Fast visualization of relevant portions of large dynamic networks

Przemyslaw A. Grabowicz
Institute for Cross-Disciplinary Physics and Complex Systems
University of Balearic Islands

Luca Maria Aiello
Yahoo! Research Barcelona

Filippo Menczer
Center for Complex Networks and Systems Research
Indiana University

ABSTRACT

Detecting and visualizing what are the most relevant changes in an evolving network is still an open challenge in several domains. We develop a fast algorithm that selects subsets of nodes and edges that best represent an evolving graph and visualize it by either creating a movie, or by streaming it to an interactive network visualization tool. Our code, that is already deployed in the movie generation tool of the `truthy.indiana.edu` system, is limited in memory and processor time usage.

1. INTRODUCTION

Recently, the availability of live data streams from online social media motivated the development of interfaces to process and visualize evolving graphs. Dynamic visualization is supported by tools like GraphAEL [4], GleamViz [2], Gephi [1], and GraphStream [3]. In particular, Gephi supports graph streaming with a dedicated API based on JSON events and enables the association of timestamps to each graph component.

Nevertheless, not much work has been done so far about developing information selection techniques for dynamic visualization of large graphs. In fact, for large networks in which the rate of structural changes in time could be very high, the task of determining the nodes and edges that can constitute the salient structural properties of the network at a certain time is crucial to produce meaningful visualizations of the graph evolution.

We contribute to fill this gap by presenting a new tool for graph visualization that:

- processes a chronological sequence of interactions between the graph nodes;
- dynamically selects the most relevant parts of the network to visualize, based on a scoring function that weights nodes and edges, removing no longer relevant portion of the networks and emphasizing old nodes and links that show fresh activity;
- produces a file representing the network evolution or, alternatively, connects to the Gephi graph visualization tool interface for live visualization of the evolving graph;
- is fast enough to be applied to large live data streams and visualize their representation in form of a network.

The submission package contains the source code of the module with the related documentation, four dynamic graphs datasets and the respective movies produced with our tool for these datasets.

2. ALGORITHM

First, we introduce an algorithm that takes in input a chronological stream of interactions between nodes (i.e. network edges) and converts it into a set of graph updates that account only for the most relevant part of the network. Then we convert the sequence of the updates into image frames that are combined into a movie depicting the network evolution, or we feed the updates directly to Gephi Streaming API to produce an interactive visualization of the evolving network.

Input data format

The data required as input is an ordered chronological sequence of interactions between nodes. The interactions can be either pairwise or cliques of interacting nodes. For instance, the following input:

$$\langle t_1, n_1, n_2 \rangle$$
$$\langle t_2, n_1, n_3, n_4 \rangle$$

represents the occurrence of an interaction between nodes n_1 and n_2 at epoch time t_1 and an interaction between n_1, n_3, and n_4 at epoch time t_2. Entries with more than two nodes are interpreted as interactions happening between each pair of members of the clique. A repetition of the same entries at the same time encodes the intensity of interaction.

Differential network updates

The input data is processed by an algorithm that assigns scores to nodes and edges. The score is initialized at 0 for new nodes and edges, and it is updated for each line of the input. When processing an input line $\langle t, n_1, ..., n_k \rangle$, the score of each node $n_i | i \in \{1, ..., k\}$ is incremented by a value Δ_i:

$$\Delta_i = \frac{2}{k}.$$

Also the score of the edges $(n_i, n_j) | i, j \in (1, ..., k) \wedge i \neq j$ connecting nodes involved in the interaction are incremented by $\delta_{i,j}$:

$$\delta_{i,j} = \frac{2}{k(k-1)}.$$

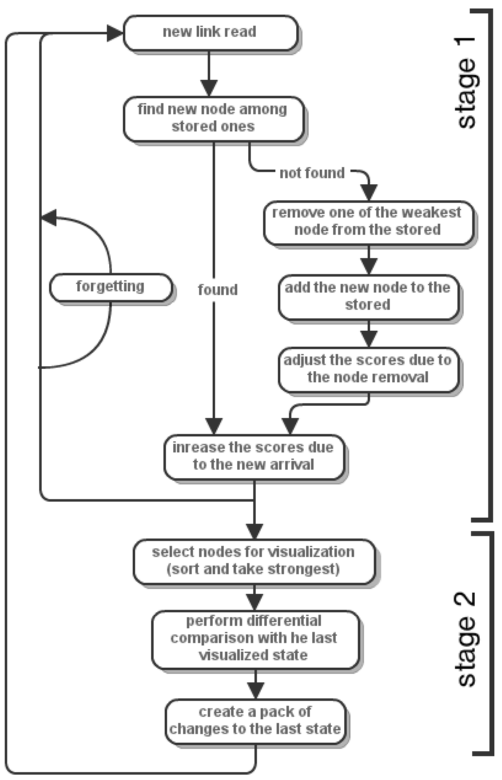
In general the increments to the scores can be adapted to the task, e.g. the above formulas give less importance to the interactions happening in large cliques. Alternatively, for one of our case studies we use another increments, defined as:

$$\Delta_i = 1, \qquad \delta_{i,j} = \frac{1}{k}. \tag{1}$$

To emphasize the most recent events and penalize stale ones, a forgetting mechanism that decreases the scores of all edges and nodes is run periodically every F_{forget} frames by multiplying the scores by a forgetting constant C_{forget}. The algorithm outputs for the purpose of the visualization N_v nodes with the highest scores, that are not singletons, and edges that have scores above a certain treshold S_{edge}^{min}.

The algorithm has two phases: buffering and generation of differential updates (see Figure 1). In the first stage, at most N_b nodes with the highest scores are saved in a buffer together with the interactions among them. Whenever a new node, that does not appear in the buffer yet, is read from the input, it replaces the node in the buffer with the lowest value of the score. If an incoming input line involves a node that is already in the buffer, then its score and scores of its edges are increased by Δ_i and $\delta_{i,j}$, respectively. In the second stage of the algorithm, the differential updates to the visualized part of the network are created. To this end, the N_v nodes in the buffer with the highest scores are selected. The subgraph induced by the N_v nodes is compared with the subgraph in the previous frame and a differential update is created. Each of the differential updates corresponds to a frame of the final visualization. The updates are created every time interval T_{frame}. The time interval is fixed and given at the beginning of the algorithm with the parameter corresponding to time contraction. Value of this parameter set to 10 means that the time will flow in the visualization 10 faster than in the data given as the input.

The differential updates are written in output in the form of a JSON file formatted according to the Gephi Stream-



new link read
find new node among stored ones
not found
remove one of the weakest node from the stored
add the new node to the stored
adjust the scores due to the node removal
forgetting
found
inrease the scores due to the new arrival
select nodes for visualization (sort and take strongest)
perform differential comparison with he last visualized state
create a pack of changes to the last state
stage 1
stage 2

Figure 1: Simple diagram of main components of the algorithm.

ing API (see `bit.ly/16uGJKm`). In short, each line of the JSON file corresponds to one update of the graph structure and contains a sequence of JSON objects that specify the addition/deletion/attribute change of nodes and edges. We introduced also a new type of object to deal with labels on the screen (for example to write the date and time on the screen).

Computational complexity

We call the numbers of buffered and visualized nodes N_b, and N_v, respectively. The computational complexity of the buffering stage of the algorithm is $\mathcal{O}(EN_b)$, where E is the total number of the pairwise interactions read (the cliques are made of multiple pairwise interactions). The memory usage scales as $\mathcal{O}(N_b^2)$. The second, frame-generating, stage has computational complexity of $\mathcal{O}(FN_v\log(N_v))$, where F is a total number of frames, that is a fraction of E and commonly it is many times smaller than E. The memory trace of this stage is very low and scales as $\mathcal{O}(N_v)$. We summarize, that our method has computational complexity linearly depending on number of interactions. Due to this characteristic it is able to deal fast with extremely large dynamical networks.

Visualization

The JSON stream produced by the algorithm is fed to a python module that builds a representation of a *dynamic graph*, namely an object that handles each of the updates and reflects the changes to its current structure. The transition between the structural states of the graph determined by received update can be depicted by a sequence of image frames. In its initial state, the nodes in the network are arranged according to the Fruchterman Rehingold graph layout algorithm [5]. For each new incoming event, a new layout is computed by running N iterations of the layout algorithm, using the previous layout as a seed. Intermediate layouts are produced at each iteration of the algorithm. Every intermediate layout is converted to a png frame that is combined through the mencoder tool (`bit.ly/zBryy`) to produce a movie that shows a smooth transition between different states. To avoid nodes and edges to appear or disappear abruptly in the movie, we use animations that smoothly collapse dying nodes and expand new ones. A configuration file allows to modify the default movie appearance (e.g, resolution, colors) and some layout parameters.

3. CASE STUDIES

We test our method on datasets very diverse by nature, size and time span. The movies produced from each dataset are attached to the submission, and are described next.

Datasets

Twitter

We use data downloaded with the Twitter *gardenhose* service, that covers around 10% of the tweet volume. We focus on two events: the announcement of Osama Bin Laden's death and the Super Bowl 2013. We consider user-mentions and hashtags as entities and their co-appearance in the same

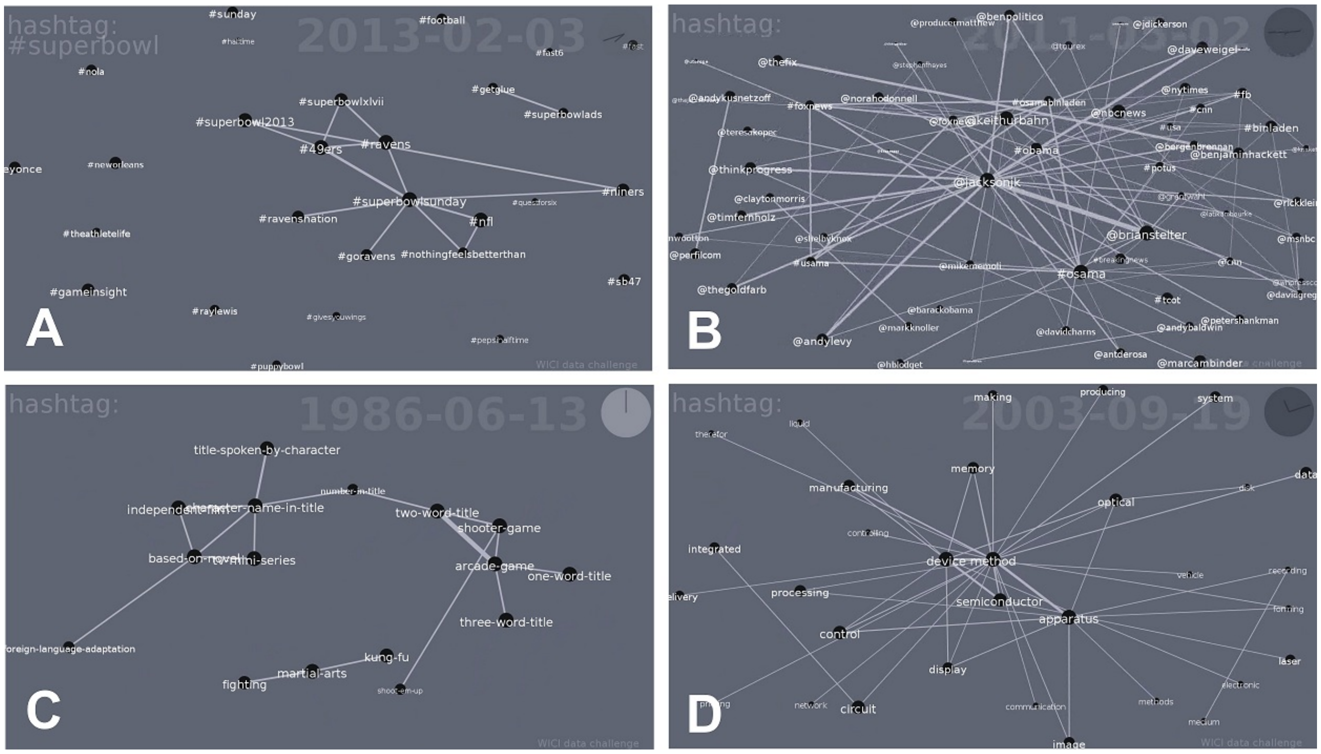


Figure 2: Screenshots of the movies generated from the datasets: A) SuperBowl, B) Bin Laden's death, C) IMDB keywords, D) US Patents.

tweet as interactions between them.

The video about the announcement of Bin Laden's death shows the initial burst caused by @keithurbahn and how the breaking news was being spread by users @brianstelter and @jacksonjk. The video shows that later the news appears in #cnn and is announced by @obama. The breaking of this event in Twitter is described in detail in `bit.ly/iPLoOc`.

The second video shows how the anticipation of Super Bowl steadily grows on early Sunday morning and afternoon, and how it explodes when the game is about to start. Hashtags related to #commercials and concerts e.g. #beyonce are evident. Later, the impact of the #blackout is clearly visible. The interest about the event drops rapidly after the game is over and stays low during the next day.

Patents

We use a set of US patents that were issued between 2003 and 2004. We analyse the appearance of words in their titles. Whenever two words appear in a title of a patent we create a link between them at the moment when the patent was issued. Our video demonstrates the the topics of interest at that period were "device", "semiconductor", and "apparatus".

IMDB movies

We use a dataset from IMDB of all movies, their year of release and all the keywords assigned to them (from `imdb.to/11SZD`). We create a network of keywords that are assigned to the same movies. For this dataset we use the score increments defined by Equation 1, due to the fact that the most popular movies have many keywords attached to them. Our video shows interesting evolution of the keywords from "character-name-in-title" and "based-on-novel" (first half of 20th century), through "martial-arts" (70s and 80s) to "independent-film" (90s and later), 'anime" and "surrealism' (21st century)'.

Discussion

	Period	Nodes	Edges	Nodes drawn
Bin Laden Death	2h	95k	198k	291
Super Bowl	2d	49k	1.1M	170
US patent title words	6m	24k	190k	76
IMDB movie keywords	107y	101k	220M	129

Table 1: Statistics on the experimental datasets

The datasets that we use are fairly diverse in time span, topicality and size, as shown in Table 1. Nevertheless, our method is able to narrow down the visualization to a meaningful small subgraphs with less than 300 nodes for all cases. The high performance of the algorithm makes it viable for real-time visualizations of live and large data streams. On a desktop machine the algorithm producing differential updates of the network in the form of JSON files took less than 20 seconds to finish for most of the datasets (80 for IMDB). Given such performance, it is possible to visualize in real-time highly popular events such as Super Bowl, that produces up to 4500 tweets per second.

Figure 2 displays some frames of the videos. Other than those experimental datasets, on-demand videos of Twitter hashtag co-occurrences graphs, mention networks or retweet networks can be generated with our tool via the Truthy service (`truthy.indiana.edu/movies`). Hundreds of videos have been already generated by the users of the platform and are available to view. We note that our algorithm can also stream its results directly to Gephi, and that the user can interact with the dynamic network that it produces. Instructions attached to the submission explain how to test the Gephi visualization.

4. REFERENCES

[1] M. Bastian, S. Heymann, and M. Jacomy. Gephi: an open source software for exploring and manipulating networks. In *ICWSM'09: Proceedings of the International AAAI Conference on Weblogs and Social Media*. AAAI, 2009.

[2] W. Broeck, C. Gioannini, B. Goncalves, M. Quaggiotto, V. Colizza, and A. Vespignani. The gleamviz computational tool, a publicly available software to explore realistic epidemic spreading scenarios at the global scale. *BMC Infectious Diseases*, 11(1):37, 2011.

[3] A. Dutot, F. Guinand, D. Olivier, and Y. Pigné. Graphstream: A tool for bridging the gap between complex systems and dynamic graphs. In *EPNACS: Emergent Properties in Natural and Artificial Complex Systems*, 2007.

[4] C. Erten, P. Harding, S. Kobourov, K. Wampler, and G. Yee. Graphael: Graph animations with evolving layouts. In G. Liotta, editor, *Graph Drawing*, volume 2912 of *Lecture Notes in Computer Science*, pages 98–110. Springer Berlin, 2004.

[5] T. M. J. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Software Practice and Experience*, 21:1129–1164, November 1991.