Random Reference Models and Network Rewiring in Temporal Network Clustering

Patrik Seiron
Abstract

**Random Reference Models and Network Rewiring in Temporal Network Clustering**

*Patrik Seiron*

Computing on temporal networks is difficult because of their dynamic nature. One way to solve this is to slice them into multilayer networks, but this results in a loss of information. This thesis tries to find out at which number of slices this loss of information is at a minimum by using random reference models, algorithms that randomize a specific part of the network, and community detection to extract the impact of the slicing. This is done by calculating modularity, how strongly connected the communities are, before and after randomization. For three of the four datasets that were tested a maximum was found where a larger part of the network’s community structure was destroyed and thus a smaller part connected to the conversion from a temporal network to a multilayer network. The method tested could be used for some networks to find when the loss of information is at its lowest, but further experiments are required to prove to which networks the technique can be applied.
Contents

1 Introduction ......................................................... 4
  1.1 Temporal networks ............................................. 4
  1.2 Communities .................................................... 4
  1.3 Randomized reference models ................................. 5
  1.4 Thesis purpose ................................................ 5
    1.4.1 Thesis questions .......................................... 5

2 Temporal networks and randomization .............................. 5
  2.1 Temporal networks ............................................. 5
  2.2 Temporal network structures ................................ 7
    2.2.1 Topological structure .................................... 7
    2.2.2 Burstiness .................................................. 8
    2.2.3 Event sequence ............................................ 8
  2.3 Randomized reference models ................................ 9
    2.3.1 Link altering methods ................................... 9
    2.3.2 Time stamp altering methods ........................... 10

3 Community and community detection .............................. 12
  3.1 Communities .................................................. 12
    3.1.1 Communities in temporal networks ...................... 12
  3.2 Generalized Louvain ......................................... 13
  3.3 Normalized mutual information .............................. 14
  3.4 Modularity ..................................................... 14

4 Datasets .................................................................. 14
  4.1 Hypertext ......................................................... 14
  4.2 Infectious: stay away ......................................... 14
  4.3 Haggle ............................................................ 15
  4.4 Primary school ................................................. 15

5 Implementation ...................................................... 15
  5.1 Time stamp shuffling .......................................... 15
  5.2 Link shuffling .................................................... 15
  5.3 Experiments ..................................................... 16

6 Results ................................................................ 18
  6.1 Impact of random reference models on community structure... 18
  6.2 Change in community structure for different number of slices ... 19
  6.3 Conclusions ....................................................... 20
  6.4 Future Work ..................................................... 20
1 Introduction

As the human world grows larger and more complex the needs and fields of application for methods for understanding and finding correlations in large connected systems have increased. These systems are frequently modeled as networks [1]. One of the areas where the interest have exploded is human interaction both online, especially with increasing importance of social media in today society, and offline.

An example of offline human interaction could be a network modeled after a city where every citizen is a vertex and the edges would be that people that know each other or have regular interactions, this could be everything from friends and family to coworkers to even the employees at the local supermarket. In this network it would be feasible to find a connection between every person in the city, possibly an obscure connection but a connection.

The most basic way to model a system as a network would be a static graph, that is where the only information is vertices and the edges between them. There is a multitude of ways to extend this model by introducing new information.

1.1 Temporal networks

One of the possible extensions for the static network model is temporal networks, that is a model where information about when the edges occur is included. This creates the limitation that the vertices in the network can only communicate within specified time intervals. The intervals do not affect the entire network but are instead specified for every edge and create an opportunity for two vertices to communicate either directly or indirectly. This is known as a contact [1].

In the city example above this would limit the edges to only exist during time intervals when people are close to each other. This implies that two people can only interact if they are at the same place at the same time, which is for example useful when trying to create simulations of information or disease spreading [2].

A diverse number of systems can be modeled as temporal networks. The two most common are human communication networks, for example social media and phone- or email communication, and human proximity networks, which the city example is a part of. Temporal networks have also found uses in a variety of research fields such as economics and neuroscience [1].

1.2 Communities

Temporal networks are often created from huge datasets, hundreds upon hundreds of vertices and enormous amounts of contacts, of unsorted data. To be able to handle that amount of information, community detection methods can be used to partition the data into communities. A community is a set of vertices in the network that have a stronger connection with each other than they do the rest of the network [6]. In the city example a community would be a group of people that often interact with each other e.g. a group of friends or coworkers.

But there are issues with running community detection on a temporal network, which is a dynamic network that keeps changing and whose groups of vertices can be strongly connected at one moment and not connected at all in the next. One way to tackle this problem is called slicing, which turns the network into a number of static graphs with time intervals stacked on top of each
other, called a multilayer network. But this method is not perfect, converting the network causes a loss of information.

1.3 Randomized reference models

One way to analyze and better understand a temporal network is to use randomized reference models (RRMs) to destroy a specific part, a structure, of the network and observe the change. RRMs is a group of methods that go over a network and randomize a predetermined variable by either swapping between contacts picked at random or randomizing a new value. Examples of RRMs are link shuffling which swaps vertices between contacts and time stamp randomization which randomizes a new time stamp within an interval $[1,5]$.

1.4 Thesis purpose

This thesis will investigate if observing the change in community structure when the network is randomized by applying random reference models can be used to determine an optimal number of slices for a specific dataset. The idea behind this is that a network’s community structure after it has been sliced depends on two things: the network itself and the number of slices. Randomizing the network will remove the part based on the network leaving only the part based on the number of slices. When this part is at a minimum is the impact of the choice of slices on the network’s community structure also at a minimum.

1.4.1 Thesis questions

The thesis will try to answer the following questions.

- What happens with the network’s community structure when randomized with different RRMs?
- Can the destruction of community structure be used to find an optimal number of slices?

2 Temporal networks and randomization

This section will be a more in depth overhaul of temporal networks and randomized reference methods. It will cover: what they are, how they works, how temporal networks can be depicted and a few existing randomized reference models.

2.1 Temporal networks

A temporal network is a network where connections between vertices are not always active but are instead limited by time. Every edge has a time annotation that can either be an interval $T = [t_x, t_y]$ or a specific time $T = t_x$. In this report the time annotations will be called time stamps. It is only during these time stamps that communication between vertices will occur, either directly or indirectly. This interval of communication is called a contact $\llbracket \rrbracket$.

An edge in a temporal network consists of two vertices and a time stamp $([Vertex_1, Vertex_2, Timestamp])$, this is important to keep in mind because
even if in Figure 1 an edge has two time stamps this is in fact a compact representation of two separate contacts with their own time stamps. This is noteworthy because it is possible to change the vertices of one contact, without affecting the others.

The model used in Figure 1 is not the only way to depict a temporal network. Figure 2 shows another way to model a temporal network, a space-time graph. In this model every vertex is a horizontal line and every contact is a vertical line connecting the horizontal lines [1]. This model can be used when there is a focus on the temporal aspect because it is often easier to extract what happens at a specific time stamp but harder to get an understanding of the layout.

To make it possible to perform a number of operations including community detection on a temporal network a method called slicing is used. Slicing will turn the network into a number of static graphs, each graph is an accumulation of all the contacts that occur within a given number of timestamps, see Figure 3. This process will turn the temporal network into a multilayer network at the cost of information.

Figure 1: A temporal network depicted as an aggregated graph. Each edge has time stamps for when a contact happens.

Figure 2: An alternative way to depict temporal networks. It is the same network as in Figure 1. Each horizontal line is a vertex and the vertical lines between them are contacts.
2.2 Temporal network structures

This section will go over a few concepts in temporal networks that describe how the network is built. These concepts or so called structures are important to know to better understand how the use of RRM's adjust the networks. Each RRM will destroy at least one of the following structures to some degree.

2.2.1 Topological structure

The topological structure is not unique to temporal networks, it describes the space the network occupies. It is in layman’s terms the layout of the network. Destroying the topological structure implies that the layout been altered and if changed enough the end result is a completely different network, see Figure 4. By modifying the layout, how the vertices interconnect, will lead to different communities.

As an example, if the topological structure of a social network was destroyed could mean that a person has the same number of friends but some or all of the person’s previous friends have been replaced with other people. Because communities are a group of vertices that are strongly connected like a group of friends in a social network, by adjusting the friends the group will also change.
2.2.2 Burstiness

Now we look at the structures unique to temporal networks, one of these structures is called burstiness. The name is in a way self explanatory, burstiness means that the contacts between two vertices come in quick succession i.e. in bursts. [1]. The bursts can for example be two people sending numerous text messages between each other under a short time instead of sending messages sporadically or evenly distributed over time. Destroying burstiness can affect communities because when the contacts are more spread out over a time interval makes it harder for the community detection algorithm to determine how to divide the vertices and the end result can be a lot of very small or/and a few very large communities.

2.2.3 Event sequence

Event sequence is the idea of cause and effect in temporal networks i.e. that a contact causes another contact to happen [1]. This could be that person A calls
person B to tell that he/she won a million dollars, the cause, person B then calls person C to tell the news, the effect. If these contacts switches order it would not make sense, person B would not have a reason to call person C. Same as with burstiness, the destruction of event sequence can affect the communities. This can e.g, cause the algorithm to place vertices in different communities because the correlation between them is no longer there.

![Figure 6: A simple temporal network showing how the destruction of event sequence can affect it. On the left the event sequence is intact while on the right it is destroyed.](image)

2.3 Randomized reference models

Existing RRM\textsuperscript{s} can, to simplify it, be categorized into two different groups. The first group is methods that focus on altering the links between vertices, this might change temporal structures but is primarily used to destroy the topological structure. The second group are methods that will in one way or another change the time stamps on the edges, this will alter the temporal structure of the network.

2.3.1 Link altering methods

The first group are few methods that at the core do the same thing. The basic idea is to sequentially go over a number of edges, often the whole network. At each iteration one additional edge will be picked at random, either from the whole network or with some limitation. The vertices which the edges are between will then be swapped and since there are two possible ways to swap, this will be decided with a fifty-fifty chance \[1, 5\]. These methods can quickly destroy the topological structure and turn the network into a random graph, this can be seen in Figure\textsuperscript{8} where the implemented version of link shuffling has been run on a network.
Examples of link altering methods:

- Link shuffling: This is the basic method to alter the edges in the network. It works as described above and picks the second edge at random from the whole network \[1,5\].

- Slice link shuffling: This method works as the previous method but instead of picking an edge from the whole network it will only pick from edges in the same slice. This method is used when trying to preserve temporal properties \[5\].

Figure 8: A series of networks showing how link shuffling can transform a temporal network. The thickness of the line defines the number of edges between the vertices, each edge starts with five contacts in 1. After 1 a number of random edges have swapped vertices, the number is based on the size of the network. At 2 0.2 * size edges have been swapped, at 3 1 * size and 4 10 * size.

2.3.2 Time stamp altering methods

The second group are all methods that in one way or another modify the time stamp of the edges. The methods are design to preserve/destroy different temporal structures so it is possible to see the effect the different structures have.
These methods often work by swapping the time stamps between edges or randomizing them with limitations.

Examples of time stamp altering methods:

- **Time stamp shuffle**: This method will iterate over all edges and at each iteration pick another edge at random and swap the time stamp between them. It will destroy temporal structures like burstiness and event sequence, an example of how it affects a network can be see in Figure 9 [1,5].

- **Time stamp randomization**: This RRM will iterate over all edges and at each iteration randomize a time between the first and last time stamp in the network. It will not only destroy burstiness and event sequence like the previous method but also the day and night cycle, which is when the network is very active during certain times and not during others [1].

- **Sequence Shuffling**: This method does not work on the edges themselves but will instead randomize the order of the slices. This will preserve temporal structures within the slices [5].

- **Equal-weight link-sequence shuffle**: This method will find two pairs of vertices with an equal number of edges between them, it will then swap all the time stamps between the pairs of vertices effectively swapping the order the contacts happen [3].

Figure 9: Showing how time stamp shuffling can change a network. A) is the original network. B) show how the time stamps are swapped. C) is the network after randomization. Adapted from Holme, P. (2015, September 26). Modern temporal network theory: a colloquium. European Physical Journal B. Springer Berlin.
3 Community and community detection

This section will cover communities and how they function in temporal networks, the community detection algorithm used and a few central concepts.

3.1 Communities

The idea behind communities, also known as clusters, is to partition the network. This is done by finding a set of vertices, the community, that are more interconnected with each other than the rest of the network [6]. This thesis will focus on methods that find non-overlapping communities i.e. when a vertex can only be part of a single community compared to overlapping communities when it can part of multiple. Communities are often used to analyze unsorted data to look for properties of the vertices or relationships between them.

The first network in Figure 10 is an example of how communities can look. There are two groups of vertices with connections between them and a connection that links the groups. But it is not always that clear how the network will be divided into communities. In the second network there is a vertex that has an equal number of links to two different communities but can only be a part of one or the other, this can cause the algorithm to find different results between runs.

Figure 10: 1 is a static graph with clear communities. 2 have a vertex that can be part of either community.

3.1.1 Communities in temporal networks

Communities in temporal networks behave slightly differently because the network is dynamic. In temporal networks it is possible for a vertex to be part of different communities as time flows, see Figure 11. This is natural because a person can while at work be part of a community containing his/her colleagues but in the evening while at a concert be part of a community made of the attendees.
The most common approach to community detection in temporal networks is to calculate the communities for each slice while also taking neighboring slices into account, because the fact that a person does something for a short period of time might not be interesting depend on how detailed the results need to be. Other methods exist but will not be covered in this thesis.

Figure 11: Result after running the community detection algorithm on the temporal network shown in Figure 1. In the figure active contacts in each slice are shown as a line. Each slice has three communities but which vertex that is apart of which communities differ between the slices. The algorithm ran with a low omega value, mentioned in 3.2, to show that vertices can be part of different communities at different time stamps.

3.2 Generalized louvain

The algorithm used in this thesis is a C++ implementation of the generalized louvain algorithm [7], which original is created in Matlab. The algorithm is based on the louvain method but uses an adjacency matrix for each slice. Generalized louvain is a nondeterministic algorithm i.e. it can return different results after running multiple times on the same network, this can be seen in the results section (6) because after running the algorithm without changing anything it will still not find exactly the same communities. The algorithm does not work on temporal network but is instead implemented for multilayer networks, which means that the network needs to be sliced before the algorithm can be applied.

The algorithm takes the following five inputs:

- A multilayer network.
- Move: A parameter to decide if vertices should be placed in random communities or not. If set to "move" it will always place a vertex in the community which gives the highest increase to modularity. If move is set to "moverandw" the vertex will be placed in a random community with a chance that is proportional to the increase in modularity.
• Gamma: Is a resolution parameter, is always 1 in this report as recom-
mended by the creators [7].

• Omega: Is an inter-layer coupling weight parameter. This parameter de-
fines to what extent neighboring slices in the multilayer network affect
each other. The value can be between 0 and 1, at 0 each slices will be
considered a static graph and if omega is set to 1 each slice will affect the
other slices equally.

• Limit: Limits the number of modularity scores that can be kept in mem-
ory, a higher number will increase the algorithm’s speed because fewer
recalculations are needed but will consume more memory.

3.3 Normalized mutual information

Normalized mutual information (NMI) is used to compare two sets of commu-
nities, returns a value between 0 and 1. 1 is returned when the communities are
equal and 0 where they are completely different.

3.4 Modularity

Modularity is a measure that is used to determine how strongly connected the
communities in a network are. A set of communities with high modularity have
a higher number of edges within the communities compared to the rest of the
network and thus are better and more clear cut communities. Modularity is
often used to optimize community detection and gives a value between -1 and
1, a higher number is better.

4 Datasets

In this report four different datasets have been used, they are undirected tem-
poral networks i.e. all the edges have time stamps and communication is pos-
sible in either direction. The datasets have varying size and type. Hypertext, infectious: stay away and primary school was obtained from sociopa-
tterns [http://www.sociopatterns.org].

4.1 Hypertext

Hypertext is a human proximity network collected over 2.5 days at the ACM
Hypertext 2009 conference. The contacts are face-to-face contacts longer than
20 seconds gathered by radio badges voluntarily worn by attendees [8]. The
dataset contains 113 vertices and 20818 edges.

4.2 Infectious: stay away

This is a human proximity network collected from one day at the Infectious:
stay away exhibition held in 2009 at the Science Gallery in Dublin [8]. Every
vertex is an attendee and the edges represent face-to-face contact longer than
20 seconds. The network is composed of 410 vertices and 17298 edges.
4.3 Haggle
A human proximity network collected with help of carried wireless devices. Every vertex is a person and edges are contacts between two people. The dataset contains 274 vertices and 28244 edges [9].

4.4 Primary school
This datasets is a human proximity network containing contacts between students and teachers at a primary school. The time stamps are divided into intervals of 20 seconds and all contacts occurring during the interval get the end of the interval as time stamp. The datasets contains 242 vertices and 125773 edges [11,12].

5 Implementation
This section will go over which RRMs have been implemented and how. It will also explain the experiments that have been run and to what purpose.

5.1 Time stamp shuffling
The first method implemented is a modified version of the normal time stamp shuffling mentioned in section 3.1.2. This RRM was implemented to observe to what extent the destruction of temporal structure affects the community structure. The way this implementation differs from the one in the literature is that it picks two edges at random instead of one at each iteration to avoid that a part of the network is affected to higher degree when doing small amounts of randomization. This is important because one of the objectives was to look at how the community structure starts to differentiate from the original community structure as the network is randomized.

The function takes two parameters, a pointer to a temporal network and a double to decide which percentage of the network size the function will repeat, and works as following:

1. The double parameter size of the temporal network to determine how many times the function will repeat.
2. Get two random edges from the network.
3. Get the time stamp for both edges.
4. Set the time stamp for each edge with the opposite time stamp.
5. Return to Step 2 as many times as calculated in Step 1.

5.2 Link shuffling
The second RRM implemented is a modified version of the basic link shuffle mentioned in 3.1.1, it was implemented to primarily observe how the destruction of the topological structure affects the communities. Why and how the implemented version is modified is first and foremost the same as the time stamp shuffling mentioned above but also because of how the library used is defined.
In the library the vertices of the edges are defined as constant and can therefore
not be changed, so to be able to swap vertices between edges will edges first be
erased and then created anew with the vertices swapped.

The function takes two parameters, a pointer to a temporal network and a
double to decide which percentage of the network size the function will repeat,
and works as following:

1. The double parameter choosed to size of the temporal network to determine how
   many times the function will repeat.

2. Get two random edges from the network.

3. Check so the edges are not the same edge, if that is the case re pick an
   edge and repeat Step 3. This is so the same edge will not be erased from
   the network twice while two new edges are created, effectively increasing
   the number of edges in the network by one.

4. Get the vertices and time stamps of the edges.

5. Randomize an integer between 0 and 1 to determine which way the vertices
   will be swapped.

6. Determine if the edges being swapped will result in a self loop, if that is
   the case go back to Step 2.

7. Remove the edges from the network.

8. Create two new edges with the vertices swapped.

9. Set the time stamp for both edges.

10. Return to Step 2 as many times as calculated in Step 1.

5.3 Experiments

To get the results in this report two different types of experiment have been run.
The first type are experiments to observe the change in community structure
as the network continues being randomized. This is done by randomizing the
network a little at a time and between each randomization running the com-
munity detection algorithm. The communities found are then compared with
the communities found before any randomization is performed on the network.
If the communities before and after randomization start to differentiate from
each other this would mean that the RRM used has an effect on the network’s
community structure.

Generalized louvain is also being run a second time before changing the
network, the communities found of each run are then compared to get a base
understanding of the network’s community structure. This is possible because
the algorithm is nondeterministic and if the communities found are very differ-
ent from each other then the network probably has an undefined community
structure from the start.

All experiments of the first type follow the following pattern:

1. Slice the network and run community detection.
2. Run community detection again and calculate NMI of the two community structures.

3. Use either link shuffling or time stamp shuffling.

4. Slice the network and run community detection.

5. Calculate NMI of the community structure before and after randomization.

6. Return to Step 3.

The second type of experiments are focusing on finding an ideal number of slices for a specific network by comparing communities before and after the community structure has been destroyed with the help of link shuffle. The idea behind these experiments is to find to what extent the number of slices influences modularity, which is the measurable strength of communities. If it is possible to destroy a larger portion of the community structure, this could mean that the number of slices had less of an impact. The first experiment is used to find out to what extent the network needs to be randomized before the destruction of the community structure is achieved.

The experiments work by performing the community detection for a varying number of slices on the same network and saving the communities found. The network is then randomized until existing community structure is destroyed followed by running the community detection algorithm a vast number of times for each number of slices, this is to minimize the effect of the generalized louvain’s non-deterministic nature. The community detection is run in groups of ten and the communities with the best modularity are then picked out and compared with the communities found before randomization for that specific number of slices.

All experiments of the second type follow the following pattern:

1. Slice the network and run community detection for a specific number of slices.

2. Run community detection multiple times and calculate modularity.

3. Return to Step 1 until performed for all number of slices.

4. Use link shuffling until community structure is destroyed.

5. Slice the network, run community detection multiple times and calculate modularity for a specific number of slices. Repeat Step 5 for all the numbers of slices.
6 Results

The results will be sectioned into two parts, the first part looks at the community structure of the datasets and how the two RRM s affect it. The second part is about the results of using the destruction of community structure to find an optimal number of slices for the datasets.

6.1 Impact of random reference models on community structure

This section will show the results of running both link shuffle and time stamp shuffle on all four datasets. The shuffles are performed with 1% of network size shuffled between each round of the community detection. Every round the generalized louvain algorithm is run five times and the communities with the highest modularity is picked. This is an attempt to reduce the random aspect of generalized louvain’s non-deterministic properties. The community detection is always run with an omega value of 1 and the network sliced into 50 slices.

There are a few things to look for in the following graphs to understand the structure of the network. The first thing is the NMI before any randomization, when the x-axis is 0. This is a measure of how well-defined the network is from the start, a low NMI means that the community detection algorithm finds very different communities each run. This most likely implies that large parts of the network are very loosely connected and can end up in different communities depending on the algorithm’s path. In figure 12 we can see that hypertext and haggle datasets have low NMI while infectious: stay away and primary school have high NMI.

The second thing to look for is the difference in NMI before and after randomization as this is a measure of the impact said method had on the network’s community structure. In infectious: stay away and primary school, starting out at a high NMI, link shuffle has a higher impact on the community structure than time shuffle. The same effect can be seen in hypertext and haggle however the difference is smaller due to starting out at a lower NMI.

One extra thing to note is the extent of randomization performed on the networks: for all dataset except the primary school dataset the RRM s have been applied 50% of the network’s size times. For the primary school dataset this number is 100%. This is when it is presumed that the link shuffle have destroyed the network’s community structure and turned the network into a random graph. This is also the extent to which the networks will be randomized in the next set of experiments.
6.2 Change in community structure for different number of slices

Community structures before and after network randomization was compared for different number of slices in an attempt to find an optimal number. Each graph contains three lines corresponding to the modularity before (original) and after (shuffled) randomization and the absolute difference (diff) between them. Each dataset has been sliced with the number slices going from 1 to 200. For each slice the community detection algorithm is run ten times and the communities with the highest modularity before and after randomization is then picked from the ten. The reason for not testing a higher number of slices is because as the number increases so does the memory usage and the computation time. For most use cases the possible increase in performance is not worth the resources and time.

In figure 13 showing the hypertext dataset, both original and shuffled starts with a low modularity at a low number of slices. The modularity increase more in original but the absolute difference reaches a maximum at around 50 slices. After 50 slices original starts to plateau while shuffled continues to increase leading to a decrease in absolute difference.

In figure 14 showing the infectious: stay away dataset, the original starts with a high modularity and shuffled with a low. Therefore the absolute difference reaches a maximum at 1 slice.

In figure 15 showing the haggle dataset, it exhibits a similar pattern to hypertext but reaches a maximum earlier at about 30 slices.

In figure 16 showing the primary school dataset, it starts with a high absolute
difference similar to the infectious dataset. However, the modularity in original
increase more, compare to shuffled, as the number of slices increases leading to
a maximum at around 35 slices.

6.3 Conclusions
Comparing link and time shuffle, it is apparent that link shuffle has a stronger
impact on the community structure. This effect is clearer in datasets with a
higher base NMI. This is probably because between which vertices that contact
occurs is more important than at what time stamp it occurs, at least for these
datasets.

In three out of four datasets the absolute difference increases to reach a max-
imum around 30 to 50 slices. This is when we assume that the choice of number
of slices has the least impact on the community structure. The infectious: stay
away does not follow this pattern, instead the absolute difference starts at a
maximum. That the maximum occurs at one slice could imply that much of
the community structure is not dependent on the temporal aspect. However,
using only one slice would defeat the purpose of a temporal network. Three out
of four dataset still exhibits a maximum at 1 slice which indicates that this
method could be used to determine an optimal number of slices, where commu-
nity structure is sufficiently clear compared to the computing power necessary
to use it. This would have to be shown in future studies.

6.4 Future Work
To prove that the maximum of absolute difference yields the optimal number of
slices we would need to perform experiments on datasets where the communities
are known from the beginning and compare the results for different number of
slices. This method should also be used on more datasets to both find to what
extent the method finds the maximum and try to determine why it finds it for
some networks but not others.
Figure 13: Results of looking at the change in modularity after randomization on the hypertext dataset.
Figure 14: Results of looking at the change in modularity after randomization on the infectious: stay away dataset.
Figure 15: Results of looking at the change in modularity after randomization on the haggle dataset.
Figure 16: Results of looking at the change in modularity after randomization on the primary school dataset.
References


