Fast animation 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 an open challenge in several domains. We present 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 algorithm, which is already deployed in the movie generation tool of the truthy.indiana.edu system, uses limited memory and processor time, and we release it as open-source software.

1. INTRODUCTION
Network visualization is widely adopted to make sense of, and gain insight from, complex and large interaction data. These visualizations are typically static, and incapable to deal with quickly changing networks. Dynamic graphs, where nodes and edges churn and change over time, can be effective means of visualizing evolving networked systems such as social media, similarity graphs, or interaction networks between real world entities. The recent 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 several tools [18, 13, 5, 16]. In particular, Gephi [5] supports graph streaming with a dedicated API based on JSON events and enables the association of timestamps to each graph component.

While there is some literature on dynamic layout of graphs [8, 9, 10], not much work has been done so far about developing information selection techniques for dynamic visualization of large and quickly changing networks. Yet, 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 represent and transmit the salient structural properties of the network at a certain time is crucial to produce meaningful visualizations of the graph evolution.

In this paper, we contribute to filling this gap by presenting a new graph animation tool 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; and
- is fast enough to be applied to large live data streams and visualize their representation in form of a network.

In Appendix A we present the source code of the visualization tools with the related documentation, four dynamic graph datasets and instructions to recreate respective visualizations introduced in the latter part of this manuscript.

2. RELATED WORK
Graph drawing [30, 14] is a branch of information visualization that has acquired great importance in complex systems analysis. The rapid development of computer-aided visualization tools and the refinement of graph layout algorithms [31, 23, 24, 29] allowed increasingly higher-quality visualizations of large graphs [28]. As a result, many open tools for static graph analysis and visualization have been developed in the last decade. Among the most known we mention Walrus [36], Pajek [6, 14], Visone [11], GUESS [1], Networkbench, and NodeXL [42].

Visualization of dynamic graphs has received considerable attention recently. Despite static visualizations based on sliding time windows [2], alluvial diagrams [41], or matrices [45, 43, 25] have been explored as solutions to capture the graph evolution, dynamic graph drawing remains the technique that has attracted more interest in the research community so far.

Much work has been done to define efficient update operations on graphs [39, 27] and also in the definition of criteria and guidelines for a good visualization of graph evolution with animations [21], with the goal of preserving the mental map [35, 20] that the observer has of the graph structure. Methods to preserve the stability of nodes and the consistency of the network structure leveraging hierarchical
organization on nodes have been proposed \cite{37,38,3}. More in general, some work has been done to adapt spectral and force-directed graph layouts \cite{7} to incremental layouts that recompute the position of nodes at time $t$ based on the previous positions at time $t-1$ minimizing displacement of vertices \cite{12,13,8,22} or to propose new "stress-minimization" strategies to map the changes in the graph \cite{10}.

Recently, the interest in depicting the shape of online social networks \cite{26,19} and the availability of live data streams from online social media motivated the development of interfaces to process and visualize the evolution of graphs in time. Dynamic visualization is supported by tools like GraphAEL \cite{18}, GleamViz \cite{13}, Gephi \cite{5}, and GraphStream \cite{16}. Exploratory work on information selection techniques for dynamic graph visualization has been done in the past, including solutions based on temporal decay of nodes and edges \cite{17}, node clustering \cite{40}, and centrality indices \cite{32,4}. However, none of the modern visualization tools provide features for the detection of the most relevant components of a graph at a given time.

3. 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 illustrate how to convert the sequence of updates into image frames that can be combined into a movie depicting the network evolution. Alternatively, the updates can be fed directly to the 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. Repetition of the same entry with 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 $\Delta_i$:

$$\Delta_i = \frac{2}{k}.$$ \hfill (1)

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

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

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

$$\Delta_i = 1, \quad \delta_{i,j} = \frac{1}{k}.$$ \hfill (3)

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 $F_{\text{forget}}$ frames by multiplying the scores by a forgetting constant $C_{\text{forget}}$. 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 threshold $S_{\text{min}}$.

![Figure 1: Simple diagram of main components of the algorithm.](image)

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 $\Delta_i$ and $\delta_{ij}$, 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 per every time interval, that is determined at the beginning of the algorithm with the parameter corresponding to time contraction $T_{contra}$. 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 Streaming 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 also introduced 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 $O(E N_b)$, where $E$ is the total number of the pairwise interactions read (the cliques are made of multiple pairwise interactions). The memory usage scales as $O(N_b^2)$. The second, frame-generating, stage has computational complexity of $O(F N_v \log(N_v))$, 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 $O(N_v)$. We summarize, that our method has computational complexity that scales linearly with the number of interactions. It is therefore fast, that is, able to deal with extremely large dynamic networks efficiently.

**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 the 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 [24]. 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/2Bryy) 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.

4. **CASE STUDIES**

We test our method on datasets very different from each other in nature, size, and time span. The datasets and movies produced from each dataset are described next. Additionally, we publish them online (see Appendix A and Additional Materials).

**Twitter**

We use data obtained through the Twitter *gardenhose* streaming API, that covers around 10% of the tweet volume. We focus on two events: the announcement of Osama Bin Laden’s death and the 2013 Super Bowl. We consider user mentions and hashtags as entities and their co-occurrence in the same tweet as interactions between them.

The first video (Figure 2A) shows how the anticipation for the 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.

The video about the announcement of Bin Laden’s death (Figure 2B) shows the initial burst caused by @keithurbaum and how the breaking news was spread by users @brianstelter and @jacksonjk. The video shows that the news appears later in #cnn and is announced by @obama. The breaking of this event in Twitter is described in detail by Lotan [34].

**IMDB movies**

We use a dataset from IMDB of all movies, their year of release and all the keywords assigned to them (from 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 8 due to the fact that the most popular movies have many keywords attached to them. Our video (Figure 2C) 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” (2000s).

**Patents**

We use a set of US patents that were issued between 1976 and 2010 [33]. We analyze the appearance of words in their titles. Whenever two or more words appear in a title of a patent we create a link between them at the moment when the patent was issued. To improve readability we filter out stopwords and the generic frequent words: “method”, “device” and “apparatus”. Our video (Figure 2D) demonstrates that at the beginning of the period techniques related to “engine” and “combustion” were popular, and later start to cluster together with “motor” and “vehicle”. Another cluster is sparked by patents about “magnetic” “recording” and “image” “processing”. It merges with a cluster of words related to “semiconductor” and “liquid” crystal” to form the largest cluster of connected keywords at the end of the period.

**Discussion**

The datasets in our case studies are fairly diverse in topicality, time span, and size, as shown in Table 1. Nevertheless, our method is able to narrow down the visualization
to meaningful small subgraphs with less than 300 nodes in 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 several minutes to finish for the US patents and less than two minutes for the other datasets. Given such a performance, it is possible to visualize in real-time highly popular events such as Super Bowl, which produced up to 4500 tweets per second. For the explanatory purposes we provide and describe values of the parameters of the algorithm for the diversified case studies in the Appendix B.

Table 1: Statistics on the experimental datasets.

| Dataset                  | Period | Nodes | Edges | Nodes drawn |
|--------------------------|--------|-------|-------|-------------|
| Bin Laden Death          | 2h     | 95k   | 198k  | 291         |
| Super Bowl               | 2d     | 49k   | 1.1M  | 170         |
| IMDB movie keywords      | 106y   | 101k  | 220M  | 129         |
| US patent title words    | 34y    | 414k  | 90M   | 68          |

Other than these experimental datasets, on-demand animations of Twitter hashtag co-occurrence and diffusion (retweet and mention) networks can be generated with our tool via the Truthy service [truthy.indiana.edu/movies]. Hundreds of videos have already been generated by the users of the platform and are available to view on YouTube [youtube.com/user/truthyatindiana/videos].

As already noted, our algorithm can also stream its results directly to Gephi, and the user can interact with the dynamic network that it produces. In Appendix A we explain how to test the Gephi visualization.

5. CONCLUSIONS AND FUTURE WORK

Tools for dynamic graph visualization developed so far do not provide any way to dynamically select the most important portions of large evolving graphs. We contribute to fill this gap by proposing an algorithm to select nodes and edges that best represent the network structure at a given time. We implemented our approach in an open source tool that takes in input a stream of interaction data and outputs a movie of the network evolution or a live Gephi animation. As future work, we wish to improve our algorithm by means of further optimization and to enhance the tool by providing a standalone module for live visualization of the graph evolution, as well as generalizing our method to weighted networks.

Author’s contributions

All authors designed the research. PAG wrote the source code of the algorithm and LMA wrote the source code of the visualization tool. All authors deployed the tools. PAG and LMA analyzed the data. All authors wrote, reviewed and approved the manuscript.

Acknowledgements

We are grateful to André Panisson for inspiration and to Jacob Ratkiewicz, Bruno Gonçalves, Mark Meiss, and other
by the NSF (ICES award CCF-1101743), and the James S. McDonnell Foundation.

6. REFERENCES

[1] E. Adar. Guess: a language and interface for graph exploration. In Proceedings of the SIGCHI conference on Human Factors in computing systems, CHI ’06, New York, NY, USA, 2006. ACM.

[2] J.-W. Ahn, M. Taieb-Maimon, A. Sopan, C. Plaisant, and B. Shneiderman. Temporal visualization of social network dynamics: prototypes for nation of neighbors. In SBP’11: Proceedings of the 4th international conference on Social computing, behavioral-cultural modeling and prediction, Berlin, Heidelberg, 2011. Springer.

[3] D. Archambault. Structural differences between two graphs through hierarchies. In GI’09: Proceedings of Graphics Interface, Toronto, Ont., Canada, Canada, 2009. Canadian Information Processing Society.

[4] S. Asur, S. Parthasarathy, and D. Ucar. An event-based framework for characterizing the evolutionary behavior of interaction graphs. In KDD’07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining, New York, NY, USA, 2007. ACM.

[5] 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.

[6] V. Batagelj and A. Mrvar. Pajek - analysis and visualization of large networks. In Graph Drawing, volume 2265 of Lecture Notes in Computer Science. Springer, 2002.

[7] U. Brandes. Drawing on physical analogies. In Drawing graphs. Springer, London, UK, 2001.

[8] U. Brandes, D. Fleischer, and T. Puppe. Dynamic spectral layout of small worlds. In GD’05: Proceedings of the 13th International Symposium on Graph Drawing. Springer, 2005.

[9] U. Brandes, D. Fleischer, and T. Puppe. Dynamic spectral layout with an application to small worlds. Journal of Graph Algorithms and Applications, 11(2), 2007.

[10] U. Brandes, N. Indlekofer, and M. Mader. Visualization methods for longitudinal social networks and stochastic actor-oriented modeling. Social Network, 34(3), 2011.

[11] U. Brandes and D. Wagner. Visone - analysis and visualization of social networks. In Graph Drawing Software. Springer, 2003.

[12] J. Branke. Dynamic graph drawing. In Drawing graphs. Springer, London, UK, 2001.

[13] 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), 2011.

[14] W. De Nooy, A. Mrvar, and V. Batagelj. Exploratory Social Network Analysis with Pajek. Cambridge University Press, 2005.

[15] S. Diehl and C. Gorg. Graphs, they are changing. In Graph Drawing, volume 2528 of Lecture Notes in Computer Science. Springer, 2002.

[16] 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.

[17] S. B. C. Dynes, P. A. Gloor, P. A. Gloor, P. A. Gloor, R. Laubacher, R. Laubacher, Y. Zhao, Y. Zhao, and S. Dynes. Temporal visualization and analysis of social networks. In NAACSOS’04: Conference of North American Association for Computational Social and Organizational Science, 2004.

[18] C. Erten, P. Harding, S. Kobourov, K. Wampler, and G. Yee. Graphael: Graph animations with evolving layouts. In Graph Drawing, volume 2912 of Lecture Notes in Computer Science. Springer Berlin, 2004.

[19] T. Falkowski, J. Bartelheimer, and M. Spiliopoulou. Mining and visualizing the evolution of subgroups in social networks. In Wit06: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence, Washington, DC, USA, 2006. IEEE Computer Society.

[20] M. Freire and P. Rodriguez. Preserving the mental map in interactive graph interfaces. In AVI’06: Proceedings of the working conference on Advanced visual interfaces, New York, NY, USA, 2006. ACM.

[21] C. Friedrich and M. Houle. Graph Drawing in Motion II. 2002.

[22] Y. Frishman and A. Tal. Online dynamic graph drawing. IEEE Transactions on Visualization and Computer Graphics, 14, 2008.

[23] T. M. J. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. Software Practice and Experience, 21(11), 1991.

[24] E. R. Gansner and S. C. North. Improved force-directed layouts. In GD’98: Proceedings of the 6th International Symposium on Graph Drawing, London, UK, 1998. Springer.

[25] R. Gove, N. Gramsky, R. Kirby, E. Sefer, A. Sopan, C. Dunne, B. Shneiderman, and M. Taieb-Maimon. NetVisia: Heat Map & Matrix Visualization of Dynamic Social Network Statistics & Content. In SocialCom ’11, Proceedings of the 3rd IEEE Second International Conference on Social Computing, Washington, DC, USA, 2011. IEEE Computer Society.

[26] J. Heer and D. Boyd. Vizster: Visualizing online social networks. In InfoVis ’05: Proceedings of the IEEE Symposium on Information Visualization, Washington, DC, USA, 2005. IEEE Computer Society.

[27] M. R. Henzinger and V. King. Randomized fully dynamic graph algorithms with polylogarithmic time per operation. J. ACM, 46, 1999.

[28] I. Herman, G. Melançon, and M. S. Marshall. Graph Visualization and Navigation in Information Visualization: A Survey. IEEE Transactions on
Visualization and Computer Graphics, 6(1), 2000.

[29] Y. F. Hu. Efficient and high quality force-directed graph drawing. The Mathematica Journal, 10(1), 2005.

[30] T. Kamada. Visualizing Abstract Objects and Relations. World Scientific, 1989.

[31] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. Information Processing Letters, 31(1), 1989.

[32] G. Kumar and M. Garland. Visual exploration of complex time-varying graphs. IEEE Transactions on Visualization and Computer Graphics, 12, 2006.

[33] G. LaRowe, S. Ambre, J. Burgoon, W. Ke, and K. BÄürner. The scholarly database and its utility for scientometrics research. Scientometrics, 79(2):219–234, 2009.

[34] G. Lotan. Breaking bin laden: A closer look, 2011. http://bit.ly/iPLOoC.

[35] K. Misue, P. Eades, W. Lai, and K. Sugiyama. Layout adjustment and the mental map. Journal of Visual Languages and Computing, 6(2), 1995.

[36] T. M. Munzner. Interactive visualization of large graphs and networks. PhD thesis, Stanford, CA, USA, 2000.

[37] S. C. North. Incremental layout in dynadag. In GD‘93: Proceedings of the Symposium on Graph Drawing, London, UK, 1996. Springer.

[38] S. C. North and G. Woodhull. Online hierarchical graph drawing. In GD‘01: Revised Papers from the 9th International Symposium on Graph Drawing, London, UK, 2002. Springer.

[39] G. Ramalingam and T. Reps. On the computational complexity of dynamic graph problems. Theoretical Computer Science, 158(1), 1996.

[40] E. M. Rodrigues, N. Milic-Frayling, M. Smith, B. Shneiderman, and D. Hansen. Group-In-a-Box Layout for Multi-faceted Analysis of Communities. In SocialCom‘11, Proceedings of the 3rd IEEE Second International Conference on Social Computing, Washington, DC, USA, 2011. IEEE Computer Society.

[41] M. Rosvall and C. T. Bergstrom. Mapping change in large networks. PLoS ONE, 5(1), 01 2010.

[42] M. A. Smith, B. Shneiderman, N. Milic-Frayling, E. Mendes Rodrigues, V. Barash, C. Dunne, T. Capone, A. Perez, and E. Gleave. Analyzing (social media) networks with nodexl. In CET’09:Proceedings of the fourth international conference on Communities and technologies, New York, NY, USA, 2009. ACM.

[43] K. Stein, R. Wegener, and C. Schlieder. Pixel-oriented visualization of change in social networks. In ASONAM’10: Proceedings of the International Conference on Advances in Social Networks Analysis and Mining, Washington, DC, USA, 2010. IEEE Computer Society.

[44] I. G. Tollis, G. Di Battista, P. Eades, and R. Tamassia. Graph Drawing: Algorithms for the Visualization of Graphs. Prentice Hall, 1999.

[45] J. S. Yi, N. Elmqvist, and S. Lee. Timematrix: Analyzing temporal social networks using interactive matrix-based visualizations. International Journal of Human-Computer Interaction, 26(11-12), 2010.

APPENDIX

A. SOURCE CODE

We have implemented two independent tools described in the manuscript. The first tool takes in a chronological stream of interactions between nodes in the format that we call sdnet (standing for sorted dynamic network) and converts it into a set of graph updates that account only for the most relevant part of the network in the JSON format. The second tool converts the sequence of updates into image frames that can be combined into a movie depicting the network evolution. We release the source code of both of the tools with the documentation under the GNU General Public License (see the project github.com/WICI/fastviz). Together with the tools we release the datasets used in this paper and instructions on how to recreate all the examples given in the manuscript. Alternatively, the updates can be fed directly from the first tool to the Gephi Streaming API to produce an interactive visualization of the evolving network. Respective instructions can be found at github.com/WICI/fastviz

WICI/fastviz#launching-interactive-visualizations.

B. ALGORITHM PARAMETERS

The exact behavior of the algorithm depends on the parameters introduced in the manuscript. We demonstrate the values of the parameters used in the case studies and their default values in the Table 2. The default values of the parameters are meant to be universal and give reasonably good visualizations for any dataset. The two parameters that need to be chosen carefully depending on the dataset at hand are the time contraction Tcontr and the minimal edge score smin. The time contraction corresponds to the number of seconds in data time scale that are going to be contracted to one second of the visualization. The larger the time span of the dataset the larger should be this parameter in order to keep the length of visualization fixed. The minimal edge score is a threshold above which edges appear in the visualization. Low value of this parameter may results in many edges of low weight appearing in the animation, while high value of the parameter may prevent any edges from being visualized.
Table 2: Values of the parameters of the algorithm for the introduced case studies. The last row contains the default values of the corresponding parameters of the algorithm.

| Dataset                        | $\Delta_i, \Delta_{ij}$ | $T_{contr}$ | $N_h$ | $N_v$ | $S_{\text{min}}$ | $C_{\text{forget}}$ | $F_{\text{forget}}$ |
|-------------------------------|--------------------------|-------------|-------|-------|------------------|----------------------|---------------------|
| Bin Laden Death               | Eqs.1-2                  | 500         | 50    | 2000  | 0.95             | 0.6                  | 40                  |
| Super Bowl                    | Eqs.1-2                  | 3600        | 50    | 2000  | 10               | 0.6                  | 20                  |
| IMDB movie keywords           | Eqs.3                    | 3600-24-1095| 80    | 2000  | 10               | 0.75                 | 10                  |
| US patent title words         | Eqs.1-2                  | 3600-24-400 | 50    | 2000  | 20               | 0.65                 | 10                  |
| Default                       | Eqs.1-2                  | 3600        | 50    | 2000  | 0.95             | 0.75                 | 10                  |