Visualization for Streaming Networks

Rui Sarmento[1,2], Mário Cordeiro[1], and João Gama[1,2]

[1] LIAAD-INESC TEC, University of Porto
[2] Faculty of Economics, University Porto

Abstract. Sampling from Large Scale Social Networks is a hot topic in recent research. In telecommunications services, there are many networks with millions of nodes and billions of edges. They are complex and difficult to analyze. Sampling, together with vizualization techniques, are required for exploratory data analysis and event detection. Until now, to visualize and analyze the massive network data we would rely on aggregation of communities, k-Core decompositions and matrix feature representations, among others. In social network visualization and analysis the goal is to get more information from the data with the least alienation possible from the actors of the network. Our contribution is to treat the data like a continuous data stream and represent it by sampling the full network. We also propose group visualization and analysis of influential actors in the network by using a Top-K representation of the network data stream.

Keywords: Sampling Large Scale Social Networks; Data Streams; Telecommunication Networks.

1 Motivation

Large network visualization is known to be a hard problem to solve with typical hardware or software sets. It is vulgar to see software and hardware suffer to output networks with more than a few thousands nodes and edges. The software and the observer himself seems to be the main constraints in visualization tasks of large networks. It must be said that even if the software is capable of outputting a network of millions of nodes on the screen it is a very hard task for the observer to gather valuable information from the visual outcome. This document main contribution is our proposal to treat the data as a stream of networked data with Landmark, Sliding Windows and Top-K algorithm implementations to help the observer visualize the network and be able to acquire knowledge from the output. The main goal is to sample the stream by highlighting the Top-K nodes thus providing clear insight on the most active nodes in the network. This document presents a Telecommunications network data case study for application of our methods. The network size is of several millions of nodes and edges. The results were obtained with a vulgar commodity machine.

2 Related Work

2.1 Visualization

The definition of large scale networks regarding number of nodes or edges varies a lot. While researching this field it is usual to see researchers considering a large scale network ranging from something like some dozens of thousands of nodes to millions of nodes and billions of edges. The main goal of any graph visualization technique is to be visually understandable. It is also desirable that the information it outputs to the viewer is sufficiently clear and objective to convey knowledge acquiring. Some studies have been performed to study two types of graph representations, node-link and matrix graph representations like in [13]. Those studies concluded visualization comprehensibility is highly related with the network size (number of nodes) and density (average number of edges per node). Node-link representation is mentioned to have low performance with dense networks and it requires aggregation methods reducing density to increase visual comprehensibility of output. Matrix representation is usually combined with hierarchical aggregation [1]. The hierarchical clustering implies the grouping of nodes but not the ordering of them. The main goal of this representation type is to have a fast clustering algorithm and meaningful clusters of the network. Matrix representation methods can also rely on the reordering of rows and columns of the matrix representation instead of just clustering the nodes and there are several examples of these implementations in [11]. This type of ordered matrix representation might enhance the structure visualization and give more information to the viewer because the data is not only clustered. On the other side this solution is difficult to use with networks of millions of nodes due to the computations needed to reordering of matrix. More recently some innovations were introduced for fast reordering mechanism, data aggregations and GPU-accelerated rendering to deliver solutions with higher scalability [6]. Other solutions rely on controlling the visual density of graph view and limiting the clusterization overlap probability to low levels in [17]. Moreover a new probability based network metric were introduced in [9] to identify potentially interesting or anomalous patterns in the networks. In the next sections we will write our approach by inspecting Top-K actors in the network and also to the case study in hands.

2.2 Top-K Networks

There is some effort by the scientific community to achieve efficient ways of data streams summarization. Regarding streaming networks the exact solution implies the knowledge of all the nodes and edges frequency, therefore this exact solution might be impossible to achieve in large scale networks. The problem of finding the most frequent items in a data stream S of size N is, roughly put, the problem to find the elements e_i whose relative frequency f_i is higher than a user specified support ϕN, with $0 \leq \phi \leq 1$ [7]. Given the space requirements that exact algorithms addressing this problem would need [3], several algorithms were

already proposed to find the top-k frequent elements. Generically, there are two different types of approaches: *Counter-based* and *Sketch-based* [16]. *Counter-based* techniques rely on the updates of individual counters for each element in a specific data subset. If there is no counter for some particular element and therefore it is not being monitored some algorithm action is taken. *Counter-based* techniques are said to have fast per-element processing and provable error bounds [16]. *Sketch-based* techniques are different from *Counter-based* topologies because they do not monitor a subset of elements but rely on a frequency estimation for all elements by using bit-maps of counters [16]. Therefore they are more expensive in terms of processing than the *Counter-based* implementations. Moreover *Sketch-based* implementations do not guarantee frequency estimation/approximation errors. Simple *counter-based* algorithms such as *Sticky Sampling* and *Lossy Counting* were proposed in [15], which process the stream in reduced size. Yet, they suffer from keeping a lot of irrelevant counters. *Frequent* [5] keeps only k counters for monitoring k elements, incrementing each element counter when it is observed, and decrementing all counters when a unmonitored element is observed. Zeroed-counted elements are replaced by new unmonitored element. This strategy is similar to the one applied by *Space-Saving* [16], which gives guarantees for the top-m most frequent elements. *Sketch-based* algorithms usually focus on families of hash functions which project the counters into a new space, keeping frequency estimators for all elements. The guarantees are less strict but all elements are monitored. The *CountSketch* algorithm [3] solves the problem with a given success probability, estimating the frequency of the element by finding the median of its representative counters, which implies sorting the counters. Also, *GroupTest* method [4] employs expensive probabilistic calculations to keep the majority elements within a given probability of error. Although generally accurate, its space requirements are large and no information is given about frequencies or ranking.

We adopted the *Space-Saving* algorithm described in [16] throughout our Top-K implementation because it is a memory efficient implementation and guarantees most active users which is our goal.

3 Case Study

The characteristics of the large scale data will be presented in this section. Telecommunication networks generate large amount of continuous data from phone users and network equipment. In this case study we used CDR (call detail records) log files retrieved from equipment distributed geographically. The network data has, on average, 10 million calls (edges in the social network) per day. This represents an average of 6 million of unique users (nodes in the network) per day. Each edge represents a call between A and B phone numbers. We had available 135 days of anonymized data. For each edge/call there is a timestamp information with the date and time with resolution to the second representing the beginning of the call. The volume of data speed ranges from 10 up to 280 calls per second usually around mid-night and mid-day time, respectively.

3.1 Data Description

In a first analysis, we look for the distribution of calls. We started by counting the number of calls from A→B in one day of the data. This operation was done with a MySql database query by selecting pairs of caller and receiver numbers and counting the occurrences of those pairs in the database. The obtained results imply a compressed representation of the original network i.e. without repeated edges. After the previous operation we studied the distribution of the aggregated data and concluded that this representation has a Power Law distribution [2] as can be seen in Fig. 1(left). Therefore we can expect a high number of single calls between some A→B numbers and a low number of many calls between A→B numbers per day.

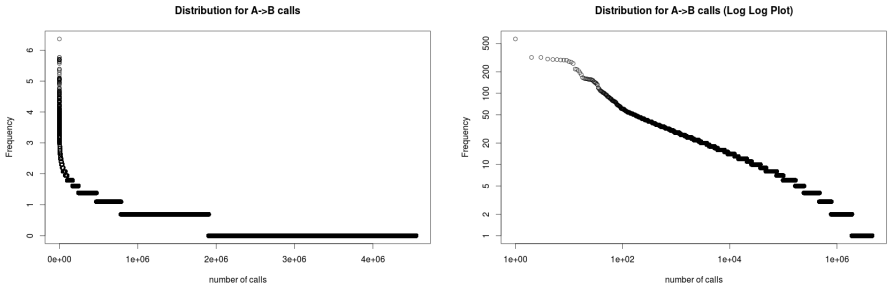

Fig. 1. A→B Calls Distribution (left) and respective Log-Log Plot (right)

We then studied the Log-Log representation of the distribution per day of aggregated data as seen in Fig. 1(right). With this representation we can visualize an approximation to a monomial.

For the received and caller calls distributions of the original data with this same representation method we could also obtain a monomial and we could also conclude both distributions follow a Power Law distribution by using the method to test the Power Law hypothesis in [8].

The previous Figures provides us a visualization of an important data characteristic which is the great amount of isolated calls between some pairs of numbers and a low amount of repeated calls between them. We conclude it is logical to disregard these isolated calls to improve visualization and analysis quality as we will later see in this document with the Top-K visualization method.

3.2 Landmark Windows

Landmark Window Algorithm 1 provides the representation of all the events that occur in the network starting at a specific time stamp, for example 01h48m09s of 1st January 2012.

Algorithm 1 Landmark algorithm Pseudo-Code

Input: *start*, *wsize*, *tinc* ▷ start timestamp, window size and time increment
Output: *edges*
1: $R \leftarrow \{\}$ ▷ data rows
2: $E \leftarrow \{\}$ ▷ edges currently in the graph
3: $R \leftarrow$ getRowsFromDB (*start*)
4: $new\_time \leftarrow start$
5: **while** $(R <> 0)$ **do**
6: **for all** $edge \in R$ **do**
7: addEdgeToGraph(*edge*)
8: $E \leftarrow E \bigcup \{edge\}$
9: **end for**
10: $new\_time \leftarrow new\_time + tinc$
11: $R \leftarrow$ getRowsFromDB (new_time)
12: **end while**
13: $edges \leftarrow E$

This type of representation is not very useful because it implies a crescent number of events outputted on the screen and comprehensibility lowers as this number reaches and surpasses some thousands of events. This landmark implementation is however useful in other contexts like for example if user network is relatively small and the user wants to check all events in the network. If the user wants to follow the evolution of a large network events the implementation described in the next subsection is better.

3.3 Sliding Windows

With the need to treat the large data stream we did a dynamic sample representation of the data designated by sliding window. This sliding window is defined as a data structure with fixed number of registered events. Each event is a call between any particular pair of phone numbers. As these events have time stamps the time period between the first call and the last call in the window is easily computed. The input parameters of this algorithm are start date and time and also the maximum number of events/calls the sliding window can have. The SNA model used in this implementation is full network directed since any nodes in the network are outputted to the screen and for the particular window of events[10].

Visually, the example result can be seen in Fig. 2. In this representation several nodes appear bigger and that represents more received/made calls by these particular numbers. This is the representation of a window with 1000 events/calls for a period of time beginning at 00h01m52s and until 00h02m40s. The reader can check the evolution of the network and visually and immediately conclude that the anonymized brown, dark blue and light blue are the nodes with more influence in this window of time.

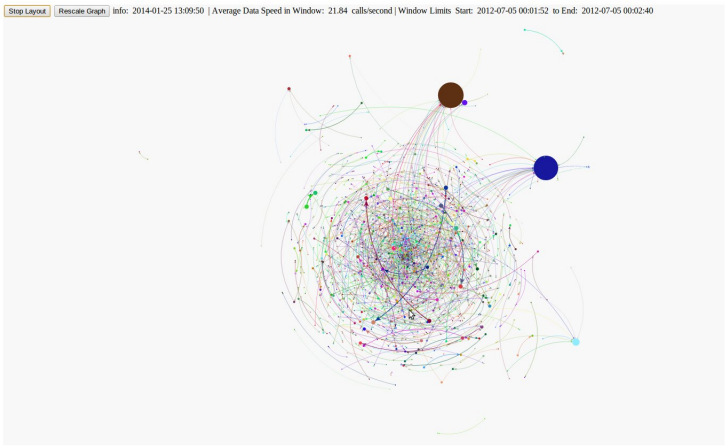

Fig. 2. Visualization of the phone calls using a Sliding Window approach

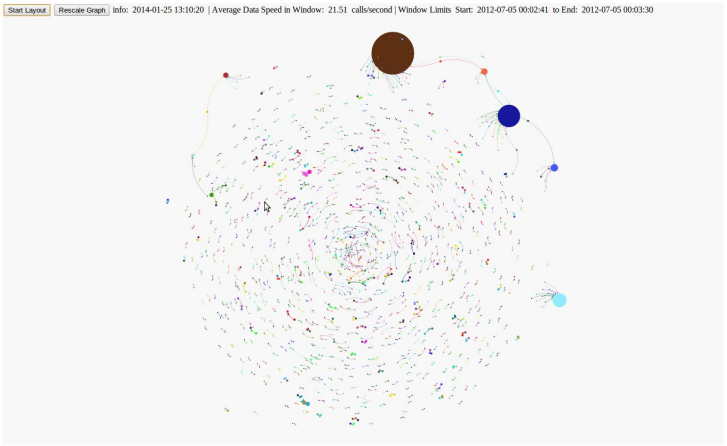

Fig. 3. Visualization of the phone calls using a Sliding Window approach

One can also see the connection between the dark blue node and the brown node being established in the represented window. Fig. 2 also displays the average data speed in the window i.e. the speed was approximately 22 calls per second. This average data speed is calculated regarding number of events/calls in the window of events and the time period between the first event time stamp and the last event time stamp represented in the visualized window. Throughout

other experimental conditions for example with windows around mid-day we experienced data speed increases with more calls per second. Considering these data speed changes and after several experiments with window size parameter we concluded that it should not be smaller than approximately 100 events and also not bigger than approximately 1000 events. With the minimum data speed conditions, 100 events represents a window period of around 10 seconds of events. With the maximum data speed and a window of 1000 events it represents around 5 seconds of calls data. Less than 100 events with this data represents changes in the window that are too fast to be visually comprehensible and more than 1000 events represents too much events decreasing visual comprehension of the screen output.

Fig. 3 represents the next window between 00h02m41s and 00h03m30s. From Fig. 2 we can visually check the evolution of the network and immediately conclude that the anonymized brown, dark blue and light blue are the nodes with more influence in this window of 1000 events.

Algorithm 2 Sliding Window algorithm Pseudo-Code

Input: *start*, *wsize*, *tinc* ▷ start timestamp, window size and time increment
Output: *edges*
1: $R \leftarrow \{\}$ ▷ data rows
2: $E \leftarrow \{\}$ ▷ edges currently in the graph
3: $V \leftarrow \{\}$ ▷ buffer to manage removal of old edges
4: $R \leftarrow$ getRowsFromDB ($start$)
5: $new\_time \leftarrow start$
6: $p \leftarrow \{\}$
7: **while** ($R <> 0$) **do**
8: **for all** $edge \in R$ **do**
9: addEdgeToGraph($edge$)
10: $E \leftarrow E \bigcup \{edge\}$
11: $k \leftarrow 1 + (p \bmod wsize)$
12: $old\_edge \leftarrow V[k]$
13: removeEdgeFromGraph(old_edge)
14: $E \leftarrow E \setminus \{old\_edge\}$
15: $V[k] \leftarrow edge$
16: $p \leftarrow p + 1$
17: **end for**
18: $new\_time \leftarrow new\_time + tinc$
19: $R \leftarrow$ getRowsFromDB (new_time)
20: **end while**
21: $edges \leftarrow E$

3.4 Top-K Networks

Algorithm 3 represents our version of the Top-K space saving algorithm. The space-saving algorithm is one of the most efficient one-pass algorithms to find

the most frequently occurring items in the stream. In our case-study, we are interested in continuously maintain the top-k most active phone users. Activity can be defined as making a call, receiving a call, or communications pairs of users. In this section, we restrict the analysis to the most active calling users.

Algorithm 3 Top-K algorithm Pseudo-Code for made calls inspection

Input: *start*, *k_param*, *tinc* ▷ start timestamp, k parameter and time increment
Output: *edges*

1: $R \leftarrow \{\}$ ▷ data rows
2: $E \leftarrow \{\}$ ▷ edges currently in the graph
3: $R \leftarrow$ getRowsFromDB (*start*)
4: $new\_time \leftarrow start$
5: **while** ($R <> 0$) **do**
6: **for all** $edge \in R$ **do**
7: $before \leftarrow$ getTopKNodes()
8: updateTopNodesList(*edge*) ▷ update node list counters
9: $after \leftarrow$ getTopKNodes()
10: $maintained \leftarrow before \bigcap after$
11: $removed \leftarrow before \setminus maintained$
12: **for all** $node \in after$ **do** ▷ add top-k edges
13: **if** $node \subset edge$ **then**
14: addEdgeToGraph(*edge*)
15: $E \leftarrow E \bigcup \{edge\}$
16: **end if**
17: **end for**
18: **for all** $node \in removed$ **do** ▷ remove non top-k nodes and edges
19: removeNodeFromGraph(*node*)
20: **for all** $edge \in node$ **do**
21: $E \leftarrow E \setminus \{edge\}$
22: **end for**
23: **end for**
24: **end for**
25: $new\_time \leftarrow new\_time + tinc$
26: $R \leftarrow$ getRowsFromDB (*new_time*)
27: **end while**
28: $edges \leftarrow E$

For this representation the input parameters of this algorithm are start date and time and also the maximum number of nodes to be represented (the K parameter). From the inputted start date and time the Top-K implementation would visually output the evolving network of the Top-K actors. The user has also the option to choose to inspect the Top-K network of the numbers that initiate calls, the numbers that receive calls and finally the Top-K representation of the A→B calls. From now on and in this document we define the caller number as the main actor for our Top-K model and we will only provide results and experiments for this situation. Therefore the weight of each actor is related to

the number of made calls by each actor i.e. the number of edges representing initiated calls by the focused network phone number.

Fig. 4 represents the output of the Top-100 nodes or phone numbers with more made calls until 00h44m33s. The program started running at midnight of the first day of July 2012 and shown the 100 most active phone numbers in that period.

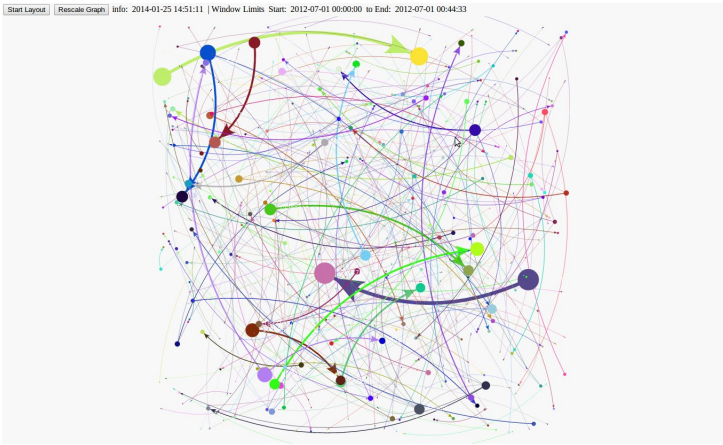

Fig. 4. Top-100 numbers with more made calls and their connections without running the layout algorithm

Fig. 5 represents the output of the Top-100 anonymized nodes or phone numbers with higher number of made calls but now with the layout algorithm running. The output only considers algorithm results until 01h09m45s.

ForceAtlas2 was the chosen layout algorithm. This layout algorithm has some good characteristics [12], [14]. These special ForceAtlas2 characteristics are:

- Continuous layout algorithm, that allows the manipulation of the graph while it is being rendered. It is based on the linear-linear model where the attraction and repulsion are proportional to distance between nodes. The convergence of the graph is considered to be very efficient once that features an unique adaptive convergence speed.
- Proposes summarized settings, focused on what impacts the shape of the graph (scaling, gravity...). It is suitable for large graph layout because it features a Barnes Hut optimization (performance drops less with big graphs).

This layout algorithm although being reported to be slow with more than dozens of thousands nodes is perfectly capable of outputting the layout of the used windows sizes throughout all our experiences. As explained before it is

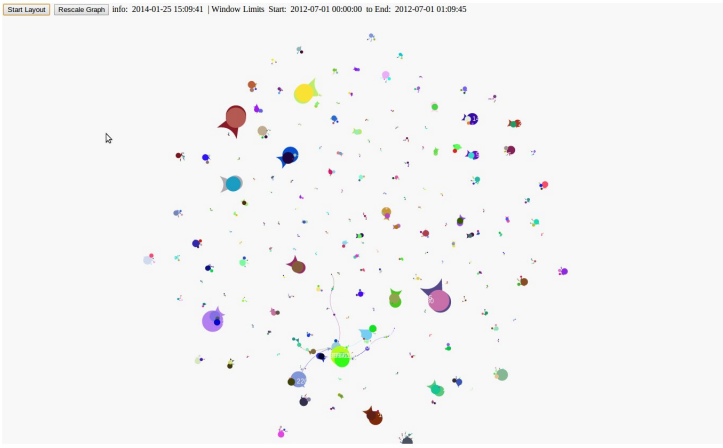

Fig. 5. Top-100 numbers with more calls and their connections with layout algorithm running

expected with our data that the windows size do not get much bigger than 1000 events for the Sliding Windows representations and also K parameter lower than 100-200 nodes for the Top-K representation. Numbers much higher than these start to turn the output less understandable and the layout algorithm also becomes slow.

4 Conclusions

This document tries to expose a new type of treatment for Large Scale Telecommunications Networks visualization. With the use of data time stamps we approach the data with a streaming point of view and try to visualize samples of data in a way that is both understandable to the viewer and also allows him/her to gather knowledge from the visual output.

Landmark Windows experiments proved to suffer from the problems we wish to avoid i.e. low visual comprehensibility of the network and even memory issues with the software. This happens when the number of nodes and edges exceeds dozens of thousands of nodes. With our data this number of nodes represented in the screen typically corresponds to a time period of just a few minutes. Sliding windows were used as a way to continuously check for the full network events. Sliding Windows allow us to continuously inspect temporal evolution of the networks. The Top-K implementation is a very good approach to our data presenting a Power Law distribution for calls. This allows us to focus on the influential individuals and discard isolated calls which are the majority of calls in our data. Finally we conclude that our method for evolving networks visualization, specially with Sliding Windows or the Top-K model is a light method to

visualize massive networks. The use of a vulgar commodity machine made possible to simulate a data stream and achieve visualization results very proximate to the node-link level. This means we tried avoiding other types of representations previously mentioned in this document's related work. These other types however use hierarchical aggregation of features, for example node communities that we could also add i.e. simultaneously make community detection of the network and output the network with added information to the node-level. This is true for communities, centrality measures like betweeness or closeness centrality could also be added to the node information complementing its visualization on the screen.

The goal and future work is to use this kind of real time data streaming leveraging telecommunication systems and being able to visualise the evolving network in real time. This can lead to applicable uses for fraud detection by inspection of Top-K users in the network or commercial purposes by detecting central actors in the network for example. If more information was available in the data it could also be added to the visual output. A simple mouse pointer over the node and the additional node information could be interactively checked by the system user. Future work also includes testing the models with time decay factors to be possible to use for example the Landmark model, giving more importance to recent data and disregarding old data.

Acknowledgments

This work was supported by Sibila and Smartgrids research projects (NORTE-07-0124-FEDER-000056/59), financed by North Portugal Regional Operational Programme (ON.2 O Novo Norte), under the National Strategic Reference Framework (NSRF), through the Development Fund (ERDF), and by national funds, through the Portuguese funding agency, Fundação para a Ciência e a Tecnologia (FCT), and by European Commission through the project MAESTRA (Grant number ICT-2013-612944). The authors also acknowledge the financial support given by the project number 18450 through the "SI I&DT Individual" program by QREN and delivered to WeDo Business Assurance. Finally the authors acknowledge the reviewers for their constructive reviews on this document.

References

1. James Abello and Frank van Ham. Matrix zoom: A visual interface to semi-external graphs. In *Proceedings of the IEEE Symposium on Information Visualization*, INFOVIS '04, pages 183–190, Washington, DC, USA, 2004. IEEE Computer Society.
2. A. L. Barabasi. The origin of bursts and heavy tails in human dynamics. *Nature*, (435):207–211, 2005.
3. Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. In *Proceedings of the 29th International Colloquium on Automata, Languages and Programming*, ICALP '02, pages 693–703, London, UK, UK, 2002. Springer-Verlag.

4. Graham Cormode and S. Muthukrishnan. What's hot and what's not: tracking most frequent items dynamically. 2003.
5. Erik D Demaine, Alejandro López-Ortiz, and J Ian Munro. Frequency estimation of internet packet streams with limited space. In *Algorithms-ESA 2002*, pages 348–360. Springer, 2002.
6. Niklas Elmqvist, Thanh-Nghi Do, Howard Goodell, Nathalie Henry, and Jean-Daniel Fekete. ZAME: Interactive Large-Scale Graph Visualization. In IEEE Press, editor, *IEEE Pacific Visualization Symposium 2008*, pages 215–222, Kyoto, Japon, 2008. IEEE.
7. Joao Gama. *Knowledge Discovery from Data Streams*. Chapman & Hall/CRC, 1st edition, 2010.
8. Colin S Gillespie. *Fitting heavy tailed distributions: the poweRlaw package*, 2014. R package version 0.20.5.
9. Frank Ham, Hans-Jörg Schulz, and Joan M. Dimicco. Honeycomb: Visual analysis of large scale social networks. In *Proceedings of the 12th IFIP TC 13 International Conference on Human-Computer Interaction: Part II*, INTERACT '09, pages 429–442, Berlin, Heidelberg, 2009. Springer-Verlag.
10. Robert A. Hanneman and Mark Riddle. *Introduction to Social Network Methods*. University of California, Riverside, Riverside, CA, USA, 2005.
11. Nathalie Henry and Jean daniel Fekete. Matrixexplorer: a dual-representation system to explore social networks. *IEEE Transactions on Visualization and Computer Graphics*, 12:677–684, 2006.
12. M. Jacomy. Forceatlas2, the new version of our home-brew layout., 2013. [Online; accessed 21-Dec-2013].
13. Bongshin Lee, Catherine Plaisant, Cynthia Sims Parr, Jean-Daniel Fekete, and Nathalie Henry. Task taxonomy for graph visualization. In *Proceedings of the 2006 AVI Workshop on BEyond Time and Errors: Novel Evaluation Methods for Information Visualization*, BELIV '06, pages 1–5, New York, NY, USA, 2006. ACM.
14. T. Venturini M. Jacomy, S. Heymann and M. Bastian. Forceatlas2, a graph layout algorithm for handy network visualization, 2011. [Online; accessed 29-Dec-2013].
15. G. S. Manku and R. Motwani. Approximate frequency counts over data streams. In *Proceedings of the 28th International Conference on Very Large Data Bases*, 2002.
16. Ahmed Metwally, Divyakant Agrawal, and Amr El Abbadi. Efficient computation of frequent and top-k elements in data streams. In *Proceedings of the 10th International Conference on Database Theory*, ICDT'05, pages 398–412, Berlin, Heidelberg, 2005. Springer-Verlag.
17. Lei Shi, Nan Cao, Shixia Liu, Weihong Qian, Li Tan, Guodong Wang, Jimeng Sun, and Ching-Yung Lin. Himap: Adaptive visualization of large-scale online social networks. In Peter Eades, Thomas Ertl, and Han-Wei Shen, editors, *PacificVis*, pages 41–48. IEEE Computer Society, 2009.