ADAGE: A software package for analyzing graph evolution

Mary McGlohon, Christos Faloutsos

May 2007
CMU-ML-07-112

Carnegie Mellon

ADAGE: A software package for analyzing graph evolution

Mary McGlohon, Christos Faloutsos

May 2007
CMU-ML-07-112

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

Understanding common properties in time-evolving graphs is useful for gaining important information about specific networks. Certain graph properties may provide a better understanding of how a network is evolving, and critical time points in its evolution. Furthermore, being able to detect anomalies in interactive networks is useful for detecting fraud, insider trading, and other illegal practices. This work introduces a tool for analyzing networks: ADAGE, a software package, plots and points out anomalies and "phase transitions" in graphs.

This material is based upon work supported by the National Science Foundation under Grants No. IIS-0209107, SENSOR-0329549, EF-0331657, IIS-0326322, IIS-0534205, and also by the Pennsylvania Infrastructure Technology Alliance (PITA). Additional funding was provided by a generous gift from Hewlett-Packard. Mary McGlohon was partially supported by a National Science Foundation Graduate Research Fellowship.

Keywords: networks, evolving graphs, graph properties, phase transition

1 Introduction

In the analysis of a time-evolving graph, there are several questions one might ask. In what fashion do new entities enter a network? Does the network retain certain graph properties as it grows and evolves? Does the graph undergo a "phase transition", in which its behavior suddenly changes? In answering these questions it is of interest to have a diagnostic tool for tracking graph properties and noting anomalies and graph characteristics of interest.

In this work we perform analysis on real evolving networks. In doing this we present the first version of a software tool called Anomaly Detection and Analysis for Graph Evolution (ADAGE) that tracks properties of interest, provides plots of how these properties evolve over time and with the size of the graph.

2 Software

2.1 Software objective

ADAGE is a MATLAB package that takes as input a collection of edges labeled with timestamps and outputs several plots illustrating the evolution of the graph. These plots may be used to better understand general behavior of evolving graphs and to detect anomalies and deviations from normal graph evolution.

2.2 Using ADAGE

2.2.1 Installation

ADAGE was built for use on MATLAB 2007 version, although it will likely work on previous versions. Since MATLAB is an interpretive language, no compilation is needed. Simply place all files in the ADAGE software folder into a directory, begin MATLAB, and set directory to the path containing the ADAGE software.

2.2.2 Demo

To run the ADAGE demo, from a bash shell run the following:
`make demo`
Alteratively, type
`demo`
in a MATLAB session, working from the appropriate directory.

This will run ADAGE on the NIPS coauthorship dataset. Plots will be saved as `nips-laws.ps`, with the component size plot as nips.fig. Description of the output, as well as examples, may be found in the next section.

2.2.3 New datasets

The `ADAGE()` function takes as input three arguments, `input`, `name`, and `isBipartite`. The bipartite feature has not yet been implemented, so this argument is always 0. name is a string (enclosed by ' ') that will be used as the filenames for the output. Finally, `input` may have one of two formats, a pre-loaded MATLAB matrix or a filename. A matrix should have one row for each edge, and three columns representing *node-in*, *node-out*, and *timestamp*. A filename should be a tab-separated text file with the same format. In this case `input` is a string.

If the MATLAB matrix has already been saved, simply run

```
> input = load('filename')
> ADAGE(input, name, 0)
```

The ADAGE() function saves plots as *name*`-laws.eps` and *name*`-components.eps`.

3 Experiments

3.1 Output

ADAGE tracks several variables. Among them are:

- Number of edges over time
- Number of nodes over time
- Densification law: Edges vs. Nodes
- Eigenvalues over increasing nodes
- Size of largest connected component vs. nodes
- Number of connected components vs. nodes and time
- Comparative sizes of connected components over time

All but the last item are saved in *name*`-laws.ps`, the last as *name*`.fig`.

Number of edges and nodes is monotonically increasing, as the current version does not allow decay of edges– such properties are very domain-dependent. However, the rate at which edges and nodes increase may vary by the dataset. The *densification law* states that number of edges vs. number of nodes should follow a power law [**?**]. The eigenvalues of a graph are of interest, as they might indicate a phase transition. The sizes and numbers of connected components are also indicative of phase transitions such as *gelling* or *differentiating*. One would expect the last item to follow power laws with the same slope in each timestep.

3.2 Experiments

We ran the toolkit on three real datasets. The first, as mentioned in the demo, is a coauthorship network over 13 years at the NIPS conference. Each node is an author, and an edge represents a coauthorship on a single paper. In the output, shown in Fig. 1, we note that edges and nodes both increase linearly (noting that the first two plots are on a log-lin scale). As expected, they obey the densification law. Also, one may note that in `nips.fig`, the slope of the counts of each size of connected component remain approximately the same over time.

Second, we used a dataset gathered from $44,000$ blogs over a three-month period. Edges are hyperlinks between blog posts, and each node is a blog. Some output from ADAGE on this set may be found in Fig. 2. One notes that around $t = 30$, edges and nodes begin to plateau. We call this a "burning off period", which we attribute as an effect of the data gathering procedure. We also note that at about the same time, components begin to merge together as the number of connected components decreases.

Thirdly, we used a citation dataset from Arxiv. This set contains $27,770$ papers and $352,000$ citations. Some results may be found in Fig. 3. There is a long period of inactivity in the first eight timesteps, after which activity resumes as similar in the blog dataset: notice that in both instances, the size of largest connected component increases linearly (power law with slope near 1). It is possible that there is some formula for the size of the greatest connected component- such as $0.99N$ or $N-k$, where k is some constant number of nodes remaining separated from the largest connected component.

4 Applications

ADAGE might be useful for a number of applications. Several domains have evolving networks, where analyzing such a network may be beneficial. For instance, spotting anomalies may be useful for finding terrorist cells in a social network, as in [1]. It may also be useful in fraud detection in online trading domains, as in [2].

Applications of tracking graph anomalies has further applications than security, however. Often graphs go through "phase transitions", where . For instance, a new graph may undergo a "burning off period", as in the blog network. Or, a graph may reach a "capacity" for separated comonents, and adding edges simply makes the components clump together, or "gel". It may be useful to recognize when a graph properties change in order to point out these phase transitions.

5 Conclusion

5.1 Future work

Future versions of ADAGE will include implementation for bipartite graphs. Once further work has been done in analysis, ADAGE will also feature automatic detection of graph anomalies.

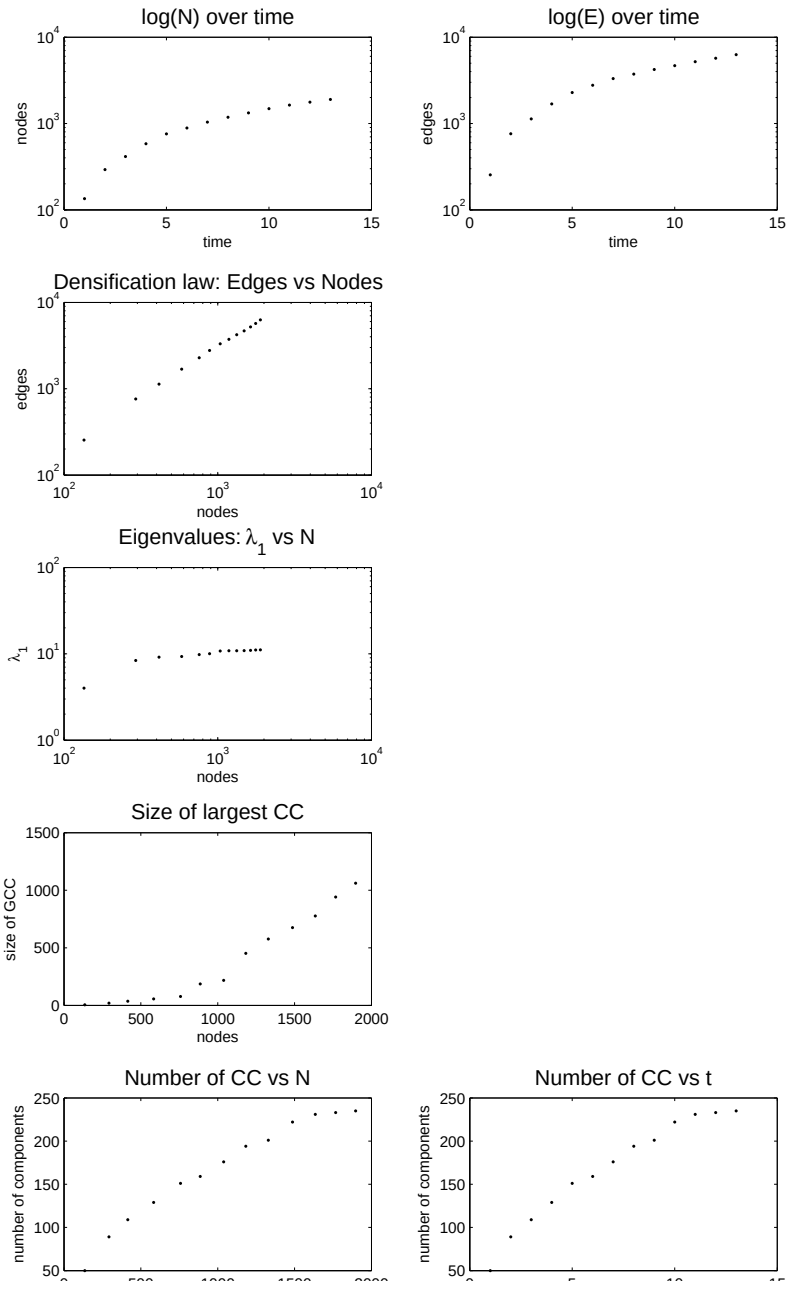

Figure 1: Output for NIPS dataset. Note how edges and nodes increase linearly with time, while the network becomes more dense over time. Number of connected components begins to level off.

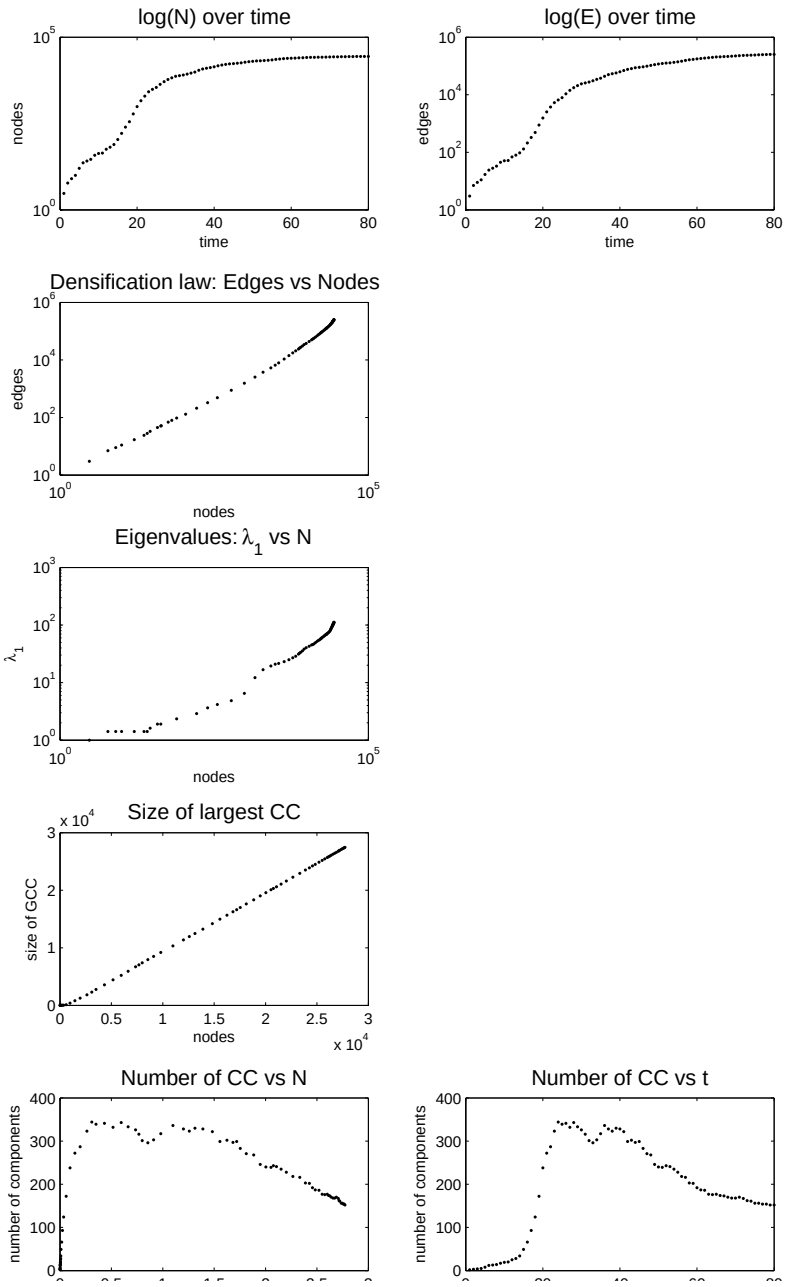

Figure 2: Output for blog dataset. There appears to be a "burning off period" in the first 30 days, which is reflected in nodes, edges, eigenvalues, and the number of connected components.

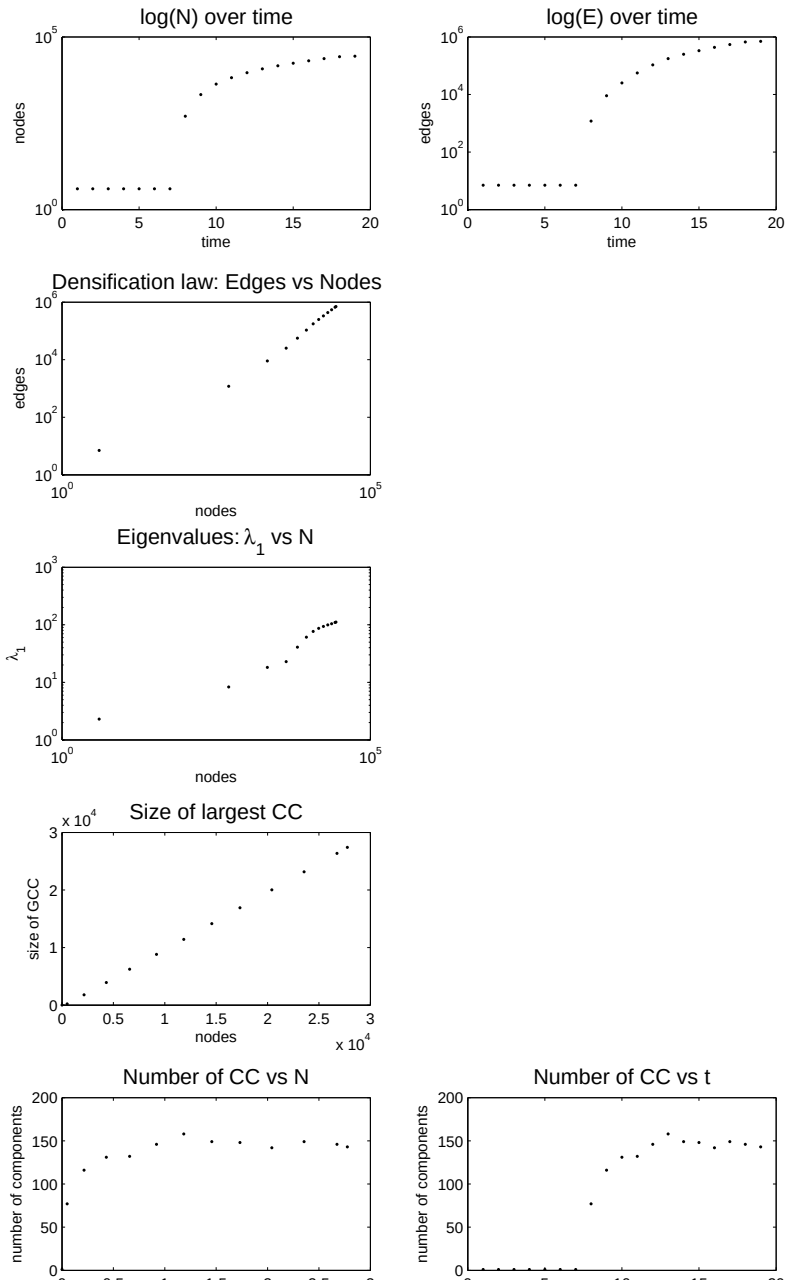

Figure 3: Output for Arxiv dataset. The surge of activity at $t = 7$ is reflected in all the output graphs, and the largest connected component grows linearly with N, with slope near 1. So, in each time step, a certain percentage of new nodes join the large component.

5.2 Summary

We presented a software package ADAGE that tracks and plots several useful properties in evolving graphs. This software may be used to analyze graphs, spot evolving properties of graphs in general, and find "phase transitions". We provided a tutorial for using the software, as well as some example output and interpretation.

References

[1] J. Neville, Özgür Şimşek, D. Jensen, J. Komoroske, K. Palmer, and H. Goldberg. Using relational knowledge discovery to prevent securities fraud. In *KDD '05: Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 449–458, New York, NY, USA, 2005. ACM Press.

[2] S. Pandit, D. H. Chau, S. Wang, and C. Faloutsos. Netprobe: a fast and scalable system for fraud detection in online auction networks. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 201–210, New York, NY, USA, 2007. ACM Press.

Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213

Carnegie Mellon