T-GRAPH: Spotting Anomalies in Time-Evolving Graphs

Soumya Wadhwa
SCS CMU
soumyaw@cs.cmu.edu

Dhivya Eswaran
SCS/CMU
deswaran@andrew.cmu.edu

Christos Faloutsos
SCS CMU
christos@cs.cmu.edu

December 4, 2017

Abstract

How can you find strange nodes in a who-calls-whom graph? Spotting anomalies in graphs is an important topic. It is especially challenging when the graph is changing over time. Our method has the following properties: (a) it is scalable, being nearly linear on the input size (b) it is effective, spotting over 90% of the anomalies in synthetic data and giving observably good qualitative results on real data (c) it can detect anomalous microclusters along with isolated outlying points for static graphs (d) it can detect global discontinuities w.r.t. graph measures (d) it can detect anomalies at a much smaller scale compared to the graph size (e) it can operate over different time granularities Experiments on real data from the Enron email dataset illustrate the benefits of our method.

1 Introduction

Online reviews for products, movies and restaurants are important. So are followers on a social network like Twitter. They are often faked, for monetary gain. How can we spot this fraud?

Our project is based on the detection of discontinuities at the global and local levels in time-evolving graphs for a wide variety of domains - from bank transactions to online product reviews. Anomalous activity is prevalent in most industries today including the financial sector, insurance, the retail sector (online and offline), and government agencies. Attempts at fraud have seen a mammoth rise in recent years, making its detection very important in todays context. Since there are relatively few anomalous users or transactions in a large population, finding these can be quite challenging. Also, since there are so many different types and methods of committing fraud or attacking a network, identifying them becomes more difficult. Tackling this problem becomes especially hard when the network (social, communication, transaction) is changing over time.

Given any data which can be expressed as triplets of the form (source, destination, timestamp), for example (user, product, timestamp) for product reviews and (from, to, timestamp) for financial transactions and communication (email or phone) networks, the challenge is to detect which of the given triples represent anomalies or discontinuities with respect to the general trend, and subsequently identify the anomalous nodes. For example, identifying fake users on Twitter, or spammers in a ‘who called whom’ network. Specifically, we aim to focus on detecting events in the Enron email dataset [2] currently.

In the previous semester, we had worked on improving the existing graph feature visualization tool, Perseus [1], by efficiently incorporating the temporal aspects of data along with its structural graph attributes. We also explored a few approaches for the identification of interesting (suspicious) micro-clusters in the plots of structural properties for static graphs (at specific timestamps).

This semester, the project will be focused on analysing the evolution of various graph properties at different timestamps. More specifically, we are working on a hierarchical approach to detect anomalies in time-evolving data, wherein we first identify an anomalous timestamp by analysing global graph properties over time, with different granularities, and then further attribute the anomalous behavior to a certain subset of nodes who are misbehaving in that timestamp. We also want to be able to detect local discontinuities (small anomalous subgraphs) which cannot directly be detected at the global level. Our work this semester will be aimed at tackling as many of these facets of anomaly detection in time-evolving graphs as is possible. The outline of the paper is typical: we give the survey (Section 2), the proposed method (Section 3), experiments (Section 4), and conclusions (Section 5).

2 Background and Related Work

Some of the recent work in fraud detection has focused on modeling the adversarial setting (when the fraudsters know your strategy for detecting fraud). FRAUDAR [5] is one such algorithm that is camouflage-resistant (based on the density metric used f(S)/—S—), provides upper bounds on the effectiveness of fraudsters, is effective on real-world data, and is scalable. Their approach was to start with the entire set of nodes, and then remove nodes successively, one at a time. At each step, a node is selected to be discarded so as to leave behind the set of nodes with the highest value of the density metric. Other approaches aim to incorporate the different dimensions of the data while detecting the dense regions corresponding to fraudsters. For example, M-Zoom [6] efficiently detects dense blocks in multi-dimensional tensors which are constructed from the data features. Other pattern-based techniques are also used for identifying fraud. Authors of EigenSpokes [7] observed spoke-like patterns while mapping the singular vectors obtained using Singular Value Decomposition in an EE graph, and interpret these patterns to identify a seed point to start clustering fraudulent communities. Belief propagation and related algorithms have also been used for fraud detection. For example, FraudEagle [8] uses belief propagation in signed networks over time to detect fraud in online reviews. The work we did last semester extended Perseus [1], which is an interactive user and system guided tool for large-scale graph mining, based on Pegasus [9] for fast and efficient calculation of graph measures. For automatic outlier detection, Perseus uses the Local Outlier Factor (LOF) [10] metric which is able to detect outliers in local neighborhoods instead of using a global clustering mechanism, and is thus more effective. Fast anomaly detection despite duplicates [11] is used to tackle the duplicate data points problem in the LOF method. Perseus, however, does not tackle the time-component of the data which can be a very good indicator of fraud when there is a deviation from the average trends. The authors of RSC [12], analyze time-stamp data from social media services (such as Twitter) and find characteristic patterns in the distribution of postings inter-arrival times (IAT). Based on their findings, they proposed a generative model to match these patterns, Rest-Sleep-and-Comment (RSC), for anomaly detection. We worked on incorporating IAT and related node-level temporal properties into Perseus last semester.

Recent work on anomaly detection in time-evolving graphs involves the analysis of delta statistics as in NetSimile [13], or assessment of the similarity between two graphs on the same nodes as in DeltaCon [14]. TimeCrunch [15] takes an incremental approach to dynamic graph summarization, and encodes a model and the associated error which can be used to losslessly reconstruct the original dynamic graph using Minimum Description Length. In [16], the authors introduce the concept of density-consistent network statistics so that these reflect the state of the network independent of the number of edges, and are able to detect network events which were previously hidden. We might use this notion while constructing the set of graph measures to be used for the detection of global discontinuities. Our work this semester is closely related to Metric Forensics [17], which combines a multi-level approach (graphs, communities and nodes), a collection of graph metrics, and various analysis techniques for anomaly detection. At each successive level, the graph is viewed at finer temporal resolutions and more computationally expensive metrics are computed, making the approach very scalable. We will extend this by adding more features (such as k-core statistics as in CoreScope [18], or reconstruction cost for degrees based on SVD as in fBox [19]), and we also aim to identify local temporal discontinuities which arent detected by this approach, such as the appearance of a near clique of hundred nodes in a million node graph.

3 Proposed Method

In this section we present the proposed method, we analyze it and provide the reader with several interesting -at least in our opinion- observations.

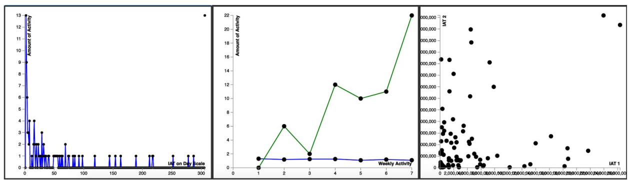

Figure 1: Temporal User Activity in Perseus

3.1 Temporal properties in Perseus

(Soumya and Tejas) Inter-Arrival Time (IAT) refers to the time between consequent activities recorded for a particular user. We incorporated the following plots (in addition to the egonet) which can be analyzed for each user who seems suspicious.

- *Weekly Activity:* We plotted this for each user along with the average user activity on different days of the week to check for spurious patterns.
- *IAT vs Activity:* The typical trend observed in this plot is random, but if a spike is observed for any particular IAT value, specifically for low IATs, the user can be a bot.
- IAT_n *vs* IAT_{n+1}: A near-linear trend in this plot indicates suspicious activity.

3.2 Algorithms for Micro-Clustering

After analysing various plots for many datasets, we observed visual patterns which these plots follow. We intended to find microclusters in them and summarize the patterns for the end user. For the sake of simplicity, we will only focus on one of the degree vs. PageRank plots in this report, although these methods can be applied to any plots following certain trends. For example, the straight line pattern which exists in the graph (Figure 2), highlighted with a grey circle, is not generic behaviour and rather indicates

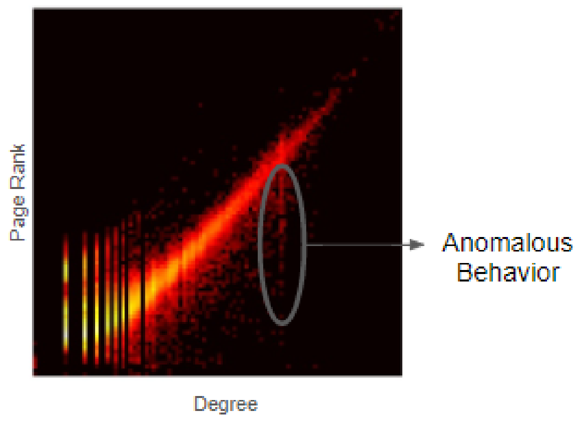

Figure 2: PageRank vs Degree plot from Perseus

suspicious activity. We tried the following methods for clustering:

- *Skeletonization:* **(Saksham)** Initially, we experimented using image morphology in which we tried image skeletonization on the plot as a means to explain it. The initial hypothesis for this experimentation was that this would enable us to find a general shape (say T-shape) which depicts the outlier clusters, but the fine level of skeletonization was too restrictive for us to find any general pattern of this sort. After careful analysis of skeletonization results, we observed that the plot under consideration can be summarized as a huge chunk of points which follows a power law distribution, and the initial straight lines which represent reasonable behaviour. However, binarization led to loss of information, and the shapes obtained were very sensitive to the marker size used for plotting.

- *Bump Hunting:* **(Soumya)** This is a statistical method for identifying various non-overlapping blocks (regions of interest) in the data plots by optimizing the values of a target density metric, removing identified contiguous blocks with optimal values in each stage. Examining the graph at various stages in the trajectory of the peeling process can lead to interesting insights and possible identification of the second densest cluster.

- *Watershed Segmentation:* **(Soumya)** This is a standard technique used in image processing for segmentation. Image segmentation is the separation of objects in the image from their background, and also from each other. The watershed segmentation algorithm operates upon the image of the heatmap like a topographic map, where the brightness of each point (grayscale pixel value) represents its height, and aims to find the lines that run along the tops of ridges. The intuition behind using this method was that it would enable us to detect contours of the different microclusters being formed in the heatmap of the plot, while being very scalable since it works in the image space and not the data space. We observed that as the grid size is increased, more meaningful clusters are formed, and as the initial flooding level was increased, the number of clusters formed was less. However, the clusters formed were somewhat non-intuitive and required parameter tuning to make some sense of them.

- *Hierarchical DBSCAN:* **(Saksham)** HDBSCAN is a hierarchical density based clustering approach as opposed to the generic flat clustering. Since this is scalable, it seemed to be something which could help us obtain the inherent patterns that we feel are necessary for summarization and depicting suspicious behaviour. The generic techniques like K-Means, G-Means or Affinity propagation limits the distribution of the clusters to a Gaussian and ends up finding circular clusters. Another

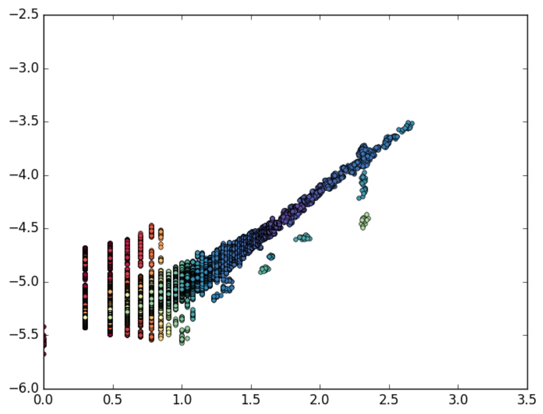

Figure 3: HDBSCAN with min cluster size 10

problem with this approach is that, it tends to even classify all the noise also as a part of some cluster. The closest clustering approach which gets closest to the natural human perceived clusters is DBSCAN. It works fine but tries to combine nearby clusters based on the presence of noisy points which act as bridging points between them. We found HDBSCAN performs the best and is able to alleviate the challenges observed in DBSCAN. The purple dots have been classified as a noise cluster due to sparsity in points compared to other clusters. The other identified clusters can be more visible if we remove the noise cluster. We see that is able to identify those weird microclusters which we capture human attention.

- *Local Maxima Clustering:* **(Tejas)** This is a technique which helps in estimating an initial configuration of all the data points by assigning a cluster label. The motivation behind using this technique was that the shape and position of the cluster can be estimated based on the distribution of the data. Even though HDBSCAN worked on the synthetic data, it failed to produce satisfactory clusters for the real data due to extreme density in particular places of the plot. It is possible to overcome the problem by finding representative points based on the local density and then cluster the points. These representatives can be the Primary, Secondary and Tertiary local maximas which are the highest, second highest and third highest among their neighbours respectively. These points are computed on the binned heatmap of the entire data. Further they are clustered using DBSCAN to find the possible shape and position of the final clusters. This method can be used for initializing a cluster configuration.

3.3 Time-Evolving Graphs

Graphs can be of different types:

- Simple and Multi
- Directed and Undirected
- Weighted and Unweighted
- Unipartite and Bipartite

Given a sequence of directed weighted graphs, our aim is 2-fold:

- *Global discontinuity detection*
  - Find anomalous time stamps indicating occurrence of events or change points in the time series
  - Attribute the anomalousness of a graph instance corresponding to a particular deviant timestamp to some nodes, edges, or subgraphs
- *Local discontinuity detection:* Detect changes in local patterns which are too small to be detected at the global level.

3.3.1 Global Discontinuities

Our general approach for this hierarchical anomaly detection in time-evolving graphs is:

- *Step 1:* Derive global features (such as number of connected components and degeneracy) from each graph.
- *Step 2:* Pass the obtained vectors through Robust Random Cut Forests (preferably) or a standard outlier detection method such as Isolation Forests to detect anomalous timestamps.
- *Step 3:* Derive local features from each node at or around (+ or - some small time) anomalous timestamps and "explain" the anomalies in terms of change in behavior of a certain set of nodes in the graph [Future Work]

Global Graph Features

- Number of nodes
- Number of edges
- Minimum, Maximum and Average degree
- Percentiles of degree distribution (median etc.)
- Giant connected component size and number of connected components
- Percentiles of sizes of connected components
- Core number (distribution) / Degeneracy
- Jaccard Similarity of Node and Edge Set (with previous timestamp) : First order
- Reconstruction cost (distribution)
- First k eigenvalues of adjacency and Laplacian matrices

Node-Level Features [Future Work]

- Degree
- Total weight (weighted)
- Core number
- Number of triangles (approximate)
- HITS / SALSA
- PageRank
- Centrality - Eigenvalue
- Number of distinct destinations (multi-edge)
- One-step-away: aggregate of each of the above, for the 1-step-away neighbors

3.3.2 Local Discontinuities

- XOR + SVD
- diff + SVD
- plain SVD
- weight-decayed XOR + SVD

3.3.3 Resolution

- Hourly
- Daily
- Weekly
- Monthly
- Yearly

3.4 The Road Not Taken

Now, at the local level, we tried to characterize different types of anomalies which could occur:

- Appearance of bridge edges
- Appearance of a large clique / core
- Appearance of a large star
- Unusual change in degrees

We observed that most anomalies can be detected using degrees of nodes, except the appearance of bridge edges which can be detected using pairwise commute time, changes in hubness scores or number of connected components. Thus, currently we are focusing on detecting unusual changes in degrees of a subset of nodes in an anomalous timestamp. The baselines that we have decided to use are:

- Using statistics from degree vector at each timestamp (total, average, number of non-zeros, mean, median, variance, quantile etc.) to find significant deviations
- Using distributional KL divergence (time-decayed average from current) as a node outlier score.
- Using the entire degree vector at each timestamp along with some standard outlier detection algorithm (such as IF).

We also plan to construct synthetic examples of cases where each baseline would fail and gather time and memory statistics for each baseline, so that we can beat these!

The main research question that we ultimately want to tackle is: Can we do better in smaller space and lesser time by keeping track of a fewer number of locality-sensitive statistics? More precisely, we plan to come up with a technique of partitioning the graph nodes into subsets such that meaningful results can be obtained by analyzing the properties of each of these subsets rather than having to process each node separately.

4 Experiments

Here we report experiments to answer the following questions

- Can our method detect global discontinuities?
- Can our method detect local discontinuities?
- Is our method scalable?
- Is our method effective?

4.1 Global Discontinuity Detection

4.1.1 Real Data: Enron Emails

Isolation Forest assigns an outlier score to each timestamp which is represented as a vector of its global graph properties. The more negative the score, the more anomalous that snapshot.

We discover that the top detected anomalous timestamps are:

- 21st - 22nd November 2001: Nov. 19, 2001 – Enron restates its third quarter earnings and discloses it is trying to restructure a 690 million obligation that could come due Nov. 27.
- 7th - 8th February 2002: Feb. 7, 2002 – Fastow and his former top aide Michael Kopper invoke the Fifth Amendment before Congress; Skilling testifies, saying he knew of no problems at Enron when he resigned.
- October 2001 (Several Dates): Many organizational changes + SEC inquiry and investigation

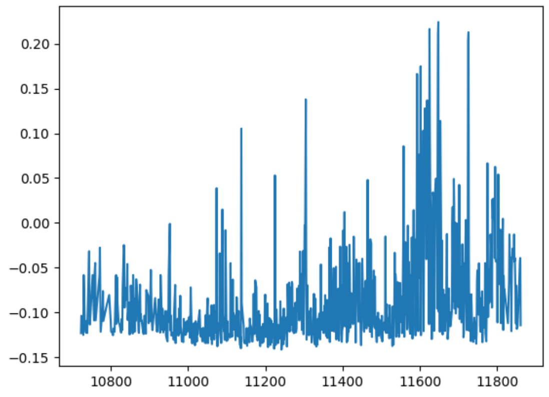
Figure 4: Negative anomaly score versus time

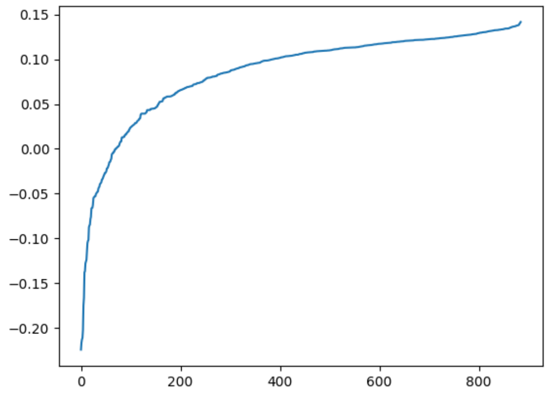
Figure 5: Negative anomaly score versus rank

4.2 Local Discontinuity Detection

4.2.1 Synthetic Data: Kronecker Graphs

Random Kronecker Graphs with noise The first set of experiments that we conducted were on synthetic data, which was basically a set of stochastic Kronecker graphs, with different edges being drawn from the same probability distribution (fixed parameters) at different timesteps. Each graph contained almost 18000 edges. Cliques of increasing sizes (25, 50, ... 225, 250) all multiples of 25 were inserted at times 7, 15, ... 71, 79 at a regular interval of 8.

We analysed the top-4 eigenvalues for these series of time-evolving graphs, with varying noise parameter, and obtained figures 6 to 8. Observe that the general trend is that, while the principle eigenvalue shoots up (depending on the size of the injected clique), the second eigenvalue also becomes abnormally high to replace the first.

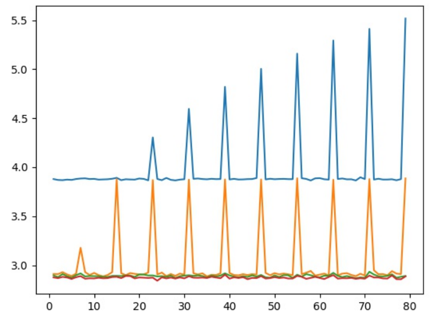
Figure 6: Log of the first 4 eigenvalues of each snapshot vs time with noise level 0.0

We capitalized on this fact to devise a detection algorithm which identifies a timestamp as anomalous if the second eigenvalue is greater than its value in its previous timestamp above a certain threshold. I experimented with different values of this threshold, and relative thresholds w.r.t. the expected value of

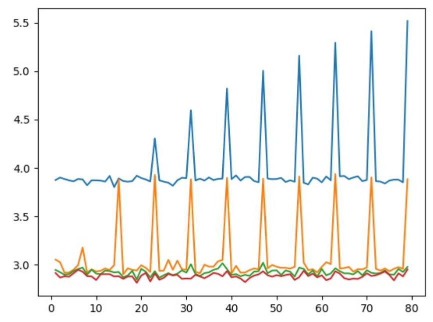

Figure 7: Log of the first 4 eigenvalues of each snapshot vs time with noise level 0.5

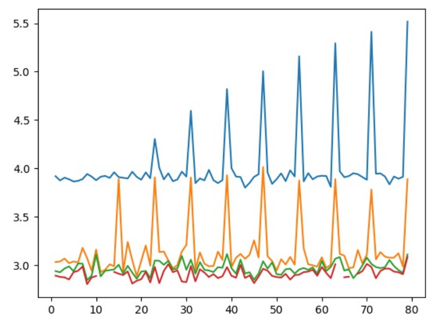

Figure 8: Log of the first 4 eigenvalues of each snapshot vs time with noise level 1.0

the principle eigenvalue seemed to work the best! Results obtained for relative threshold 0.08 were:

- Noise level 0: 100% precision, 100% recall
- Noise level 0.001: 100% precision, 100% recall
- Noise level 0.01: 100% precision, 100% recall
- Noise level 0.1: 100% precision, 100% recall
- Noise level 0.5: 100% precision, 90% recall
- Noise level 0.8: 100% precision, 90% recall
- Noise level 1.0: 90% precision, 90% recall

The threshold helps balance between precision and recall. A higher threshold favors more precision, and a lower one favors more recall, as expected.

4.2.2 Real Data: Enron Emails

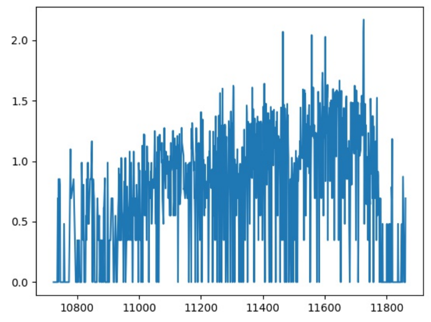

Figure 9: Log of the first eigenvalue of current snapshot vs time

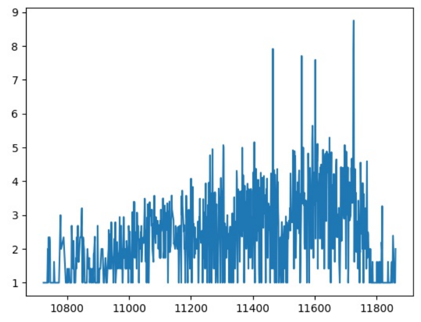

Figure 10: First eigenvalue of current snapshot vs time

Timestamps with the highest principle eigenvalues:

- 7th - 8th February 2002: Feb. 7, 2002 – Fastow and his former top aide Michael Kopper invoke the Fifth Amendment before Congress; Skilling testifies, saying he knew of no problems at Enron when he resigned.
- 23rd May 2001: May 17, 2001 – "Secret" meeting at Peninsula Hotel in LA – Schwarzenegger, Lay, Milken.

Q1. **Scalability**: How fast is our T-GRAPH
Q2. **Effectiveness**: How well does T-GRAPH work on real data?
Q3. **Accuracy**: Can it detect smaller-sized discontinuities (missed by the global methods)?

The graphs we used in our experiments are described in the table 2.

Nodes	Edges	Description
Real-world Networks		
150	500000	Enron Email

Table 1: Summary of real-world networks used.

4.3 Q1 - Scalability

Nodes	Edges	Timesteps	Time Taken
125	290	80	1.5s
625	2300	80	13s
3125	17500	80	68s

Table 2: Summary of real-world networks used.

4.4 Q2 - Effectiveness

In previous sections, we show the precision/recall of T-GRAPH for synthetic data. Notice that [the results are great]

5 Conclusions

We presented T-GRAPH, which addresses the problem of anomaly detection in time-evolving graphs.

The advantages of the method are:

1. **Scalability**: it scales nearly linearly with the input size, as shown in Section since we have carefully picked features (linear / log-linear) in our global discontinuity detection mechanism, and the local discontinuity detection method just involves singular value decomposition.
2. **Effectiveness**: it gives excellent precision and recall (¿ 90%), on synthetic as well as real world data
3. **Accuracy**: It can detect smaller-sized discontinuities (missed by the global methods)
4. it can detect anomalous microclusters along with isolated outlying points for static graphs
5. it can detect global discontinuities w.r.t. graph measures
6. it can operate over different time granularities

We also presented experiments on *3GB* of real data, where T-GRAPH outperformed the competitors by 10 percentage points of accuracy, and 3x faster execution time.

[nice to have:] **Reproducibility:** We have already open-sourced our code, at `www.cs.cmu.edu`

References

[

A Stealth appendix

[here we put self notes etc]

A.1 Discussion - practitioner's guide

Given the above, we recommend that practitioners choose [bla] for the parameter values of T-GRAPH, and [do preprocessing using bla-bla] for their datasets.

A.2 Discoveries - T-GRAPH at work

Thanks to our method, we processed real data and noticed the following observations [elaborate... - if too many nice observations, promote this to a full section]]