Generating Events for Dynamic Social Network Simulations

Pascal Held, Alexander Dockhorn, and Rudolf Kruse

Department of Knowledge Processing and Language Engineering
Faculty of Computer Science
Otto von Guericke University of Magdeburg
pascal.held@ovgu.de
alexander.dockhorn@st.ovgu.de
rudolf.kruse@ovgu.de
http://fuzzy.cs.uni-magdeburg.de

Abstract. Social Network Analysis in the last decade has gained remarkable attention. The current analysis focuses more and more on the dynamic behavior of them. The underlying structure from Social Networks, like facebook, or twitter, can change over time. Groups can be merged or single nodes can move from one group to another. But these phenomenas do not only occur in social networks but also in human brains. The research in neural spike trains also focuses on finding functional communities. These communities can change over time by switching the stimuli presented to the subject. In this paper we introduce a data generator to create such dynamic behavior, with effects in the interactions between nodes. We generate time stamps for events for one-to-one, one-to-many, and many-to-all relations. This data could be used to demonstrate the functionality of algorithms on such data, e.g. clustering or visualization algorithms. We demonstrated that the generated data fulfills common properties of social networks.

1 Introduction

Social Network Analysis is a hot topic since some years. The first investigations where done in a static analysis of the networks. Examples for this are networks representing friendship, or co-author relationships, between people. But social networks, like facebook, or Twitter, are very dynamic. So a static view is too simple. In [8] we described some algorithms which are based on events between nodes. This could be e.g. an interaction between users of a social network, or interactions between neurons in human brains.

The human brain consists of different regions. In each region there are a lot of neurons which are connected. These connections cause that stimuli from outside are passed through this network of neurons by fire events of neurons. A group of neurons which handle the same stimuli are called functional groups or communities. The discovery of such functional communities is very similar to social networks analysis. [14] The history of fire events of a single neuron is called spike train.

There are some data sets available, e.g. the Enron Dataset, extracts from twitter, or recordings from human brains. These are all real data sets, where we know nothing about the base structure of the underlying data.

In this paper we present a data generator to generate events for social networks or fire events in spike trains. This data represents interaction between different nodes in the network. We do not only represent static behavior, but also dynamics in the underlying structure. So we can configure the generator in a way, that clusters of nodes are changing, like in real social structures.

The generated data is based on clusters of nodes in a graph. Nodes within the same cluster have a higher communication frequency than nodes in different clusters. During the simulation it is possible to generate new clusters, or modify them by adding nodes, moving nodes from one to another cluster, merging with other clusters, split up the clusters.

The rest of the paper is organized as follows. In the second section we present some fundamentals in the field of social networks and human brain spike trains. The third section describes the generator in detail, followed by experiments with the generated data in Section 4. In the last section we discuss our results.

2 Fundamentals

The generation of graphs, especially in social network analysis is nothing new. There are a lot of generators, like the Kronecker graph generators [10] or the generator from McGlohon et al. [12]. The most algorithms generate graphs which hold some typical properties or have problems with a growing number of nodes. Akoglu and Faloutsos developed the realistic graph generator [1] which holds most typical properties and also enables the evolving of graphs. A good survey of different generators is given in [3].

The main drawback of these algorithms is that evolution of the graph in most cases is a static growing, where new nodes and edges are created. In some cases there are also changes in the connections, but the main structure is constant.

Our focus is to create a graph, where we know the main structure of the underlying communities, so we have a ground truth to check cluster algorithms on dynamic graphs. Also we want to be able to change this ground structure.

Another point is that we are primary not interested in the graph itself, but on the events between the nodes, which could be used to create such a graph.

In the following we present same requirements for spike train simulation as well as for social network generation.

2.1 Simulating Spike Trains

A neuron in the human brain can be simulated as a list of events. An event occurs when the potential of the concerning neuron increases. This means that the neuron fires. Such an event list is commonly named spike train.

The easiest way to simulate such spike trains is a point process simulation based on a Poisson process model [15]. This model is based on the assumption

that the probability of a neuron firing in a given time frame $[t, t + \delta t]$ is simply given by $R\delta t$ where R is the firing rate, for sufficient small δt. The probability is absolutely independent from the position of the last firing.

One interpretation of interacting neurons are ensembles in parallel spike trains [11]. The main idea is, that neurons that fire together are wired together. Borgelt et al. generate data parallel spike trains with multiple Poisson processes. Every spike train has one generator process. Additionally one process for every ensemble is present. The events from the ensemble process will be copied into the individual spike trains with a given probability. These probabilities describes how close the neurons are interacting with each other. The individual point processes of neurons within an ensemble have a lower fire rate, so the combined fire rate together with the common fire events is the same as from other neurons.

2.2 Social Networks

One aspect of our data generator is the simulation of interactions in social networks. For this simulation the generated data should have similar characteristics as real social networks. In this section we would like to present some of these characteristics and how to show them. These are the small-world property, scale-free characteristics, and the structure of communities and clusters.

The Small-World Property was introduced by Watts and Strogatz [16] for graphs with social network characteristics in 1998. It is based on the six degree of separation from Guare [5]. They focused on the average shortest path between two nodes (global connectivity measure) L and the average clustering coefficient (local neighborhood connectivity measure) C. In their experiments they started with a regular graph where the nodes are placed in a ring. Every node is connected to the k following and previous nodes. With a probability of p they replaced a given edge by a new random one. For $p = 0.0$ this yield to the original graph and $p = 1.0$ yield to a total random graph, with the same amount of nodes and edges. They compared three graphs with social network character with random graphs of the same complexity (same number of nodes and edges) and proofed that they all follow the small-world model, which means that $L \gtrsim L_{random}$ and $C \gg C_{random}$.

The Scale-free characteristic is a another property of social networks. Typically is that not every node has the same connectivity degree. There are some nodes with a strong connectivity to others and other nodes with much fewer connections. This is based on the fact, that such networks grow over time. New nodes connect themselves more probable with nodes with a strong connectivity. For example, a new person in a friendship network will connect first to high connected friends then to other new nodes, or a new website will link to a known common website then to another new one. Barabasi and Albert investigate in 1999 [2] this phenomenon. They found out that the degree distribution of the nodes from social networks follow a power-law distribution.

To proof this property in the generated data, we will run the same experiments and compare the L and C values with equivalent random graphs.

Communities and Clusters are the core phenomenon in social network. People organize in groups and these groups could be recognized in communities in social networks. Such a group could be a group of people from the same university or a clique of friends.

The main concept of this groups is that the connectivity within a group is much denser then the connectivity to elements outside of the group. In the field of data mining this is called the intra-cluster-density and the inter-cluster-sparseness. In our work we get this property by construction. The data generator itself supports the modeling of such clusters.

Social Networks are dynamic and not static. They evolve over time. New elements join the network, clusters are emerging, splitting, growing, or shrinking. Nodes change from one cluster to another. In [7] we describe how to analyze this dynamics in clusters.

With our data generator we can generate communication events for such dynamics in social networks.

3 Data Generator

We already introduced a broad area of applications for graphs. However, all variants require the graph structure or the type of output to comply special constraints. For that reason the generator needs to be easily adjustable for a multitude of network characteristics. The implemented data generator is able to produce different types of static and dynamic graphs. It consists of methods for defining the structure of the start graph and allows the import of scripts, which describe changes to specified points in time. The basic behavior of the generator and script functionalities will be outlined in the following sections.

The graph is divided into several clusters. Each network structure of a cluster is generated using the model proposed by Barabási in [2]. This model proposes a growing network starting with m_0 nodes. Further nodes are sequentially added to the graph and connected using $m(\leq m_0)$ edges. The probability of a node having k edges follows a power-law with an exponent of $y_{model} = 2.9 \pm 0.1$. By definition the network structure of each cluster will organize itself into a scale-free stationary state. We use the same model to decide which clusters are connected. Connecting nodes will be drawn at random.

The firing rates and intervals in a cluster can be configured by entering constant values or a distribution where values are drawn of. The generator therefore supports the creation of poisson processes by using gamma distributions to determine time intervals between two events as it was modeled in [15]. Nodes and clusters store corresponding parameters and can be manipulated through script functions as it will be explained later.

Generated events will be stored and exported in a *.csv-file*. The respective format can be set in the command line and be of one of the following types:

Communication with one source and single/multiple targets: Every entry will contain the time of the event, the id of the source node and targets of the communication. "$time; id_{source}; id_{target(s)}$"

Groups of nodes active at the same time: This format does not include information about the direction of any communication and stores simultaneously active nodes per time frame. "$time; id_{source(s)}$"

Activity of single nodes: The third format records undirected potentials per node and can be used to record spike trains. "$time; id_{node}$"

The main focus of our designed program lies on enabling the user to preconfigure changes of the network in a script file. This way analysis techniques can be brought to the test for already known features of the dynamic graph. Currently provided script-functions and their possible applications are listed below:

Delete/add nodes/clusters: Nodes and clusters can be added/deleted at specified time points. E.g. new people are joining a network, a neuron dies

Change behavior of individual nodes: Adjust firing rates, activity or cluster assignments per node. E.g. a person starts to communicate more frequently, communication partners change because of a new job, a neuron fires more often because a stimulus changed

Change behavior of clusters: Adjust activity of the whole cluster. E.g. a group of people starts to communicate more often to organize an event, a brain region changes activity level while sleeping, the visual part of the brain adapts to changes in visual stimuli

Divide or merge clusters: Multiple clusters are merged to form a new one, a cluster will be divided in parts. E.g. circle of friends splits after finishing school, departments are joined to minimize communication costs, special stimuli only activate specific parts of the brain

4 Experiments

The generator was tested on the simulation of spike trains and social networks. The former was compared to another generator used by Borgelt el al. [11] which is based on a model from Nawrot et al. [13]. We evaluated different measures relating to spike trains and social networks. Our test cases will be presented in the following sections. Our evaluation platform was an HP Z400 with 6GB RAM and a 3.45 GHz 6-core Intel Xeon processor. However the program only used one core for the generation process. Event generation for our biggest test-case (1000 Cluster, 100 000 Nodes and 25 000 000 Events) took about 7 minutes.

4.1 Spike Trains

In spike trains the activity potentials per node are recorded. Therefore we chose the third output type for our generator. Both tools were used to generate 50

nodes, 10 forming a single ensemble and 40 nodes for noise. Events were recorded over ten seconds. The nodes firing rates were set to an average of 20 events per second. The ensemble had a copy rate of 50 percent, therefore an average of half of the ensemble nodes participated on events of the whole ensemble.

First we checked for similarity of individual spike trains of both tools by using a Kolmogorow-Smirnow test for the distributions of time intervals between consecutive events per node. The average of recorded p-values was ~ 0.55. For this reason we can accept the hypothesis that both distributions were drawn from the same continuous distribution.

$$d_{Correlation} = \frac{1}{2} - \frac{n_{11}n_{00} - n_{01}n_{10}}{2\sqrt{(n_{10}+n_{11})(n_{01}+n_{00})(n_{11}+n_{01})(n_{00}+n_{10})}} \tag{1}$$

$$d_{Yule} = \frac{n_{01}+n_{10}}{n_{11}n_{00} - n_{01}n_{10}} \tag{2}$$

$$d_{Jaccard} = \frac{n_{10}+n_{01}}{n_{11}+n_{10}+n_{01}} \tag{3}$$

$$d_{Hamming} = \frac{n_{01}+n_{10}}{n_{**}} \tag{4}$$

We continued the analysis of our generator by binning resulting spike trains into 1000 bins of size 0.01 second to transform them into a binary matrix. This was used to calculated distance matrices for the four distance measures Correlation (1), Yule (2) [17], Jaccard (3) [9], and Hamming (4) [6]. The results are shown in Figure 1. The distance matrices for hamming and yule distance show clearly the similarity of the first ten nodes forming an ensemble. Distances of noise-noise and noise-ensemble combinations were much higher.

4.2 Social Network

Our second evaluation involves the generation of social networks. We created four graphs with the following configurations.

- Graph 1 = 5 clusters with 20 nodes each
- Graph 2 = 3 clusters with 50 nodes each
- Graph 3 = 5 clusters with 10, 20, 30, 40 and 50 nodes
- Graph 4 = 25 clusters with 25 nodes each

Noise was excluded in all three generation processes. The activity and communication intervals per node were gamma-distributed with $\alpha = 1$ and $\beta = 0.35$.

We compared the average path length and the average cluster coefficient of the generated graphs to random graphs with same number of total edges. Recorded values are shown in Table 1.

It has been shown that for all three generated graphs the small-world model ($L \gtrsim L_{random}$ and $C \gg C_{random}$) is applicable. We used the same set of graphs to test for scale-free characteristics. We used the procedure, described in [4]. It uses a bootstrapping hypothesis test to maximize the Kolmogorow-Smirnow

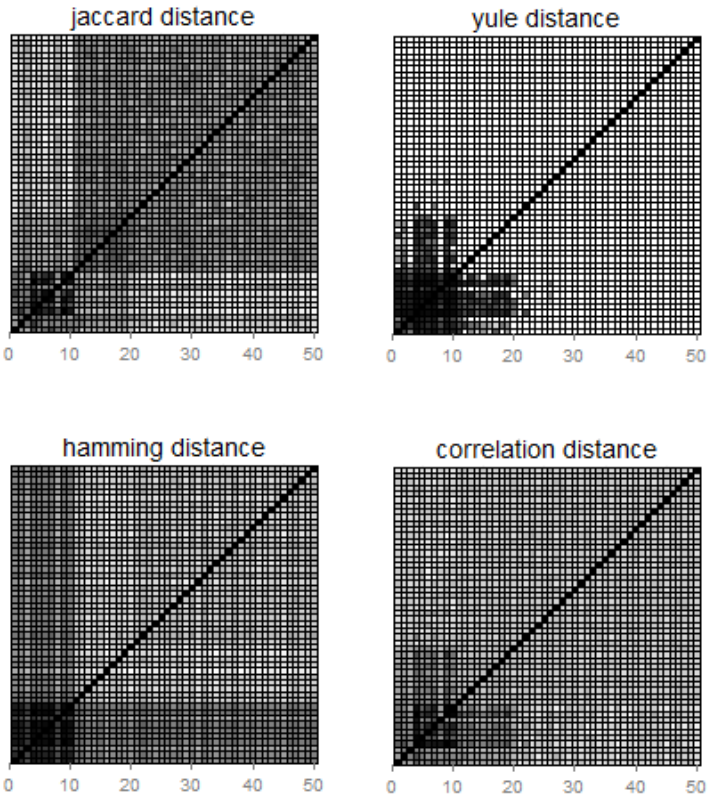

Fig. 1. Distance matrices for jaccard, correlation, yule and hamming distance

statistic to test for a goodness-of-fit between the data and the power law. The resulting p-values need to be ≥ 0.1 to accept the power-law distribution as a plausible hypothesis. Responding p-values per graph are shown in Table 1. The results were all significant (≥ 0.1) and therefore a power-law distribution can be accepted as a hypothesis.

Forming communities and clusters is directly inferred by the generation process. Each cluster will be generated as a separate graph using the model proposed by Barabási in [2]. Noise can be added by defining a base probability for inter-cluster-events for a whole cluster or by adding specified noise nodes. Dependent on the chosen cluster definition it is possible to design clusters of nodes which are active at the same time, in equal distributed time intervals or are more likely to talk to each other than to nodes from different clusters.

To show the dynamic capabilities of our generator we created graphs with two clusters of 20 nodes each and used a simple script, which included the following commands:

	Graph 1	Graph 2	Graph 3	Graph 4
$L_{generator}$	4.30	4.73	4.38	6.45
L_{random}	2.83	2.97	2.96	3.28
$C_{generator}$	0.40	0.27	0.30	0.42
C_{random}	0.07	0.05	0.35	0.14
p	0.5	0.5	0.5	0.9

Table 1. Measured average shortest path length and clustering coefficient per graph compared to a random graph with same number of edges, p-values for accepting a power-law distribution as plausible hypothesis

- t_1, move five nodes from cluster one to cluster two
- t_3, add five new nodes to cluster one
- t_5, remove five nodes from cluster two
- t_7, lower the cluster activity of cluster two to 50%
- t_9, set cluster activity of cluster two back to 100%

We recorded the number of inter- and intra-cluster events per second and averaged those over ten generation processes. Recorded values are binned per separate graph configuration and presented in Table 2 and Figure 2.

	t_{0-1}	t_{1-3}	t_{3-5}	t_{5-7}	t_{7-9}	t_{9-10}
Intra-cluster events one	133	67	139	137	134	133
Intra-cluster events two	138	200	197	161	75	164
Inter-cluster events	13	12	16	15	11	12

Table 2. Evolution of event frequency for dynamic graph, values represent average count of events per second, time-intervals are based on different graph configurations

The influence of the script can be seen in the changes of events. Both clusters start with nearly the same number of events per second. The intra-cluster communication of cluster one is reduced in time frame two, because the number of nodes changed from 20 to 15. Simultaneously an increase for intra-cluster communication of cluster two was recorded. After reinserting five nodes to cluster one, the number of events went back to the initial value. No further significant changes of cluster ones communication level were observed. In contrast records for cluster two include a decrease of events in time frame four, which is correlated with the decrease of nodes. In the following time frames the communication level was first set to 50% and in time frame six set back to 100%, which is observable in the number of events of cluster two, as well. The number of inter-cluster events

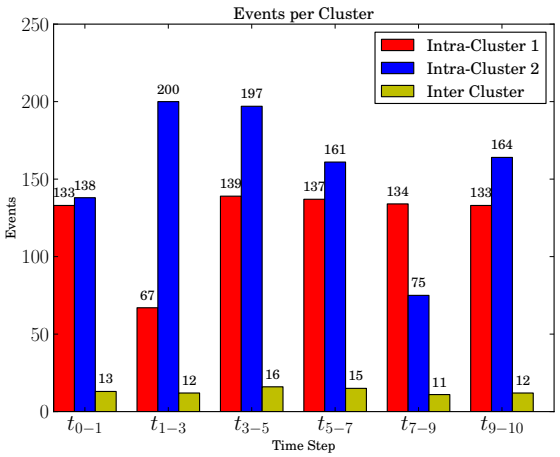

Fig. 2. Evolution of event frequency for dynamic graph, values represent average count of events per second, time-intervals are based on different graph configurations

was constant on a lower level than intra-cluster communication. This can be explained by the set base communication probability of $p_{base} = 0.01$. However, the number of expected events can be changed by adjusting this parameter.

5 Results and Outlook

We presented a generator for the creation of static and dynamic graphs. The generator in its current version was able to produce reasonable data as a base for multiple graph related problems. The comparison with a previous available spike train generator showed that spike trains with similar distributions of event-intervals can be created. Furthermore predefined ensembles could be detected in calculated distance matrices. Social network experiments demonstrated the ability to create graphs with small world properties and scale-free characteristics. All these processes can be combined with a script for planning changes in the graph structure. Therefore researchers will be able to test analysis techniques for dynamic graphs on event data containing predefined features.

Until the tool gets released we will concentrate on refurbishing the current generation process. Configurations with a high number of events per second (> 100) can still be too much afflicted with noise in the resulting distributions. Additionally further script functions will be added to insert more dynamic characteristics to the graph.

The implementation of our generator could be downloaded from `http://iws.cs.uni-magdeburg.de/~pheld/publications/IPMU2014/`.

References

1. Akoglu, L., Faloutsos, C.: Rtg: a recursive realistic graph generator using random typing. Data Mining and Knowledge Discovery 19(2), 194–209 (2009)
2. Barabási, A.L., Albert, R.: Emergence of scaling in random networks. science 286(5439), 509–512 (1999)
3. Chakrabarti, D., Faloutsos, C.: Graph mining: Laws, generators, and algorithms. ACM Comput. Surv. 38(1) (Jun 2006)
4. Clauset, A., Shalizi, C.R., Newman, M.E.: Power-law distributions in empirical data. SIAM review 51(4), 661–703 (2009)
5. Guare, J.: Six Degrees of Separation: A Play. Vintage Books, New York (1990)
6. Hamming, R.W.: Error detecting and error correcting codes. Bell System technical journal 29(2), 147–160 (1950)
7. Held, P., Kruse, R.: Analysis and visualization of dynamic clusterings. In: System Sciences (HICSS), 2013 46th Hawaii International Conference on. pp. 1385–1393. IEEE (2013)
8. Held, P., Moewes, C., Braune, C., Kruse, R., Sabel, B.: Advanced analysis of dynamic graphs in social and neural networks. In: Towards Advanced Data Analysis by Combining Soft Computing and Statistics, pp. 205–222. Springer Berlin/Heidelberg (2012)
9. Jaccard, P.: Étude comparative de la distribution florale dans une portion des alpes et des jura. Bulletin del la Société Vaudoise des Sciences Naturelles 37, 547–579 (1901)
10. Leskovec, J., Chakrabarti, D., Kleinberg, J., Faloutsos, C.: Realistic, mathematically tractable graph generation and evolution, using kronecker multiplication. In: Knowledge Discovery in Databases: PKDD 2005, pp. 133–145. Springer (2005)
11. Louis, S., Borgelt, C., Grn, S.: Generation and selection of surrogate methods for correlation analysis. In: Grn, S., Rotter, S. (eds.) Analysis of Parallel Spike Trains, Springer Series in Computational Neuroscience, vol. 7, pp. 359–382. Springer US (2010)
12. McGlohon, M., Akoglu, L., Faloutsos, C.: Weighted graphs and disconnected components: Patterns and a generator. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 524–532. KDD '08, ACM, New York, NY, USA (2008)
13. Nawrot, M., Aertsen, A., Rotter, S.: Single-trial estimation of neuronal firing rates: from single-neuron spike trains to population activity. Journal of neuroscience methods 94(1), 81–92 (1999)
14. Shalizi, C., Camperi, M., Klinkner, K.: Discovering functional communities in dynamical networks. In: Airoldi, E., Blei, D., Fienberg, S., Goldenberg, A., Xing, E., Zheng, A. (eds.) Statistical Network Analysis: Models, Issues, and New Directions, Lecture Notes in Computer Science, vol. 4503, pp. 140–157. Springer Berlin Heidelberg (2007)
15. Vreeswijk, C.: Stochastic models of spike trains. In: Grn, S., Rotter, S. (eds.) Analysis of Parallel Spike Trains, Springer Series in Computational Neuroscience, vol. 7, pp. 3–20. Springer US (2010)
16. Watts, D.J., Strogatz, S.H.: Collective dynamics of small-worldnetworks. nature 393(6684), 440–442 (1998)
17. Yule, G.U.: On the association of attributes in statistics: with illustrations from the material of the childhood society, &c. Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character 194, 257–319 (1900)