



jungNode	46
jungEvent	false
jungRanking	0.0
isJungGroup	false
jungGroup	0
gender	M
name	Ed

Nodes: name
Edges: jungNode

Figure 6. The SN Visualization Interface.

Network metrics are selected in the *Rankers*-tab and when a metric has finished ranking the nodes, the size of the nodes are scaled. The largest node being the most important node according to the metric. Depending on the size of the graph, calculating a metric can take a very long time. There are also many other standard social network visualization features implemented in the visualization interface.

VI. Case Study

Three user demonstrations were performed by the HiTS project, starting November 2007 and ending March 2008. After each demonstration, some changes to the tools were made based on the experiences learnt at the demonstration and on feedback from the end-users present. The end-users were representatives of law-enforcement agencies from Europe who were given the task to prevent a terrorist attack from occurring.

The exercise background was as follows. There are increased activities in different independent (human) networks in multiple European countries. Some of those networks are constituted of criminals while other networks are constituted of individuals known to have connections to extreme political organizations. An amount of radioactive material is stolen in a non-EU country. A while back, a number of odd financial transactions were performed at a large bank. At the beginning, no connections between those different activities can be seen. The authorities in the different countries exchange information using the full HiTS/ISAC solution, which enables them to exchange information securely and detect suspicious events. New connections between criminal/terrorist groups that have not previously collaborated are detected, which triggers an alarm for the law enforcement agents in the countries involved.

When a shipment of medical radioactive material is lost during routine transport, this triggers a priority alarm and becomes cause for closer investigation. An analysis file is opened and one officer is given the task to investigate the alarm. During analysis of earlier automatic alarms – working "backwards" in time – a number of things that do not fit into the normal picture start to show up. Vague connections between new criminal groups, the theft of radioactive material and lost shipment of additional radioactive material, odd financial transactions, movement of members in the different criminal and extreme political networks, etc. Thanks to the information sharing system, the authorities can gather more information from different sources, including tracing the shipment of radiological material, interrogating databases, and various sensors. The information can be fused and presented to the law enforcement operators.

While efforts are being made to prevent the planned terrorist attack using the radioactive material, further analysis into the case reveals that a bank officer has been forced to help in money-laundering and to help in diverging money to the criminal network. The money has been used to finance the illegal operation. The authorities can now concentrate on refining the information, and determine the goal and target of the illegal operation. From this point on rescue services are fed all the information in order to be prepared if rescue operations become necessary.

The SNATool is used throughout this exercise in order to detect and visualize the connections between the different groups. Test databases from different authorities in different countries were provided. Data was classified as persons, bank transactions and crime records. Level one filtering was mainly specified as a time interval, that is, in which period of time data should be collected. Level two filtering specified basic relational patterns between key suspects and non-suspects. When level three filtering was applied we managed to expand the subgraph and identify nodes that had previously been filtered out. This led to great knowledge when the graph was visualized using the visualization interface. From there we could see that there were strong indications that some individuals needed to be questioned for further information, see Figure 7. Figure 7 shows intermediate connections between two different suspected persons through some non-suspects (nodes in the middle, marked by the red circle).

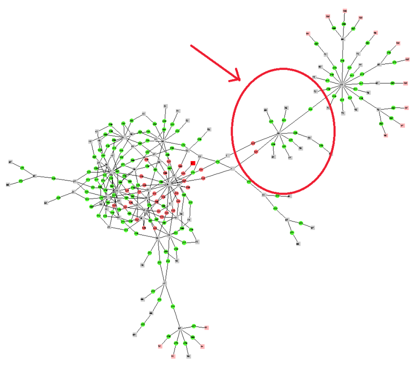


Figure 7. Visualization of the connections between two different key suspects through intermediate non-suspects (marked by red circle).

VII. Discussion

In this paper, we described the HiTS/ISAC SNATool, which was used to demonstrate how customs and police analysts in different countries could work together investigating criminal activities that were distributed across several countries. The analysis performed using the SNATool enabled the authorities to detect potential terrorist activities and therefore prevent them from occurring.

The problem of handling uncertain network information is particularly important for law enforcement applications of social network analysis, as the data used in such cases is always uncertain. Some preliminary work in this area has been performed by us [11]–[13], and we plan to continue this work an implement support for uncertain networks in the FOI SNATool.

Acknowledgments

Parts of this work was supported by the R&D programme of the Swedish Armed Forces and by the EU PASR project HiTS/ISAC (EC SEC5-PR-113700). We would like to express our thanks to our colleagues in the HiTS/ISAC project.

References

[1] P. Svenson, P. Svensson, and H. Tullberg, "Social network analysis and information fusion for anti-terrorism," in *Proceedings of the Conference on Civil and Military Readiness 2006*, 2006.

[2] Knowledge Discovery Laboratory, "Proximity," Nov. 2010, http://kdl.cs.umass.edu/proximity.

[3] GraphML Project Group, "The GraphML File Format," Nov. 2010, http://graphml.graphdrawing.org.

[4] J. O'Madadhain, D. Fisher, and T. Nelson, "Java universal network/graph framework," *Retrieved from http://jung.sourceforge.net on 15/10*, 2007.

[5] Denodo Technologies, "Denodo Virtual DataPort," Nov. 2010, http://www.denodo.com/english/virtual_dataport.html.

[6] University of Texas at Austin, "Relational Model: Normalization," Nov. 2010, http://www.utexas.edu/its/windows/database/datamodeling/rm/rm7.htm.

[7] Jiawei Han, Micheline Kamber, *Data Mining - Concepts and Techniques - Second Edition*. University of Illinois at Urbana-Champaign, Elsevier, Morgan Kaufmann Publishers, 2006.

[8] Andreas Kerren, John T. Stasko, Jean-Daniel Fekete, Chris North, *Information Visualization: Human-Centered Issues and Perspectives*. Springer, 2008.

[9] David Hand, Heikki Mannila, Padhraic Smyth, *Principles of Data Mining*. The MIT Press, Massachusetts Institute of Technology, 2001.

[10] Sara Baase, Allen Van Gelder, *Computer Algorithms - Introduction to Design & Analysises - Third Edition*. Addison-Wesley, 2000.

[11] P. Svenson, "Social network analysis of uncertain networks," in *Proceedings of the 2nd Skövde Workshop on Information Fusion Topics*, 2008.

[12] J. Dahlin, "Community detection in imperfect networks," Master's thesis, Umeå University, 2011.

[13] J. Dahlin and P. Svenson, "A method for community detection in uncertain networks," in *Proceedings of the European Intelligence and Security Informatics Conference (EISIC 2011)*, 2011.