Global Journal of
Information Technology

Volume 05, Issue 1, (2015) 41-47

www.awer-center/gjit

Visualizing social networks based on wireless sensors

Kreshnik Vukatana *, Department of Statistics and Applied Informatics, Faculty of Economy, University of Tirana, Albania.
Elira Hoxha, Department of Statistics and Applied Informatics, Faculty of Economy, University of Tirana, Albania.

Suggested Citation:
Vukatana, K. & Hoxha, E. (2015). Visualizing social networks based on wireless sensors. *Global Journal of Information Technology*. *5*(1), 41-47.

Received 22 January, 2015; revised 18 March, 2015; accepted 27 March, 2015.
Selection and peer review under responsibility of Prof. Dr. Adem Karahoca, Bahcesehir University, Turkey

Abstract

Sociometry as a quantitative method for measuring social relationships is a science also focused on social networks. The development of information technologies on both hardware and software aspects brought success stories on social networks, like facebook or twitter. The analysis over these networks is applied in different areas like management consulting, public health and crime/war fighting, just to mention a few. In this research work, we ran an experiment based on mobility of people wearing badges with wireless sensors. The goal of this experiment is to show the phenomenon of homophily, that is the tendency of individuals to associate with similar others, throw a real-time visualization.

Keywords: Real-Time Visualization; Social Networks; Wireless Sensor Networks; Gossip-Based Networking Protocols; Blockmodeling Algorithms.

*ADDRESS FOR CORRESPONDENCE: **Kreshnik Vukatana**, Department of Statistics and Applied Informatics, Faculty of Economy, University of Tirana, Albania. *E-mail address*: elirahoxha@yahoo.com

1. Introduction

One of the most potent ideas in the social sciences is the notion that individuals are embedded in thick webs of social relations and interactions. Moreno [1] was the first who introduced this notion and called it sociometry. He represented people as nodes in a graph structure and the relations between them as ties. Since then, the analysis on social networks has evolved using techniques such as matrix algebra and graph theory for formal analysis. Soon sociology researchers understood that they had new instruments in their hands to solve problems concerning human society and its actors, the people.

Nowadays, social networks have renewed interest with the introduction and success of consumer applications such as Facebook, Instagram and YouTube. But there are other practical application areas as management consulting, public health, crime/war fighting based on social network analysis that we have to mention. For example, in management consulting [2], network analysis is often applied in the context of knowledge management, where the objective is to help organizations better exploit the knowledge and capabilities distributed across their members. In public health [3], network approaches have been important both in stopping the spread of infectious diseases and in providing better health care and social support. On national security [4], terrorist groups are widely seen as networks rather than organizations, so we have a sparking research on how to disrupt the functioning of networks.

Introducing our experiment, we wish to measure the interaction between people at a social event. To this end, we consider a group of a few tens to hundred people located at the same event. We are interested in obtaining insight in the social behaviour of this group, notably the extent to which we can identify subgroups of people (often referred to by sociologists as homophily). In sociology this phenomenon is defined as the tendency of individuals to associate with similar others. It can be observed in different aspects such as age, gender, class, organizational role, and so forth. Assuming that a good measure to derive such subgroups of people can be the length of interaction between them, we can measure this interaction as the time that two individuals exchange information with each other. In the case where three or more people are interacting, we simply measure only the pairwise interactions.

In order to implement this system we have to design a robust architecture to deal with different issues. First of all is the mobility of people. For this purpose we adopted a platform based on Wireless Sensor technology that will be the lowest level of our architecture. In the case of our interaction model, it can be realized by means of wireless interactive badges worn by people that periodically broadcast messages to their neighbours, which should be at a close range. The number of messages received per time unit is then an indication for the time that two badges have been in each other's proximity. We will pay attention to clean the information from superfluous data, as will be messages delivered from two individuals that occasionally are close to each other.

The second issue is about choosing what data we are going to get from sensor nodes and how to disseminate them. We are going to explain more in detail sensor node's functionality in the experimental setup section, but basically we can say that its purpose is to create a list of its neighbours and also disseminate over the network this list and the incoming information from others. We want also the node to report the data periodically. In this way we expect to have a view of the topology of the social network over time. The dissemination is done with a gossip-based protocol called SharedState [5]. The idea behind gossip protocols is to spread information in a way similar to the spread of a virus in a biological community.

Once we have the format of reports, we want our application to be able to gather the information. So we designed a network of nodes that only listen from their neighbours and pass the data to our application. We call those node observers. The network of observers and our application devices create a separate network, because we don't want to interfere with the network behaviour of our core application.

The designed system is modular which is crucial for the analysis phase as it allows including different social analysis modules and therefore, visualizing different characteristics of the social network. For our experiment, in the analysis module we integrated an algorithm that periodically takes as input a sociomatrix, that is a matrix where the celli,j is set either to 1 if a relation exists between individuali and individualj, or 0 otherwise. In our case the relation is the interaction between two persons. After processing and analyzing the sociomatrix, the algorithm produces a list of clusters which are the groups of individuals that are having a conversation at a given period.

In visualization we stress the division of the individuals in groups based on affiliation using colors. For example the division of the individuals based on gender affiliation is boys and girls, so we color boys with blue and girls with pink. The purpose of using colors is to exalt the phenomenon of homophily when it happens. For example, in a classroom model, if the application visualizes two clusters, one with all the elements blue and the other with all the elements pink, we certainly have social division on gender affiliation.

Two visualization modes are used for the data: a matrix view and a graph view. The first one is more "technical" and visualizes the groups that are having a conversation, while the second is more "user friendly" and visualizes individuals as vertices and conversations as edges.

2. Experimental Setup

For our experiment there are used Chalcedony nodes. They are composed of ATXMega128 32MHz CPU; 8 KB SRAM; 128 KB Program Flash and a 2 KB EEPROM (non-volatile memory, to be used for storing node-specific data, e.g. node ID). They run software developed by Chess [6] and DevLab [7].

Nodes operate in rounds where each round is approximately one second. They are responsible for two tasks, that is, sending their own list of neighbours and relay others' messages. In order to perform the first task, nodes keep track of the immediate sources which are considered as possible interlocutors. We use very short range communications, i.e., up to a couple of meters, in order to have messages' exchange only among nodes in vicinity. At the end of every round the nodes build a list of their own neighbours that is broadcast along with the round number. The round number that identifies uniquely a packet sent by a given node, acts as a timer. All the nodes have the same round number (after a start-up phase) which helps in the process of gathering and merging the received data.

The second task of the nodes, i.e., the relay of the messages towards the observers, is performed through a gossip-based protocol, named SharedState. In SharedState the data to be disseminated is structured in items and each packet contains a set of items. The received items are kept in a cache structure and at every round, a packet is created containing the item of the source node and items are picked from the cache randomly.

In our experiment, an item is the basic information that is generated by the nodes. It contains the round number, the source node ID and the list of neighbours at the given round number.

The Chalcedony nodes have a small packet size, namely 32 bytes. This limitation, and also the fact that only one packet is sent per round addresses two problems: first, the list of neighbours contained in an item should be limited to an upper bound and, second, the number of items contained in a packet should be relatively small. As a result of the limited disseminated data, the visualization might not be accurate at 100% as we will see in the conclusions section.

The life cycle of a node is the following:

```
active(every round):
/* create a new packet */
myItem = createItem(roundNo,myID,neighborsList)
pkt.add(myItem)
for slot in ITEMS PER PKT - 1:
item = pickFromCache()
pkt .add(item)
upon reception of packet:
```

```
neighborsList .add(recPkt . getSource ())
for item in recPkt :
if item in Cache:
discard(item)
else
Cache.add(item)
```

For collection and processing of data, we use a post-mortem approach and a real-time approach. Intuitively, we can guess that in the first approach we can construct a global view of the social network based on the information stored locally by each node, whereas, in the second approach that is not possible. Therefore, the information must be propagated constantly to the observers, so that the application can extract partial views based on observers and merge them to generate a real-time view of the social network.

As mentioned in the introduction, the implementation of the encounter that keeps trace of interactions should differentiate between occasional collocations (for example, two people walking by each other) and a conversation. For this purpose we use an encounter which changes its behaviour whether the user has chosen to process the data in cumulative or noncumulative mode.

When using the *cumulative mode* we keep track of all interactions. For this purpose we use two matrices: the matrix of encounters and the sociomatrix. The encounter$_{i,j}$ has a *lower threshold* set to zero and an *upper threshold* that is set at run-time. At each period of time (round), every encounter$_{i,j}$ is incremented by 1 (if the event interaction$_{i,j}$ occurs) or is decremented by 1 (if no interaction occurs and the encounter is positive). If the upper threshold of any special encounter$_{i,j}$ is reached, no action is taken and the relation$_{i,j}$ on sociomatrix is set to be visualized. For example, if the user chooses cumulative and a slot of 60 rounds, this means that we are going to visualize all the interactions in which "**two individuals encounter more than 60 rounds in total**". We keep track of the encounter from the time that the statement is satisfied until the end of experiment's execution.

Through the *noncumulative mode* we are interested in visualizing the encounters during a time slot. The duration D of a time slot is expressed in rounds, and an encounter is registered as such whenever two nodes have interacted for a minimum of M rounds during a time slot. Clearly, M <= D. Counting interactions starts anew every time a new time slot starts. In this situation, the special encounter $_{i,j}$ behaviour changes a bit. It does not have a lower threshold and its upper threshold is set to M that is a value taken at run-time by user's choice. At each round we increment by 1 only the encounters$_{i,j}$ where an interaction$_{i,j}$'s event was registered. If the upper threshold of any special encounter$_{i,j}$ is reached, relation$_{i,j}$ on the sociomatrix is set to be visualized. At the end of this time slot, each encounter$_{i,j}$ is reset to zero. So, if we re-examine the example above, in the non-cumulative way we visualize all the interactions satisfying the statement "**two individuals encounter more than M rounds within a slot of 60 rounds**" and the refresh of the visualization is done every 60 rounds.

In the *post-mortem model*, we assume that each node saves its own state periodically in a log file. When the experiment is terminated all log files are processed and merged. This file is a time-based sequence of events, where each line represents a triple (round number, node id, neighbour id). The post-mortem module process goes through three steps of execution. Initially, the whole file is loaded in memory. Afterwards, the module processes only the data identified by the current round number. Finally, at the third step, it updates encounter's matrix and sociomatrix, seeks to the next round and sleeps for one second. The process is iterated through the steps two and three until the rounds are finished. During this execution time, either analysis module or visualization module can use the updated matrices for their purposes. This kind of design provides an approximate "playback" visualization of the experiment, which is very useful for debugging.

The design of the *real-time module* is more complicated than the post-mortem one. It has to gather information in real-time from a couple to some dozens of observers' applications. This information has to be synchronized, scheduled in rounds and also cleaned from the duplicates. Most importantly, the module must implement all the functionalities mentioned in the post-mortem module that are, gathering cumulative and noncumulative data. Also, at any time, either the analysis or the visualization module should access the current state of the information. To this end we implemented a multi-threaded process. The main thread implements a hash table that has round numbers as keys and a sociomatrix as entry. For each observer, it launches a thread that integrates serial port functionalities. These functionalities are implemented via RXTX Api [8] based on the Java Communications Api [9]. Initially, some parameters are set to start up the communication with the observer-application. After the connection is set up, each of the launched threads continues in listen mode. When the "serial port receiving data" event occurs, the received report message is read from the buffer and the information is used to update the appropriate matrix into the hash table, based on the round number. For this purpose we pay attention in designing a concurrent-safe communication between the main and these secondary threads. The main thread, initially, ensures that the hash satisfies the invariant "filled in with at least thirty matrices". This invariant is intentional due to the latency time that the messages disseminated over network spend to reach the observers. Note that the functionalities are the same for the post-mortem module and the main thread of the real-time module. The only difference is the stored information, i.e., using a text file (post-mortem approach) or a hash table (real-time approach).

In the analysis stage is used the blockmodeling technique, which is a matrix method for sorting network actors into jointly occupied, structurally equivalent positions. Blockmodel methods were initially developed by H.White and his associates [10]. They divide a sociomatrix of n actors that describes one or more relational networks, into two or more discrete subgroups or positions, called block. The term block refers to a square submatrix of structurally equivalent actors that have very similar, if not identical, relations with the actors occupying the same block. Blockmodeling is a data reduction technique that systematically searches for relational patterns in network data by regrouping actors. The output are permuted density or binary matrices (in our case) displaying the pattern of ties within and between the blocks for each type of relation. Our method is based on CONCOR (convergence of iterated correlations) algorithm [11]. It uses the techniques of blockmodeling where at each iteration to each couple in the matrix is applied the Pearson correlation coefficient. The result is that each pair with the same pattern of membership shows a correlation equal to 1, while a couple that have a completely different pattern will have a correlation of -1. The correlations matrix is then permutated and the result are two composed blocks or classes of membership.

To test the visualization for both matrix and graph view, we borrowed the classmates' example data based on material from Sherman [12]. This data is collected from children of a 4th grade classroom. Each child is asked to list the three persons he or she likes the most (known as a positive nomination) or the least (i.e., negative nominations). An entry (i, j) on matrix, marked with "+" indicates that child i liked child j, whereas a "-" indicates that child i disliked child j. In our analysis, we consider only the "positive" interactions. Also, in this example the "time" factor is not considered. The information is adapted in a post-mortem format input; where each line has the round set to 1 and a relation is "child_x likes to be friend with child_y". Once the data is processed by the post-mortem module and the sociomatrix is created, we can have two kinds of visualizations as shown in the next images, on Fig. 1. The user can set a configuration file, where he can classify groups of actor by using different colours. In this example we classify the children, in boys (blue) and girls (pink). The left image shows the result of the sociomatrix after it was analyzed by the CONCOR algorithm. The colours settings in the configure file are automatically reconfigured to represent relations. So, the $\text{cell}_{i,j}$ in blue means boy_i likes to be friend with boy_j, the one in pink means girl_i likes to be friend with girl_j and in green the rest of possible relations. The right image shows a graph, where the vertices are the children and the ties are the relations "likes to be friend". For drawing the graphs we used the jung library [13]. It uses a force-based algorithm, called spring embedding. This method assigns forces as if the edges are springs (Hooke's law) and the nodes are electrically charged particles (Coulomb's law). The entire graph is then simulated as if it were a physical system. The forces are applied to the nodes, pulling them closer together or pushing them further apart. This is repeated iteratively until the system comes to an equilibrium state; i.e., their relative positions do not change anymore from one iteration to the next.

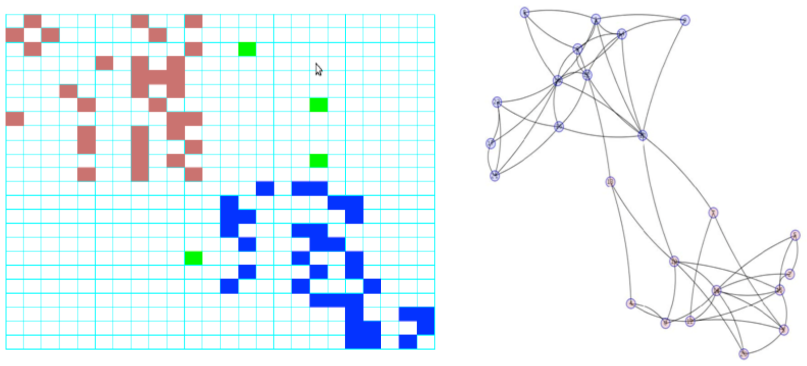

Fig. 1. Matrix and Graph View

This is another technique for analyzing the data, where we can notice immediately the division between boys and girls. As Sherman explained, it is an example of homophily based on gender, which often occurs in upper elementary school classrooms.

3. Conclusions

In this paper we show the design and implementation of a system that analyzes and visualizes mobile social networks in real time. All the process goes through four phases: gathering of social information, processing of data, analysis and visualization of a social network. For this purpose we adopt a Wireless Sensor Network which gets real time neighbour list feedback, by the participants of the network. The collected data are analyzed by means of social analysis techniques, such as CONCOR algorithm, and the results are displayed in real time.

In order to see how accurate our real time visual analysis is, we ran an experiment involving 20 nodes, where the information proceeds through the whole process, from gathering to visualization. During the execution of the experiment data was also logged for a later off-line processing (post-mortem approach explained above), which provides a full insight of each node's state in the network. By comparing the real time with the post-mortem approach (Fig. 2) we assess the accuracy of the visual analysis. The results show that this accuracy does not go below 90-95%. This is an important outcome because as far as we keep the accuracy over these thresholds we have good possibilities to maintain the characteristics of the social network untouched.

As future work, we want to extend the results for a large-scale network and a more realistic mobility. For this purpose, we want to investigate deeper through simulations and experiments on how to improve factors that directly affect the visual analysis of mobile social networks, such as dissemination of data, number and position of the observers. A second goal is to introduce several analysis algorithms to our modular system in order to show different characteristics of a given social model. For instance, we can analyze the betweenness centrality [14] (the percentage of all the shortest paths that a node is involved) and show that the individual$_i$ is more influent that the individual$_j$ or analyze the degree dentrality (the number of the connections a node has) and show which individual is the most active in the network at a given time, just to name a few.

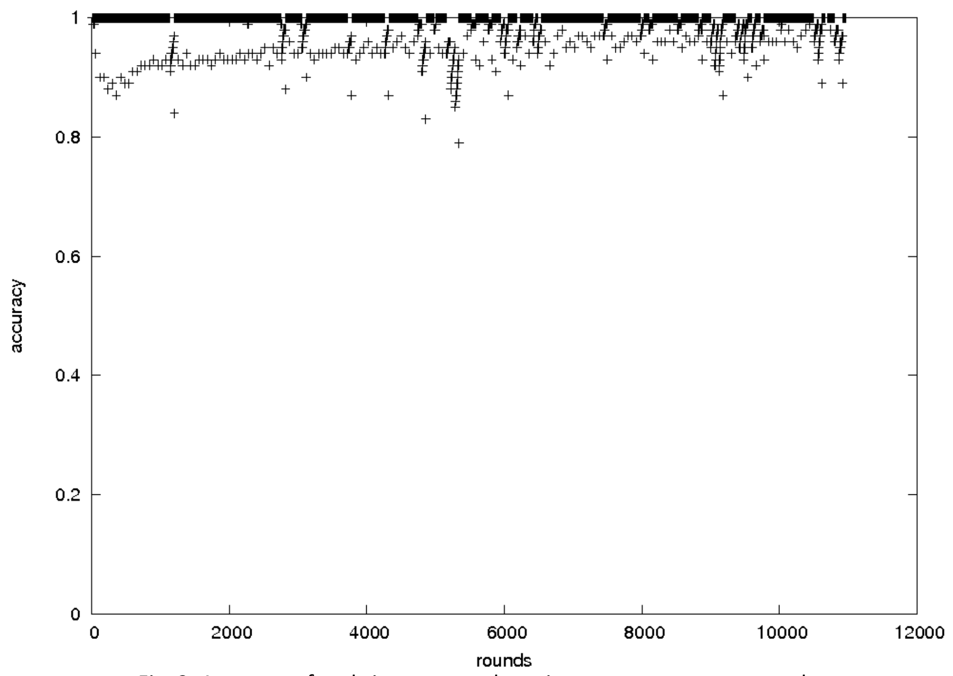

Fig. 2. Accuracy of real-time approach against post-mortem approach

References

[1] Moreno, J. L., & Jennings, H. H. (1934). Washington, DC: Nervous an Mental Disease Publishing Company, 1934.

[2] R0ross, R., & Parker, A. (2004). The Hidden Power of Social Networks, Harvard Business School Press, Boston, MA.

[3] Levy, J.A., & Pescosolido, B.A. (2002). Social Networks and Health, Elsevier.

[4] Sageman, M. (2004). Understanding Terror Networks, University of Pennsylvania Press, Philadelphia.

[5] Gavidia, D. (2009). Epidemic-Style Information Dissemination in Large-Scale Wireless Networks, Ph.D. Thesis.

[6] CHESS, doi: http://www.chess.nl/.

[7] DevLab Consortium, doi: http://www.devlab.nl/.

[8] RXTX, an serial I/O driver in java for Windows, doi: http://rxtx.qbang.org/wiki/index.php/FAQ .

[9] COMM, the official serial I/O driver in java for Linux, Solaris and Mac, doi: http://java.sun.com/products/javacomm/.

[10] White, H. C., Boorman, S. A., & Breiger, R. L. (1976). Social structure from multiple networks. I. Blockmodels of roles and positions. *American journal of sociology*, 730-780.

[11] Breiger, R. L., Boorman, S. A., & Arabie, P. (1974). *An Algorithm for Blocking Relational Data, with Applications to Social Network Analysis and Comparison with Multidimensional Scaling*. Technical Report 244, Institute for Mathematical Studies in the Social Sciences, Stanford University.

[12] Sherman, L. (2000). Sociometry in the Classroom: How to do it. doi: http://www.users.muohio.edu/shermalw/

[13] Jung, doi: http://jung.sourceforge.net.

[14] Brandes, U. (2008). On variants of shortest-path betweenness centrality and their generic computation. Social Networks30: 136–145.
doi: http://www.sciencedirect.com/science/article/pii/S0378873307000731