University of Alberta

COMMUNITY MINING: FROM DISCOVERY TO EVALUATION AND VISUALIZATION

by

Justin Fagnan

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

©Justin Fagnan
Spring 2012
Edmonton, Alberta

Permission is hereby granted to the University of Alberta Libraries to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only. Where the thesis is converted to, or otherwise made available in digital form, the University of Alberta will advise potential users of the thesis of these terms.

The author reserves all other publication and other rights in association with the copyright in the thesis, and except as herein before provided, neither the thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatever without the author's prior written permission.

Abstract

Social networks are ubiquitous. They can be extracted from our purchase history at on-line retailers, our cellphone bills, and even our health records. Mining techniques that can accurately and efficiently identify interesting patterns in these networks are sought after by researchers from a variety of fields. The patterns they seek often take the shape of communities, which are tightly-knit groups of nodes that are more strongly related within the group than outside of the group.

This thesis proposes a series of algorithms that both accurately identify and evaluate communities in social networks. In particular we show that relative validity criteria from the field of database clustering do not serve as adequate substitutes in lieu of a ground truth. Furthermore we propose a novel community mining algorithm that considers the number of internal and external triads within each community. Finally, we present two visualization algorithms that visually expose previously difficult to obtain information regarding the structure and relationships of communities. We conclude this thesis with a brief summary of some open problems in the area of community mining and visualization.

Acknowledgements

Our research has been supported by the Canadian Natural Sciences and Engineering Research Council (NSERC), by the Alberta Ingenuity Centre for Machine Learning (AICML), the Alberta Informatics Circle of Research Excellence (iCORE), and the Government of Alberta.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction

The advent of the digital age has led to an unprecedented level of data collection, such that even the most mundane information is captured and recorded with the hope that it will lead to new and profitable insights. Many organizations seek to improve their services, and ultimately their profits, by mining this collected data for interesting patterns. For example, a hospital may improve their patient care by analyzing medical histories, a financial firm may increase profits by reviewing transactions in the stock market, or a government may seek to reduce the spread of disease by monitoring the movement of cattle in agriculture.

Although these datasets contain wildly different structures, they can all be transformed into a universal format that can easily be interpreted and analyzed by data mining algorithms. This universal format is best described as a series of entity-entity relationships. For example, a medical history could be expressed as a relationship between the patient (entity) and a symptom (entity), the patient and a disease, or the patient and an antibiotic. Once the data is converted into this entity-entity format, we can generate a network where the entities are represented by nodes and the relationships are drawn as edges (or lines) between the nodes. If these relationships are social in nature, such as friendship between people, then we refer to the generated network as a Social Network, otherwise we use the more general Information Network term.

The collected datasets are often massive and thus their associated networks can contain hundreds, thousands, or even millions of entities and relations. In the health record example, this large scale allows data mining algorithms to find patterns that span multiple patients, such as discovering that all patients with a relation to a specific disease also have a relation to a specific set of symptoms.

Deciding just how to discover these patterns in information and social networks has been an increasingly hot topic for data mining researchers. One popular technique is to focus on discovering structural patterns known as communities, which are loosely defined as a tightly-knit group of nodes that are more strongly related within the group than outside of the group. Being able to accurately and efficiently

identify communities in social networks would aid a variety of academic and industrial applications, such as the possibility of creating better pharmaceuticals by studying protein communities, detecting and thwarting organized crime communities in criminology, targeted advertising to specific communities in Facebook, and many more.

1.1.1 Thesis Statements

In this thesis we will elaborate on the challenges associated with identifying, evaluating, and visualizing communities in social or information networks. We aim to explore and improve upon each of these challenging areas by addressing the following statements:

- **TS1**: Previous metrics from the related field of data clustering may also be effective at evaluating community mining results, provided they are reimagined in the context of social networks.
- **TS2**: A node is more likely to belong to the same community as its neighbours if these neighbours are also neighbours of each other.
- **TS3**: Not all nodes need to participate in community structures.
- **TS4**: Previous well-known visualization techniques may be adapted to highlight discovered communities without sacrificing familiarity or efficiency.

1.1.2 Thesis Contributions

The three main contributions of this thesis are: a thorough evaluation and analysis of clustering metrics in social networks, a novel community mining algorithm based on the detection of triads, and a novel visualization technique that highlights the discovered communities in social networks. These contributions represent three different perspectives of the same problem: how to define and identify community structure in social networks. In the evaluation section we seek metrics that can capture the definition of a community and abuse it to determine the quality of community mining results. In our discovery section we reveal new definitions of a

community based on the distribution of triads in the network. Finally, in our visualization section we allow human intution to play a role by allowing the viewer to draw their own conclusion on what truly defines a community.

In our first contribution we focus on evaluating whether or not we can use relative validity criteria, from the existing field of data clustering, to accurately determine the quality of a community mining result. This evaluation also includes a thorough survey of the existing criteria. However, instead of presenting the original formula for each metric, we have instead provided a social network equivalent. This is necessary because the original criteria were not designed to work on datasets that are only described by relationships.

We evaluate the criteria against a series of synthetic ground truth networks and reveal that some of the criterion can outperform the existing community mining metrics on more difficult networks. Unfortunately we also show that neither the community mining metrics nor the criteria can evaluate a mining result with the same accuracy as a ground truth.

We also present a novel community mining algorithm based on the well-known local framework [13]. This algorithm is largely based on our T metric, which aims to identify communities by measuring the number of internal and external triads they contain. Here, a triad is defined as a group of three nodes that each share an edge with each other.

Furthermore we also contribute a method to detect nodes that belong to many communities (hubs) and nodes which do not belong to any community (outliers) by considering the statistical distribution of triads within each community. We show that our T metric and outlier/hub detection stage can achieve exceptional accuracy when evaluated against a collection of real-world networks.

Finally, we also propose three community visualization layouts that allow human intuition to play a role when evaluating community mining results. As far as we know these are the first layouts that reveal the structure and relationships of the communities without explicitly sacrificing their aesthetic qualities.

The first visualization we propose implements the concept of bounding boxes within a layout algorithm by forcing the nodes within a community to stay inside the

community's bounding box. This method could also be applied to a variety of other applications that seek to visually segregate a network. Our second visualization method uses a novel slotting system to force nodes onto the perimeter of a circle and features a very low time-complexity. In our final visualization algorithm we modify the attractive and repulsive forces in the Fruchterman-Reingold layout [24] such that they encourage nodes belonging to the same community to stick together, while pushing non-members away. We also contribute a preliminary evaluation of these layout algorithms and argue that our techniques provides significantly more insight into the community structure than other state-of-the-art visualization methods.

1.1.3 Thesis Organization

This thesis is organized in four parts. The first part is entirely contained in Chapter 2, where we present a brief community mining survey that provides an introduction and summary of the major frameworks and mining algorithms for social networks. This chapter also explores the future of dynamic community mining and the challenges we are faced with when trying to evaluate mining algorithms.

In the second part we present our evaluation of data clustering criteria and compare them to existing community mining metrics. In Chapter 3 we survey the existing measures and present a social network equivalent for each one. In Chapter 4 we detail our evaluation framework, including the external indices we will use to score the results. Lastly, in Chapter 5, we present the results of our evaluation and analyze why some metrics perform better on 'easier' or 'harder' networks.

In part three we discuss our novel community mining algorithm based on our T metric. In Chapter 6 we motivate the problem. In Chapter 7 we explore how others have tried to solve the community mining problem, and in Chapter 8 we present our T metric and a modified local framework that detects outliers and hubs using statistical methods. In Chapter 9 we reveal our evaluation results and show that our algorithm is significantly better than the existing approaches.

In part four we showcase both of our community visualization algorithms. In Chapter 10 we introduce the challenge of visualizating networks and in Chapter 11 we cover the existing methods in the field and show that they are inadequate at

revealing community structure. In Chapter 12 we present our modifications to the Fruchterman-Reingold [24] algorithm to support sized vertices. In Chapter 13 we discuss our COMmunity Boundary (COMB) algorithm that lays out the network by forcing each community to remain within its own separate bounding box. In Chapter 14 we present our COMmunity Circles (COMC) layout that forces nodes into 'slots' on the perimeter of a circle. In Chapter 15 we discuss our COMmunity Attraction (COMA) algorithm that uses attraction and repulsion forces to encourage nodes within a community to stay together, while pushing non-members away. In Chapter 16 we propose our edge bundling technique to reduce visual clutter and in Chapter 17 we present an evaluation of these techniques and argue that our methods offer significantly more insight into the community structure than the alternatives.

Finally, in Chapter 18 we conclude and show that each of our thesis statements have been adequately addressed. Furthermore we explore some of the open problems in the field of community mining and provide our interpretation of the challenges that researchers will face.

Part I

A Survey of Community Mining Methodologies

Chapter 2

A Survey of Community Mining Methodologies

2.1 Early Techniques

Early adopters of social network analysis tried to perform community mining by applying algorithms from the well-studied field of graph partitioning. However, these early attempts proved to be rather unsuccessful and Newman and Girvan argued that the idea was misguided as the assumptions in graph partitioning do not necessarily hold in community mining [53]. To address these concerns, researchers began looking for new methods that would be a better fit in the context of social network analysis.

One of the first successful methods they discovered was a divisive framework where one initially assumes that all nodes belong to a single massive community. The algorithm then greedily splits the community in two by choosing the division that best satisfies some metric. This continues for each iteration until there are no possible divisions that would further improve the metric. Note that we greedily split to avoid exploring every possible division of the communities, which reduces the complexity from exponential time to polynomial time.

The way this metric is defined will greatly influence the effectiveness of the framework. The first major breakthrough in defining a suitable metric came from Newman and Girvan in what is now one of the most cited papers in the field of community mining [53]. Their motivating observation was that any community should have relatively few edges linking itself to other communities. To identify

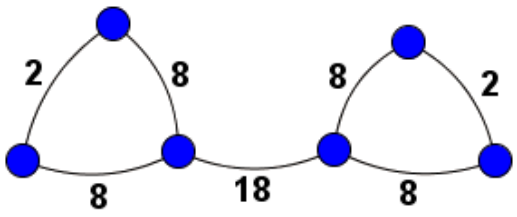


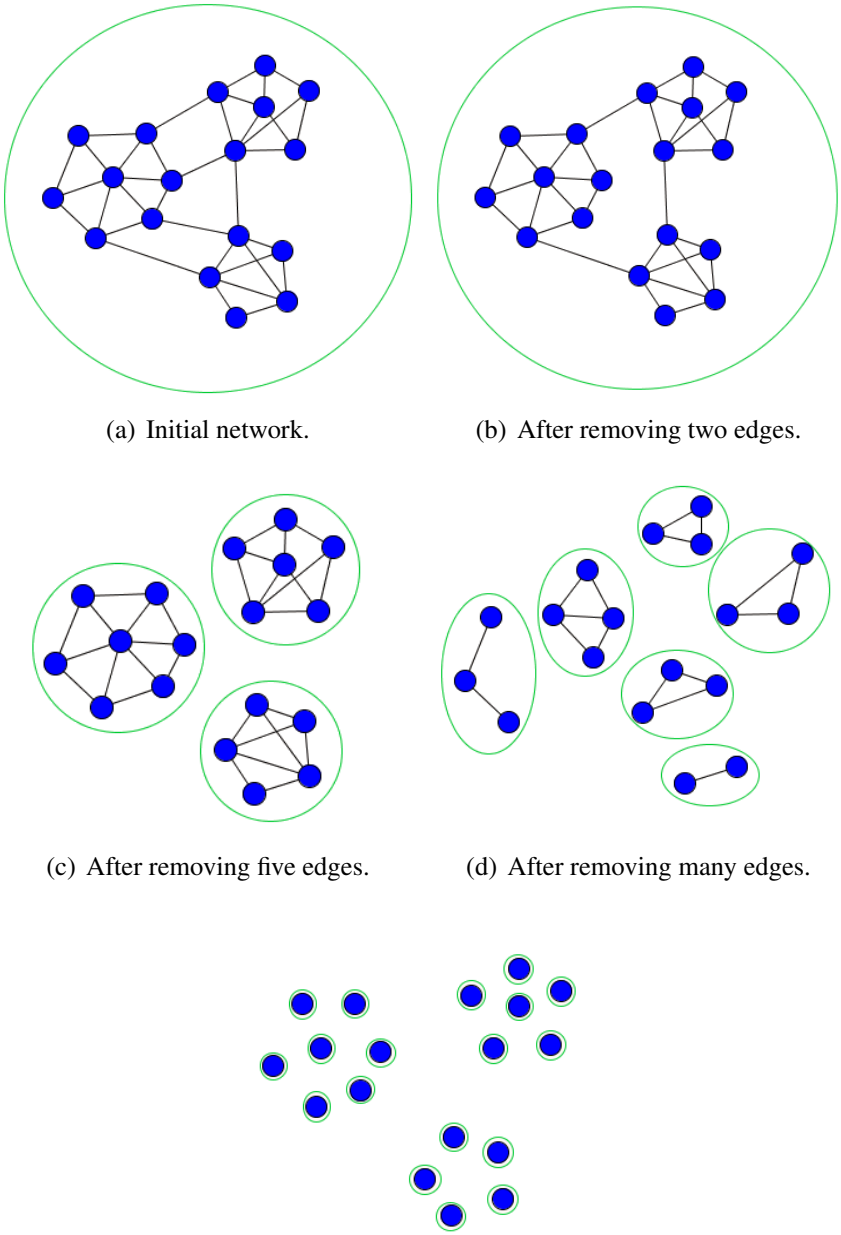
Figure 2.1: A sample network with the edge betweenness scores labeled on the edges.

these inter-community edges we can compute the shortest path between all pairs of nodes and assign a score to each edge based on the number of paths running through it. Edges that are included in a large number of paths are more likely to be inter-community edges. An example is depicted in Figure 2.1. Although this measure was previously defined as the 'rush' in an unpublished report by Anthonisse, the authors preferred to call it 'edge betweennes' based on work done by Freeman [2, 21].

To include this measure in the divisive framework, we first compute the edge betweenness score for all edges, and then remove from the network the edge which has the highest score. The resulting network should now have one less edge between the communities. We do this iteratively until there are no edges that connect the communities, as shown in Figure 2.2.

This is, however, only half of the authors' contribution, as we can see that by continually removing edges we will eventually be left with communities made up of single nodes, as in Figure 2.2(e). To avoid this we need an optimization metric that lets us know when to stop dividing the communities. A suitable metric should evaluate a proposed assignment of communities and return a score that indicates the quality of that assignment. We can then stop removing edges when we reach a maximum score for this metric, where removing any more or any less edges would detract from the score. In our example, this maximum score should occur at Figure 2.2(c).

For this purpose, Newman and Girvan introduced the Q-Modularity metric which is formulated as:



(a) Initial network. (b) After removing two edges.

(c) After removing five edges. (d) After removing many edges.

(e) No remaining edges to remove.

Figure 2.2: Divisive algorithm using Edge Betweenness. The circular borders around nodes indicate the communities.

$$Q = \sum_i (e_{ii} - a_i^2) \tag{2.1}$$

where e_{ii} is the number of edges within community i, normalized by the number of edges in the entire network. Here, a_i is the number of edges with at least one endpoint in community i, once again normalized by the number of edges in the entire network. According to our definition, a good community should maximize e_{ii} and minimize a_i, resulting in a high value of Q. In practice, a Q value between 0.3 and 0.7 often indicates a good community [53].

The results obtained by combining Q-Modularity and edge betweenness within the divisive framework were quite striking. Newman and Girvan tested their method against a generated network that had a pre-defined number of communities and were able to correctly identify all of the communities. They then applied their method to Zachary's Karate Club network, a real-world network containing 34 nodes and two known communities [66]. The membership list of these communities is known as the ground truth, which can be compared against the results of the mining algorithms. A good mining algorithm should produce the same communities as the ground truth.

The results from this evaluation were very good, with only a single node being assigned to the wrong community. Overall their algorithm looked promising, but the authors pointed out that the key disadvantage to their approach was the high computational demands, as the worst cast time complexity was $O(n^3)$ where n is the number nodes [53].

2.2 Recent Improvements

In a follow-up paper published in the same year, Newman realized that the edge betweenness measure, although very effective, was far too expensive to compute for any large network [51]. To resolve this he proposed a new community mining algorithm that relied entirely on the Q-Modularity metric defined by Newman and Girvan [53]. His new algorithm was based on the agglomerative framework, where we initially assume that each node belongs to its own community. In each iteration

of the algorithm we greedily choose the two communities whose merger would most improve the score of some metric and merge them. The algorithm concludes when there are no mergers that would further optimize the metric. As in the divisive framework, we greedily merge to avoid an exponential complexity.

In this framework, Newman used the Q-Modularity metric to determine which two communities should be merged. After showing how to further optimize the metric, he then proved that his resulting algorithm had a worst cast time-complexity of $O(n^2)$ [51]. This is significantly faster than the edge betweenness discussed earlier. Newman then evaluated his algorithm and showed that the accuracy is on-par with that of the previous approach for both computer-generated networks and Zachary's Karate Club. However, his algorithm performs poorly on the NCAA Football network, which contains 180 nodes in 11 ground truth communities. Part of the difficulty with this network is that some of the nodes do not belong to any community and we call these nodes 'outliers'.

Q-Modularity does not do well on the football network because it finds only a few major communities and no outliers, when there are in fact 11 communities and numerous outliers in the ground truth. This result is contrasted by the edge betweenness algorithm from Newman and Girvan, which correctly identifies all of the communities and outliers [53]. A possible reason for Q-Modularity's poor performance was explained by Fortunato et al., who showed that modularity based approaches face a 'resolution limit' that prevents them from identifying smaller communities [20].

2.3 Alternate Methods

While modularity based approaches were being explored, other researchers investigated frameworks that could successfully mine very large social networks. One interesting approach came from Clauset's work on local community mining [13]. In his paper, Clauset proposed a framework which is significantly different from the others in that it requires only local knowledge of the network. He argued that this requirement is necessary to solve the challenges associated with very large net-

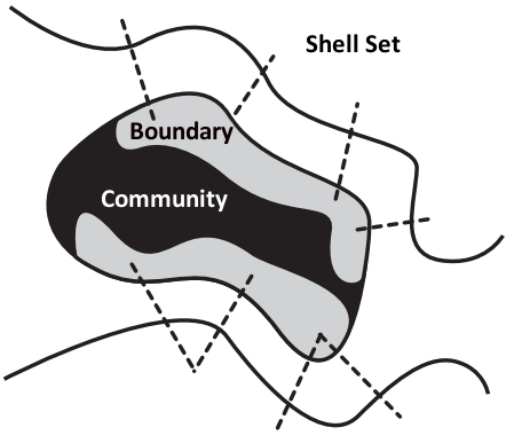


Figure 2.3: A depiction of the local community mining framework. Note that the boundary is a subset of the community. Figure adapted from Clauset [13].

works such as the World Wide Web, where we do not know the number of nodes or how they are related.

To implement his framework Clauset defined three sets: the community set, the boundary set, and the shell set. To begin, his framework randomly selects a seed node for the community. This seed node becomes a single-node community and is inserted into both the community set and the boundary set. All of its neighbouring nodes are added to the shell set. In this way, the boundary set contains all nodes which have an edge leading into the community and an edge leading into the shell set, whereas the shell set contains all possible candidate nodes that could be included in the community. A visualization of these sets is provided in Figure 2.3.

In each iteration, the algorithm greedily chooses the node from the shell set that, when included in the community, would most improve the score of some metric. This node is placed into both the community and boundary sets and its neighbours are added to the shell set. The algorithm continues until there are no nodes in the shell set that would further improve the metric. At this point the community set is output, all of the sets are cleared, and the process repeats itself on a new randomly selected node. The algorithm completes when there are no nodes left to explore in the network.

As a metric for his framework, Clauset proposed the local modularity measure R as

$$R = \frac{B_{in}}{(B_{in} + B_{out})} \tag{2.2}$$

where B_{in} is the number of edges which lead from the boundary set into the community and B_{out} is the number of edges which lead from the boundary set into the unexplored network [13]. Thus a high value of R would indicate a tightly-knit community.

Similarly, Chen et al. proposed their local community metric L based on the same framework [12]. The authors defined it as

$$L = \frac{L_{in}}{L_{ex}} \tag{2.3}$$

where

$$L_{in} = \frac{\Sigma_{i \in C} IK_i}{|C|} \tag{2.4}$$

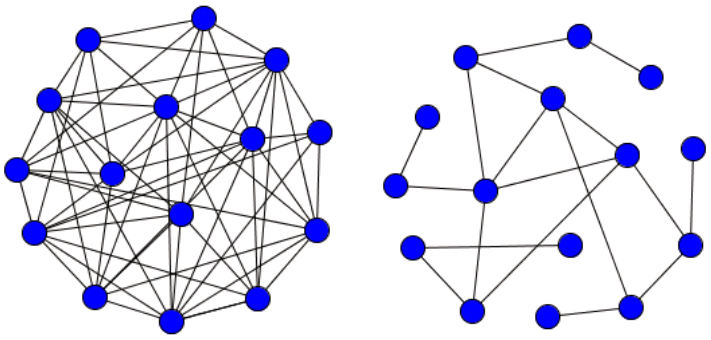
$$L_{ex} = \frac{\Sigma_{j \in B} EK_j}{|B|} \tag{2.5}$$

Here, IK_i is the number of edges between node i and the other nodes in the community, and $|C|$ is the number of nodes in the community. EK_j is the number of edges between node j and the nodes in the unexplored network, and $|B|$ is the number of nodes in the boundary set. Thus we want to maximize L_{in} and minimize L_{ex}, resulting in a high value of L.

Chen et al. argued that by being oblivious to the number of nodes, the R measure does not actively select for high density communities [12]. A community has high density if the number of edges is close to the maximum number of possible edges ($n(n-1)/2$), as shown in Figure 2.4(a). This property is desired as it coincides well with the very definition of a community.

To provide evidence for their claim, Chen et al. first revealed that the R metric outperforms all other known methods for local community detection. They then evaluated both metrics against the NCAA Football network and showed that their L metric severely outperforms the R metric in precision and recall [12].

Furthermore the results from the L metric nearly match that of the edge betweenness algorithm on the NCAA Football network. The main reason for this is the ability of the L metric to detect outlier nodes. The algorithm actively identifies



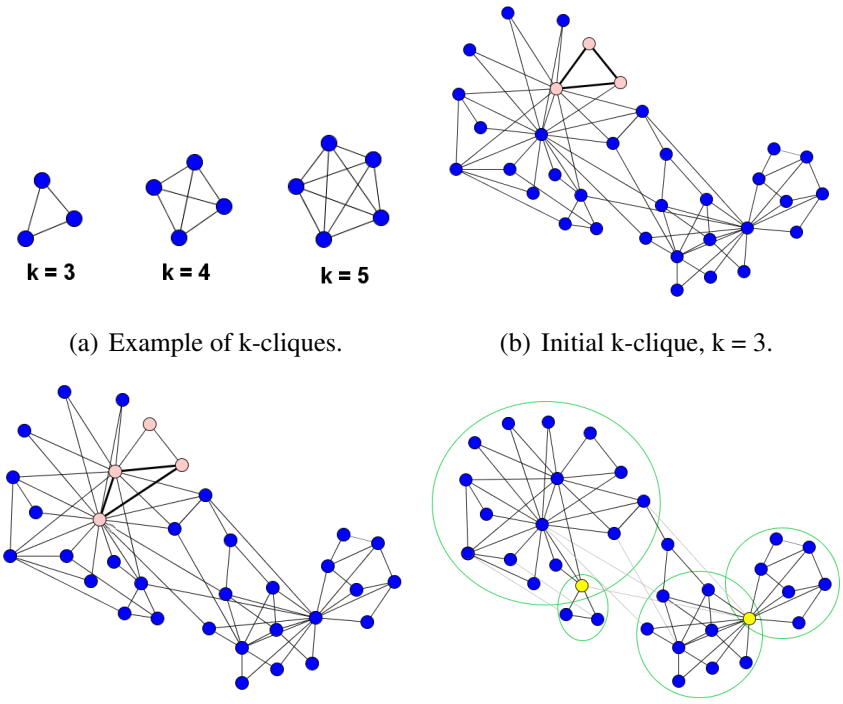
(a) A High Density Community (b) A Low Density Community

Figure 2.4: An example of high and low density communities.

outliers by investigating the values of L_{in} and L_{ex} before and after merging each node into the community. As we can see from Formula (2.3), if the L_{in} value goes up and the L_{ex} value goes down, then this node is a good candidate to merge into the community. If, however, the values of L_{in} and L_{ex} go down then this node must not have many neighbours inside the community nor does it have many outside [12]. Thus this node is an outlier and does not belong to any community.

In addition to Clauset's local framework, other researchers pursued methods that allow a node to belong to more than one community, commonly called overlap. There is much real world motivation for overlap as, for example, people rarely belong to a single club or have only one group of friends. Although Clauset's local framework can be adapted to allow for overlap, the most well known framework for this problem is the Clique Percolation Method (CPM) proposed by Palla et al. [56].

In this framework the authors propose an algorithm that discovers communities based on the existence of k-cliques. They define a k-clique as a fully-connected sub-network with k nodes, where each node has an edge to all of the other $k-1$ nodes, as shown in Figure 2.5(a). The CPM algorithm can be best described as placing a k-clique over any k nodes in the network that are fully connected and then rolling the k-clique around until it is unable to reach any unexplored nodes. We can roll a k-clique by swapping a single node in the clique for one outside of the clique and keeping the others fixed. When rolling, the clique must remain fully connected, as shown in Figure 2.5(c).



(a) Example of k-cliques.

(b) Initial k-clique, k = 3.

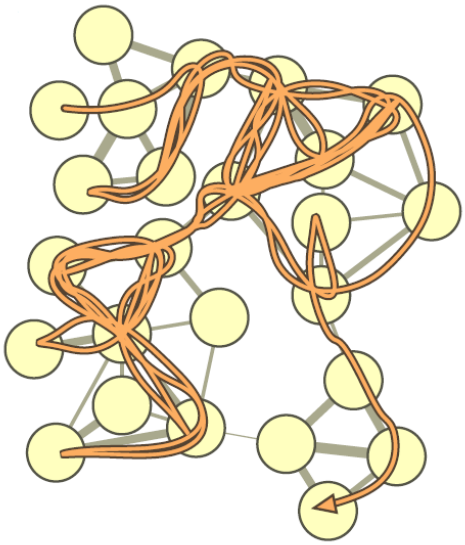
(c) Example of rolling the k-clique once.

(d) Communities found from the CPM method.

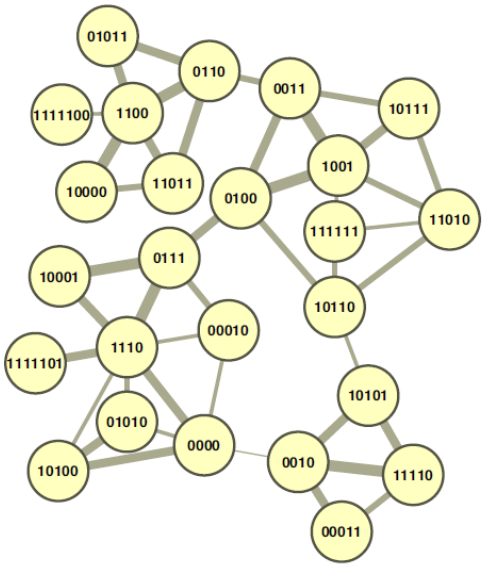
Figure 2.5: An example run of the Clique Percolation Method with k = 3.

All of the nodes encountered while rolling are considered a single community, as shown in Figure 2.5. Once we cannot roll anymore, we reposition the k-clique to an unexplored region of the network and repeat. Notice that we may roll over many of the same nodes even after we reposition; it is this behaviour that leads to overlapping communities. In their framework, the authors leave the choice of the parameter k up to the user, but they recommend $k = 3$ for small networks, $k = 4$ for large networks, and $k = 5$ for very large networks. Unfortunately, they could not provide any convincing evaluations as there are no networks with an overlapping ground truth.

Other researchers have posited that even local methods are too slow for very large networks because the metrics they require are inherently expensive to compute. To address these concerns, Rosvall and Bergstrom have proposed their Infomap algorithm, which takes the unorthodox approach of viewing community mining as an information theory problem [60]. To begin, the authors generate a

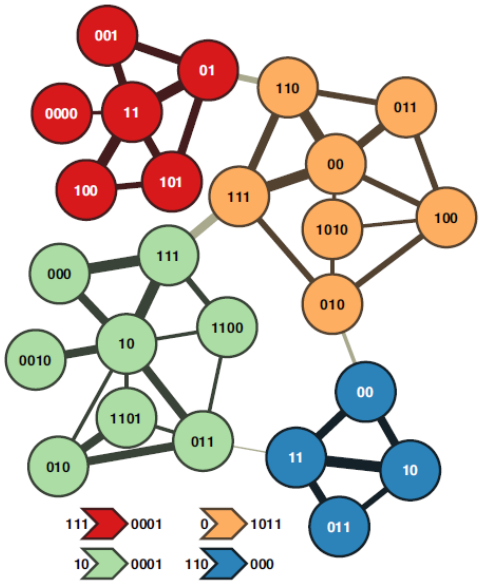


(a) Random walker path.



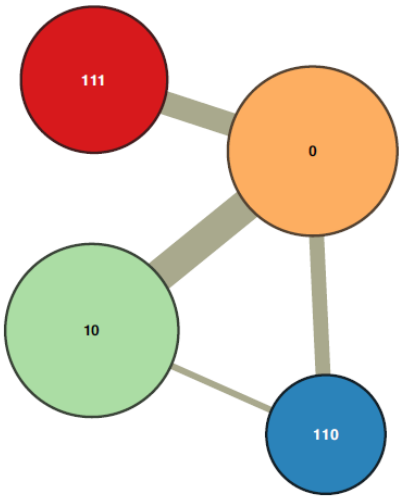
1111100 1100 0110 11011 10000 11011 0110 0011 10111 1001 0011 1001 0100 0111 10001 1110 0111 10001 0111 1110 0000 1110 10001 0111 1110 0111 1110 1111101 1110 0000 10100 0000 1110 10001 0111 0100 10110 11010 10111 1001 0100 1001 10111 1001 0100 1001 0100 0011 0100 0011 0110 11011 0110 0011 0100 1001 10111 0011 0100 0111 10001 1110 10001 0111 0100 10110 111111 10110 10101 11110 00011

(b) Example Huffman encoding.



111 0000 11 01 101 100 101 01 0001 0 110 011 00 110 00 111 1011 10 111 000 10 111 000 111 10 011 10 000 111 10 0010 10 011 010 011 10 000 111 0001 0 111 010 100 011 00 111 00 011 00 111 00 111 110 111 110 1011 111 01 101 01 0001 0 110 111 00 011 110 111 1011 10 111 000 10 000 111 0001 0 111 010 1010 010 1011 110 00 10 011

(c) Infomap community-based encoding.



111 0000 11 01 101 100 101 01 0001 0 110 011 00 110 00 111 1011 10 111 000 10 111 000 111 10 011 10 000 111 10 111 10 0010 10 011 010 011 10 000 111 0001 0 111 010 100 011 00 111 00 011 00 111 00 111 110 111 110 1011 111 01 101 01 0001 0 110 111 00 011 110 111 1011 10 111 000 10 000 111 0001 0 111 010 1010 010 1011 110 00 10 011

(d) Highlighting the community enter/exit codes.

Figure 2.6: An example network with Huffman and Infomap encoding. Images reprinted from Rosvall and Bergstrom [60].

random walk of the network, with the goal of accurately representing the walker's path in a few bits as possible. They then show that the obvious solution to this problem is to employ a Huffman encoding of the nodes, which assigns each node a unique code based on how often it is encountered in the random walk [34]. Nodes that are encountered frequently are given shorter codes than those which are rarely encountered. A potential Huffman encoding is shown in 2.6(b).

Although the Huffman encoding significantly reduces the number of bits required to express the path, it does not capitalize on the structural features in the network. In particular, once the random walker enters a dense region of the network, such as a community, it is likely to stay within that region for an extended period of time. By identifying these communities and assigning uniques codes to them we can create a hierarchical encoding, such as two nodes can have the same identifying code so long as they belong to different communities, just like how two different cities can have the same street names [60].

By leveraging the unique community codes, Infomap can now compute a separate Huffman encoding for the members of each community, rather than a single one for the entire network. This reduces the number of vertices per encoding and thus reduces the number of bits per unique code. To include this hierarchical encoding in the path, we add the community code whenever the walker enters a community and an exit code whenever it leaves a community. Thus it is imperative that the walker stay within a community for more than few steps, otherwise the overhead of using community codes will quickly erase any gains from the shorter unique vertex codes. This implies that the optimal compression of the random walker's path will also contain an optimal partitioning of the network into communities.

To determine this optimal compression, the Infomap algorithm employs the agglomerative framework, and at each step it merges the two communities that would maximize the level of compression. This compression is computed by considering the entropy of the random walk between, and within, the communities. Once the framework reaches a maximum compression the communities are extracted and their membership is fine tuned through a simulated annealing procedure. Unlike Newman's Modularity metric, the Infomap compression metric is blazingly fast to

compute and has allowed the authors to apply their method on networks as large as 2.6 million nodes and 29 million edges [51, 60].

Although no concrete evaluation is provided, the authors show some intuitive results indicating that flow of information in the network is better captured by their method than by the traditional modularity-based methods. We should also note that the Infomap method natively supports directed and weighted networks by including a small probability for the walker to teleport to a random node in the network.

2.4 Cutting Edge Research

Thus far we have explored frameworks that are used to detect communities in networks that do not change over time, otherwise known as static networks. However, there is also interest in algorithms that can identify communities in dynamic networks. This would be useful, for instance, in tracking how a disease spreads from one community to the next, or how a company's organizational structure evolves over time.

To solve this problem, Asur et al. introduced their event based framework [3]. Their main idea is to break down a dynamic network into a series of static snapshots and perform analysis on these snapshots. For example, when provided with a network that changes continually throughout the year, they take a snapshot of the network at the end of each month. The resulting snapshots are static, but when put together they provide a summary of the dynamic changes in the network. The community mining task can now be accomplished using any of the static algorithms that we previously discussed. The challenge, however, is to identify how the communities have changed between the snapshots.

The method Asur et al. proposed detects when a community is involved in one of five basic events. A community has Continued, the first event, when there is a community in the next snapshot that contains exactly the same nodes as this community. Two communities are k-Merged, the second event, if there is a community in the next snapshot that contains at least $k\%$ of the nodes from these two communities. A community has k-Split, the third event, if two communities in the next

snapshot contain at least $k\%$ of the nodes from this community. A community has Formed, the fourth event, if none of its nodes have belonged to the same community in a previous snapshot. Finally, a community has Dissolved if none of its nodes appear together in any community in the next snapshot [3].

Asur et al. showed that when applied against real-world networks these events can provide insight into how communities evolve over time. Takaffoli et al. argued that this framework is too rigid as real-world communities rarely maintain the exact same members over time and furthermore they rarely dissolve entirely [62]. To address their concerns, Takaffoli et al. redefined all of the events so that they are dependent on the parameter k. As an example, their Form event requires that only $k\%$ of the nodes are new to this community. They also argued that the framework from Asur et al. does not cover all possible events, such as when a community loses or gains nodes. To capture this information they added two new events, k-Shrink to detect when a community loses nodes, and k-Reform to detect when a community gains nodes.

Takafolli et al. [62] evaluated both frameworks against a series of networks from monthly snapshots. Unfortunately, their evaluation is not conclusive as there are no dynamic networks with a ground truth. Instead, Takaffoli et al. showed that their framework identifies an event for each community at each snapshot, while the framework from Asur et al. does not [62]. They argued that it follows from intuition that every community should be involved in at least one event at each snapshot.

2.5 Evaluation Challenges

The evaluation challenges that are encountered in dynamic frameworks are an indicator of a much broader problem facing the field of community mining. Although much time has been spent researching new frameworks, relatively little time has been spent researching methods to evaluate these frameworks. Without proper evaluation methods it is very difficult to claim that one algorithm is better than another. This is especially true in papers that only evaluate their algorithm against networks which show good results, leaving the reader with a false sense of confidence in the

algorithm.

Recently, Lancichinetti et al. have published benchmarks that evaluate community mining algorithms against computer-generated networks that have statistically valid structure [44] . More recently yet, Leskovec et al. have published a framework to empirically evaluate algorithms against a variety of real-world networks where the ground truth is known [45]. As of yet there has been no convincing evaluation or analysis regarding the accuracy of these methods and thus the evaluation problem remains unsolved.

We face additional challenges when dealing with overlap as we do not know of any networks with an overlapping ground truth. The most promising solution is offered by Lancichinetti et al., which proposes a benchmark that has been modified to handle overlap [41]. Unfortunately, it is unclear whether or not these benchmark methods will be adopted by the research community at large.

To make matters worse, to the best of our knowledge there are no suitable methods to evaluate the results of dynamic community mining. This is because it is unclear which events should be included in a computer-generated network. Any particular choice or definition of an event will bias the results of the evaluation towards the framework that proposed the event. Currently there are no approaches that address this difficult problem.

Part II

An Evaluation of Relative Validity Criteria for Community Mining

Chapter 3

Clustering Transformations & Relative Validity Criteria

3.1 Motivation

Although much progress has been made in identifying communities, very little work has been done on evaluating the results of the community mining algorithms. The obvious approach to evaluate these algorithms is to compare their identified communities to a ground truth, where the correct membership of each community is known. This approach, however, is unsatisfactory as we are rarely provided with the ground truth of real world social networks. A more suitable approach is to consider the outcome of some relative validity criteria, which ranks the results of two or more mining algorithms by evaluating them against some notion of what a good community should be. This approach is more versatile as it does not require a ground truth and thus can be applied to any social network.

Unfortunately, defining such a relative validity criterion is non-trivial as there is no clear consensus on what a 'good community' should look like. In light of this, researchers have proposed a variety of such criteria that focus on density, modularity, internal/external edge ratios, and other graph theory metrics. In this thesis we extend this list by re-introducing criteria that originated in the well-known field of database clustering, and show how these criteria can be transformed to evaluate community mining results. We then perform an experiment on these newly transformed criteria to rank them based on the quality of their evaluations when compared to the ground truth. We conclude with a brief analysis of these results.

3.2 Distance Methods

The process of clustering is commonly viewed as a sister process of community mining, as both tasks focus on producing tightly-knit groups of entities that are maximally separated from each other. Due to this similarity, it follows from intuition that many of existing relative validity criteria from clustering could be applied in context of community mining. To do so, we must first address the differences in how each method calculates the distance between two entities and how they compute the centroid of a group. We cannot, for example, calculate the Euclidean distance between two entities in a social network, as this value is undefined.

To address the differences in distance methods, we notice that although the distances used in each method are syntactically different, they are semantically the same. Thus we may simply replace any cluster-based distances with a valid graph theory distance. For this purpose we have selected four suitable distance functions:

3.2.1 Shortest Path (SP)

This distance method computes the shortest path between two nodes by using the well known Dijkstra's Shortest Path algorithm.

3.2.2 Adjacency Relation Distance (ARD)

This measure considers the structural equivalence of two nodes by comparing their immediate neighbourhoods. For example, if two nodes share all of the same neighbours they would have an ARD score of 0. Otherwise, the distance is a function of the number of neighbours the nodes do not share. More formally:

$$ARD_{i,j} = \sqrt{\sum_{k \neq j,i} (A_{ik} - A_{jk})^2}$$

where A is the adjacency matrix of the network, and i and j are nodes in the network [19].

3.2.3 Neighbour Overlap Distance (NOD)

This measure is very similar to the Adjacency Relation Distance in that it also compares the immediate neighbourhood of both nodes. The NOD between two nodes is the ratio between the intersection and union of the neighbourhoods. If two nodes share all of the same neighbours they would have an NOD of 1. More formally:

$$NOD_{i,j} = 1 - \frac{R_i \cap R_j}{R_i \cup R_j}$$

where R_i is the set of nodes in the immediate neighbourhood of node i. We subtract the ratio from 1 so that we are consistent in implying that a low distance score indicates that the two nodes are close together [19].

3.2.4 Pearson Correlation Distance (PCD)

This measure considers the Pearson correlation value between the columns and rows of the adjacency matrix. More formally:

$$C_{i,j} = \frac{\sum_k (A_{ik} - \mu_i)(A_{jk} - \mu_j)}{n\sigma_i\sigma_j}$$

where the averages $\mu_i = (\sum_j A_{ij})/n$ and the variances $\sigma_i = \sqrt{\sum_j (A_{ij} - \mu_i)^2/n}$ [19].

3.3 Centroid Methods

In addition to this distance transformation, we also resolve the differences in computing centroids by replacing any clustering-based centroid methods with graph-based centroid methods. In the context of social networks, a centroid is the single node which best represents the 'centre' of a community. This differs from clustering, where the centroid of a cluster may not be an actual data point, but rather an arbitrary point that represents the averages. To compute the centroid in a social network, we consider the following approaches:

3.3.1 Betweenness (B)

In this method, the centroid of the community is the node which has the highest betweenness centrality score [21]. If we consider the shortest path between every pair of nodes in the community, then the betweenness score for a node is the number of these shortest paths in which it is included. More formally:

$$B_C = arg\,max_{i \in C}(Score_{i,C})$$
$$Score_{i,C} = \sum_{j \in C} \sum_{k \in C} P_{j,k}(i)$$

where C is the community, $P_{j,k}(i)$ is 1 if the shortest path between node j, and k contains the node i; 0 otherwise. The time complexity of this method is $O(|C|^3)$, as we need to compute the shortest path between every pair of nodes in the community.

3.3.2 Closeness (C)

In this method, the centroid of the community is the node with the lowest closeness centrality score. The closeness centrality score of a node is its average shortest path distance to all other nodes in the community. More formally:

$$C_C = arg\,min_{i \in C}(Score_{i,C})$$
$$Score_{i,C} = \frac{\sum_{j \in C} dist(i,j)}{|C|}$$

where C is the community, and $dist(i,j)$ is the distance function between nodes i and j. The time complexity of this method is $O(|C|^3)$ due to the shortest path computation.

3.3.3 Degree (D)

In this simple method, the centroid of the community is the node with the highest degree. More formally it is:

$$D_C = arg\,max_{i \in C}(Degree_i)$$

If we assume that the degree operation is constant time then the time complexity of this method is $O(|C|)$.

3.3.4 Distance Variance (DV)

In this method, the centroid of the community is the node with the lowest distance variance score [65]. This score is the variance of the shortest path distances between a node and all other nodes in the community. More formally it is:

$$DV_C = arg\,min_{i \in C}(Score_{i,C})$$
$$Score_{i,C} = \frac{\sum_{j \in C}(dist(i,j) - avgdist_i)^2}{|C|}$$

where the average distance from one node to all others is $avgdist_i = \sum_{j \in C} dist(i,j)/|C|$, and C is the community. This method requires computing the shortest path for every pair of nodes, thus the time complexity is $O(|C|^3)$.

3.3.5 Estimated Closeness (EC)

This approach aims to reproduce the results of the Closeness method, but with a reduced time-complexity. The key idea here is to take a random sample of k nodes from the community and only compute the shortest paths from these k nodes to all other nodes. To determine k, we must set the value of epsilon, which indicates our desired tradeoff between speed and error level. In our case we have set ϵ = 1, to achieve the best possible speed optimization. More formally:

$$E_C = arg\,min_{i \in C}(Score_{i,C})$$
$$Score_{i,C} = \frac{\sum_{k \in K}(dist(i,j))}{|C|}$$
$$k = \left\lfloor \log\left(\frac{|C|}{\epsilon^2}\right) \right\rfloor$$

where $0 \leq \epsilon \leq 1$, C is the community, and K is the set of k randomly selected nodes. This method requires computing the single source shortest path for each of the k sample nodes, thus its time-complexity is $O(kn^2)$ [18].

3.4 Relative Validity Criteria

For our experiment we have selected 15 well-known relative validity criteria from the field of clustering, based on work by Vendramin et al. [64]. For each of these criteria we have replaced any distance calculations with one of the above distance functions, and any centroid calculations with one of the above centroid functions. In addition, we have removed any attribute-based operations, as current community mining algorithms do not consider attributes.

We do not include a time-complexity analysis for all of the criteria because it is dominated by the centroid and distance methods.

3.4.1 C-Index

This criterion is based on the sum of the distances between nodes in the same community [33]. This sum is then compared to both the best case scenario, where the within-community distances are the shortest distances in the graph, and the worst case scenario, where the within-community distances are the longest distances in the graph. The output of this criterion indicates how close the within-community distances are to the best case scenario. Thus a low C-Index value indicates a good community mining result. More formally:

$$CIndex = \frac{\theta - \min\theta}{\max\theta - \min\theta}$$
$$theta = \sum_{i=1}^{N-1}\sum_{j=i+1}^{N} dist(i,j) * \delta(i,j)$$

where the T shortest distances $\min\theta = \sum_{i=0}^{T} distances[i]$, the T longest distances $\max\theta = \sum_{i=1}^{T} distances[|distances| - i]$, the number of within community distances $T = \sum_{i=1}^{N-1}\sum_{j=i+1}^{N}\delta(i,j)$, and $distances[]$ is an array containing all of the distances between any pair of nodes in the network, sorted in ascending order. Here, $\delta(i,j)$ is 1 if nodes i and j are in the same community, 0 otherwise, and N is the nodes in the network.

3.4.2 C/$\sqrt{k}$

This criterion considers the ratio between the sum of the within-community distances, and the sum of each node's distance to the centroid of the entire network [59]. The former is used as a compactness measure, and the latter as an estimation of how spread out the communities are. Therefore, this criteria selects community mining results that have a high value of C/$\sqrt{k}$, which maximizes the distance to the centroid of the network and minimizes the within-community distances. More formally:

$$C/\sqrt{k} = \frac{1}{\sqrt{|k|}}\sqrt{1 - \frac{SSW}{SST}}$$
$$SSW = \sum_{C \in k} \sum_{i \in C} dist(i, \overline{C})^2$$
$$SST = \sum_{i \in N} dist(i, \overline{N})^2$$

where $\overline{C}$ is the centroid of community C, $\overline{N}$ is the centroid of the entire network, and k is the set of communities. As Vendramin et al. points out, this normalization via $1/\sqrt{k}$ is an attempt to counter-balance the decrease in SSW, and therefore the increase in the $\sqrt{1 - (SSW/SST)}$ term, as $|k|$ increases [64]. However this normalization does not fully correct for this behaviour, as an increasing $|k|$ eventually leads to the case where SSW = 0 and thus C/$\sqrt{k}$ = $1/\sqrt{k}$. Therefore this criterion is biased against community mining results that contain many communities.

3.4.3 Davies-Bouldin

This criterion considers the distances between two communities relative to their compactness [16]. Specifically, this method selects and evaluates the pair of communities that have the worst possible ratio of within-community to between-community distances. A community mining result with a low Davies-Bouldin value is desired as this indicates small within-community distances and large between-community distances. More formally:

$$DB = 1/|k| \sum_{C \in k} D_C$$
$$D_C = arg\,max_{\substack{M \in k \\ M \neq C}} (D_{C,M})$$

$$D_{C,M} = \frac{avgDist(C) + avgDist(M)}{dist(\overline{C}, \overline{M})}$$
$$avgDist(X) = \frac{1}{|X|} \sum_{i \in X} dist(i, \overline{X})$$

where $avgDist()$ represents the average within-community distance to the centroid, k is the set of communities, $\overline{C}$ is the centroid of community C, and $\overline{M}$ is the centroid of community M.

3.4.4 Dunn

This criterion considers both the minimum distance between any two communities and the length of the largest community diameter [17]. The diameter is defined as the longest shortest path within a community. This criterion rewards community mining results that have large distances between the communities and small community diameters; resulting in a large Dunn value. More formally:

$$Dunn = min_{\substack{P,Q \in k \\ P \neq Q}} \left(\frac{dist(\overline{P}, \overline{Q})}{max_{C \in k}(diameter(C))} \right)$$
$$diameter(C) = max_{i,j \in C}(dist(i,j))$$

where $\overline{P}$ is the centroid of community P, $\overline{Q}$ is the centroid of community Q, and k is set of communities. We should note that there are a variety of ways to compute both the diameter of a community and the distance between two communities. We only present a single approach here. Other cluster-based variations can be found in the paper by Bezdek and Pal [7].

3.4.5 Gamma

This criterion considers the distance between two nodes from the same community, relative to the distance between two nodes that belong to different communities [4].

If the distance of the former pair is shorter than the distance of the latter pair, the criteria increments a 'closer' counter. Otherwise the criteria increments a 'further' counter. The score of the community mining result depends on the ratio of these two counters. A high Gamma value indicates that the communities are separated by sufficient distance given their compactness. Therefore a high value of Gamma is desired. More formally:

$$
\begin{aligned}
G &= \frac{S_{closer} - S_{further}}{S_{closer} + S_{further}} \\
S_{closer} &= \frac{1}{2} \sum_{L \in k} \sum_{\substack{i,j \in L \\ i \neq j}} \frac{1}{2} \sum_{M \in k} \sum_{\substack{p \in M \\ q \notin M}} \delta(dist(i,j) < dist(p,q)) \\
S_{further} &= \frac{1}{2} \sum_{L \in k} \sum_{\substack{i,j \in L \\ i \neq j}} \frac{1}{2} \sum_{M \in k} \sum_{\substack{p \in M \\ q \notin M}} \delta(dist(i,j) > dist(p,q))
\end{aligned}
$$

This criterion has a very high time-complexity due to the comparison between every pair of nodes. As a result, the estimated time-complexity of Gamma is $O(N^3 + \frac{N^4}{k})$; a full discussion can be found in work by Vendramin et al. [64]. Therefore the Gamma criterion is impractical for many social networks.

3.4.6 PBM

This criterion is based on the within-community distances, distances to the centroid of the entire network, and the maximum distance between two centroids [55]. A large PBM value is desirable as it indicates that the community mining result has very compact and spread out communities. More formally:

$$
\begin{aligned}
PBM &= \left(\frac{1}{|k|} \frac{E_{centre}}{E_{within}} D_{between} \right)^2 \\
E_{centre} &= \sum_{i \in N} dist(i, \overline{N}) \\
E_{within} &= \sum_{C \in K} dist(i, \overline{C}) \\
D_{between} &= arg\, max_{C_1, C_2 \in k}(dist(\overline{C_1}, \overline{C_2}))
\end{aligned}
$$

where N is the set of nodes in the network, $\overline{N}$ is the centroid of the entire network, $\overline{C}$ is the centroid of the community C, $\overline{C_1}$ is the centroid of community C_1, $\overline{C_2}$ is the centroid of community C_2, and k is the set of communities.

3.4.7 Point-Biserial

This criterion computes how distances between two nodes correlate to the community membership of those nodes [50]. Intuitively, nodes that are in the same community should be separated by shorter distances than those which are not. A large Point-Biserial value indicates that the nodes in the same community are closer. More formally:

$$PB = \frac{(avgW - avgB)\sqrt{numW * numB/t^2}}{stdDev}$$
$$avgW = \frac{1}{numW}\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} dist(i,j) * \delta(i,j)$$
$$avgB = \frac{1}{numB}\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} dist(i,j) * (1 - \delta(i,j))$$

Here avgW is the average within-community distance, avgB is the average between-community distance, numW is the number of within community distances ($\sum_{C \in k} \frac{|C|(|C|-1)}{2}$), numB is the number between community distances ($numPairs - numW$), numPairs is the number of distances in the network ($\frac{|N|(|N|-1)}{2}$), stdDev is the standard deviation of all the distances in the network, and k is the set of communities.

3.4.8 Silhouette Width Criterion

This criterion measures the distance between a node and the centroid of the community it belongs to [37, 61]. It then compares this value to the distance between that node and the nearest community it does not belong to. The former distance should be shorter than the latter. The differences are summed up and normalized by the number of nodes. A large Silhouette Width Criterion value is desired, as it indicates that the nodes are much closer to their own centroids than they are to the centroids of the other communities. More formally:

$$SWC = \frac{1}{N}\sum_{i \in N} silhouette(i)$$
$$silhouette(i) = \frac{b_i - a_i}{max(a_i, b_i)}$$
$$b_i = \frac{1}{|C_{nearest}|}\sum_{j \in C_{nearest}} dist(i,j)$$
$$a_i = dist(i, \overline{C})$$

where $C_{nearest}$ is the nearest neighbouring community that node i does not belong to, and $\overline{C}$ is the centroid of the community that node i belongs to. Deciding the nearest neighbour is done by computing the distances between the centroids of each community and selecting the nearest one.

3.4.9 Variance Ratio Criterion

This criterion is a ratio of the distance from each community's centroid to the network's centroid and the within-community distance [8]. The centroid to centroid distances are also weighted by the number of nodes in the community. This ratio is then normalized to account for increases in the number of communities. A large Variance Ratio Criterion value indicates that the communities are spread far apart and are rather compact. More formally:

$$VRC = \frac{BetweenDist}{WithinDist} * \frac{|N| - |k|}{|k| - 1}$$
$$BetweenDist = \sum_{C \in k} |C| * dist(\overline{C}, \overline{N})^2$$
$$WithinDist = \sum_{C \in k}\sum_{i \in C} dist(i, \overline{C})^2$$

where $\overline{C}$ is the centroid of the community C, $\overline{N}$ is the centroid of the entire network, and k is the set of communities.

3.4.10 WG

This criterion is the sum of the within-community distances [22]. Here, the within-community distance is the sum of the distance from each node to the centroid of

its community. For a network with k as its set of communities the WG can be calculated as:

$$WG = \sum_{C \in k} \sum_{i \in C} dist(i, \bar{C})^2$$

where $\bar{C}$ is the centroid of the community C.

3.4.11 covWG

This criterion is similar to the WG and is based on the sum of the within-community distances [50]. However, the within-community distance is normalized by the difference between the number of nodes in the network and the number of communities. Formally the covWG is:

$$covWG = \frac{1}{|N| - |k|} \sum_{C \in k} \sum_{i \in C} dist(i, \bar{C})^2$$

where $\bar{C}$ is the centroid of the community C, $|N|$ is the number of nodes in the network, and k is the set of communities.

3.4.12 B/W

This criterion is a ratio of the distance from each community's centroid to the network's centroid, and the within-community distance [22]. The centroid to centroid distances are also weighted by the number of nodes in the community. This criterion is very similar to the Variance Ratio Criterion except that the ratio is not normalized. More formally:

$$B/W = \frac{\sum_{C \in k} |C| * dist(\bar{C}, \bar{N})}{\sum_{C \in k} \sum_{i \in C} dist(i, \bar{C})^2}$$

where $\bar{C}$ is the centroid of the community C, $\bar{N}$ is the centroid of the entire network, and k is the set of communities.

3.4.13 log(SSB/SSW)

This criterion is a ratio of the between-community and within-community distances [30]. The between-community distances are calculated as the distances between the centroids of each community. The between-community distances are also weighted by the number of nodes in the communities. The log(SSB/SSW) is calculated as follows:

$$
\begin{aligned}
log(SSB/SSW) &= \log(\frac{BetweenDist}{WithinDist}) \\
BetweenDist &= \sum_{l=1}^{|k|-1} \sum_{m=l+1}^{|k|} \frac{dist(\bar{C_l}, \bar{C_m})^2}{1/|C_l| + 1/|C_m|} \\
WithinDist &= \sum_{C \in k} \sum_{i \in C} dist(i, \bar{C})^2
\end{aligned}
$$

where $\bar{C}$ is the centroid of the community C, and k is the set of communities.

3.4.14 Ball and Hall

This criterion is the sum of the within-community normalized by the total number of nodes in the network [5]. More formally:

$$
BallandHall = \frac{1}{|N|} \sum_{C \in k} \sum_{i \in C} dist(i, \bar{C})
$$

where $\bar{C}$ is the centroid of the community C, $|N|$ is the number of nodes in the network, and k is the set of communities.

3.4.15 McClain and Rao

This criterion is a ratio of the between-community distances and the within-community distance [48]. Unlike most of the criteria, the between-community and within-community distances are not calculated using the distance to the centroid. Instead,

the distance is calculated for each pair of communities as follows:

$$McClainandRao = \frac{BetweenDist/(N^2 - \sum_{C \in k} |C|^2)}{WithinDist/((\sum_{C \in k} |C|^2) - |N|)}$$

$$BetweenDist = \sum_{l=1}^{|k|-1} \sum_{m=l+1}^{|k|} \sum_{i \in C_l} \sum_{j \in C_m} dist(i,j)$$

$$WithinDist = \frac{1}{2} \sum_{C \in k} \sum_{i,j \in C} dist(i,j)$$

where $|N|$ is the number of nodes in the network, and k is the set of communities.

3.4.16 Modularity

We have also included the well-known Modularity criteria from Newman [52]. This criterion considers the difference between the fraction of edges that are within the community and the expected such fraction if the edges were randomly distributed. A higher Modularity value is desired as it indicates that there are significantly more within-community edges than expected. More formally:

$$Modularity = \frac{1}{2m} \sum_{i,j \in N} \left(A_{ij} - \left(\frac{d(i) * d(j)}{2m} \right) \right) * \delta(i,j)$$

where A is the adjacency matrix, m is the number of edges in the network, $\delta(i,j)$ is 1 if nodes i and j are in the same community, 0 otherwise, and $d(i)$ is the degree of node i. We should note that Fortunato and Barthélemy showed that modularity cannot accurately evaluate small communities due to its resolution limit [20].

These 16 metrics will be considered in our evaluation in order to determine which of them performs the best in the context of community mining analysis.

Chapter 4

Experiment Setup

In order to perform our evaluation we first need a large collection of networks with an associated ground truth. Unfortunately, we are unable to use real-world networks for this purpose, as there is only a handful with an associated ground truth. Therefore we must use generated networks for the majority of our evaluations as such networks come with an associated ground truth. For this task, we employ the network generation framework that was recently proposed by Lancichinetti et al. [44]. This framework can generate a variety of networks by varying the node degree, community size, and number of inter-community edges. The networks produced by this framework are of acceptable quality as the authors have shown that their framework does a better job at revealing the differences between community mining results when compared to Modularity based network generators.

To specify the inter-community connectedness of the network, the framework provides a μ parameter. This parameter ranges from (0-1) and determines what fraction of a node's edges lead outside of its community. A low μ value indicates clear cut community boundaries, whereas a network with a high μ value has ambiguous community boundaries. For our experiment we generated 154 networks, with half containing 100 nodes, the other half containing 500 nodes. Of these networks, approximately 10% have μ=0.1, 10% have μ=0.2, 10% have μ=0.3 ... up to μ=0.9. We chose to include networks with high values of μ so that we can determine how the criteria respond to very noisy data. For each network, we also selected a random value for both the average degree and the minimum/maximum community sizes. Thus some of the networks were very dense, some had a few large communities,

and some had many small communities.

4.1 Community Mining Algorithms

To identify the communities in these networks we applied the following six community mining algorithms:

4.1.1 Fast Modularity

This is a fast implementation of an agglomerative community mining algorithm that maximizes Q-Modularity [14]. Due to the previously mentioned resolution limit of Modularity, this method is biased against small communities.

4.1.2 MaxMin Modularity

This is an agglomerative approach, introduced by Chen et al., which takes into account both the Modularity and the number of unrelated node pairs within the communities [11]. A node pair is unrelated if there is no edge linking the two nodes. The algorithm tries to maximize the Modularity score while minimizing the number of unrelated node pairs in a community. As a result, this method is also biased against small communities due to the use of Modularity [20].

4.1.3 Clique Percolation Method (CPM)

This is a clique-based method, introduced by Palla et al., where the communities are defined as the maximum union of adjacent k-cliques; where k is the number of nodes in the clique [56]. Due to the size of our networks, we have elected to use k=3.

4.1.4 Local M

This approach employs the M metric, proposed by Luo et al. [46], within Clauset's local framework [13] .The authors define the M metric as the number of internal edges divided by the number of external edges.

4.1.5 Local L

This approach employs the L metric, proposed by Chen et al. [12], within Clauset's local framework. The L metric considers both the ratio of internal to external edges and the density of the community.

4.1.6 Local TopLeaders

Local TopLeaders is a centroid based, graph-theoretic community mining approach [38]. It repeatedly explores the graph starting from the possible medoids/leaders and expands their communities using a modified breadth first search (BFS) method. This BFS considers the neighbourhood intersections of the nodes to compute distances. After expanding the followers/community of each leader, TopLeaders jumps to the next leader in the network to outline its community. Leaders are identified based on their degree centrality.

4.2 External Indices

If a criterion indicates a particular community mining result is better than another, how do we know that it chose the correct one? To answer this question we employ an external index that evaluates the correctness of a community mining result by computing its similarity to the ground truth. In the case where the membership of each community exactly matches that of the ground truth, the evaluation is trivial. However, when the memberships do not match, there are a variety of methods that focus on the number of false negatives, false positives, or both. We have selected three well-known, and one recently introduced, external index for our experiment:

4.2.1 Adjusted Rand Index (ARI)

Given a community membership X, and a ground truth G, this external index evaluates how similar X is to G by considering if pairs of nodes share the same community or not [32]. It is also adjusted to correct for chance so that the ARI of a randomly generated community membership is closer to zero. More formally:

$$ARI = \frac{(|N| * (a+d)) - M}{(|N|^2) - M)}$$
$$M = (a+b)(a+c)(c+d)(b+d)$$

where —N— is the number of nodes in the network and:

- a = number of node pairs where both nodes are in the same community in both X and G
- b = number of node pairs where both nodes are in the same community in G but not in X
- c = number of node pairs where both nodes are in the same community in X but not in G
- d = number of node pairs where both nodes are in different communities in both X and G

4.2.2 Jaccard Index

This index considers the similarity between community membership X, and ground truth G, by computing the ratio between the intersection and union of set(X) and set(G) [35]. These sets are constructed individually for both X and G, and are made up of node pairs containing nodes that belong to the same community. More formally:

$$Jaccard = \frac{set(X) \cap set(G)}{set(X) \cup set(G)}$$

where the intersection and union are computed on the node pairs.

4.2.3 Normalized Mutual Information (NMI)

This index measures the overlap between the communities in both X and G to determine how much information we can learn about one if we had the other [15]. More formally:

$$
\begin{aligned}
NMI &= \frac{2I(X,G)}{h(X)+h(G)} \\
I(X,G) &= \sum_{C_X \in k_X} \sum_{C_G \in k_G} \frac{o(C_X,C_G)}{|N|} \log\left(\frac{o(C_X,C_G)}{|C_X||C_G||N|}\right) \\
h(X) &= \sum_{C_X \in k_X} -1 * \frac{|C_X|}{|N|} \log\left(\frac{|C_X|}{|N|}\right) \\
h(G) &= \sum_{C_G \in k_G} -1 * \frac{|C_G|}{|N|} \log\left(\frac{|C_G|}{|N|}\right)
\end{aligned}
$$

where k_X is the set of communities in X, k_G is the set of community in G, and $o(C_X, C_G)$ is the overlap between communties C_X and C_G.

4.2.4 Alternative Normalized Mutual Information (ANMI)

This index was recently proposed by Lancichinetti et al. as a modified version of NMI that handles overlapping communities [43]. In practice it punishes poor community results quite heavily, and thus is not an exact duplicate of the original NMI. The specifics of this index are intricate and beyond the scope of this thesis.

Chapter 5

Experiment Methodology & Results

5.1 Correlation Analysis

The goal of our experiment is to determine the quality of each relative validity criteria and rank them accordingly. To do this we compute the scores for both the criteria and external indices for each community mining result for a particular network. A high quality criterion should express a high correlation between its scores and the scores of an external index. This would indicate that both the criteria and the index produced similar rankings and magnitudes for the community mining results.

For our experiment we ran each of the community mining algorithms on 154 generated networks and two real world networks (NCAA Football, and Zachary's Karate Club [66]). For each of these community mining results, we computed the score of each criteria with all possible combinations of distance and centroid methods. We did not include either the Betweenness centroid method or the Gamma criteria due to the excessive time-complexity of these methods.

We also computed all of the external indices for each of the community mining results. For each network, we then determined the Pearson correlation between the scores of the criteria and each of the external indices. The top 10 average correlations across all external indices are provided in Table 5.1. The top 10 correlations for each external index are shown in Tables 5.2, 5.3, 5.4, and 5.5.

We can see from the overall correlation scores that no criterion performs well across every external index. For example, Modularity is the top overall criterion,

Correl.	Criteria	Centroid	Distance
0.23845	Modularity	-	-
0.20713	BallAndHall	DV	SP
0.20538	WG	D	ARD
0.20337	BallAndHall	D	ARD
0.19712	BallAndHall	EC	SP
0.19669	WG	DV	SP
0.19588	BallAndHall	D	NOD
0.19565	WG	D	NOD
0.19505	covWG	DV	SP
0.19233	covWG	D	ARD

Table 5.1: The top 10 highest correlation scores, averaged over all external indices.

Correl.	Criteria	Centroid	Distance
0.67730	BallAndHall	D	NOD
0.67701	WG	D	NOD
0.66766	McClainAndRao	-	SP
0.66724	BallAndHall	DV	NOD
0.66525	covWG	D	NOD
0.66410	BallAndHall	EC	NOD
0.66242	WG	DV	NOD
0.66086	McClainAndRao	-	ARD
0.66055	WG	EC	NOD
0.65804	McClainAndRao	-	NOD

Table 5.2: The top 10 highest correlation scores with respect to ARI.

Correl.	Criteria	Centroid	Distance
0.40939	VarianceRatio	D	NOD
0.38886	VarianceRatio	DV	NOD
0.38127	VarianceRatio	EC	NOD
0.37758	VarianceRatio	EC	ARD
0.37692	$C/\sqrt{k}$	D	SP
0.35971	VarianceRatio	D	ARD
0.35916	$C/\sqrt{k}$	D	NOD
0.35910	VarianceRatio	DV	ARD
0.35898	VarianceRatio	EC	SP
0.35020	$C/\sqrt{k}$	EC	SP

Table 5.3: The top 10 highest correlation scores with respect to Jaccard

Correl.	Criteria	Centroid	Distance
0.31212	$C/\sqrt{k}$	D	SP
0.29937	VarianceRatio	EC	NOD
0.28880	VarianceRatio	EC	ARD
0.28365	VarianceRatio	DV	NOD
0.27786	VarianceRatio	D	NOD
0.27687	$C/\sqrt{k}$	EC	SP
0.27017	VarianceRatio	DV	ARD
0.23969	$C/\sqrt{k}$	D	NOD
0.23854	VarianceRatio	EC	SP
0.23817	VarianceRatio	D	ARD

Table 5.4: The top 10 highest correlation scores with respect to ANMI.

Correl.	Criteria	Centroid	Distance
0.58355	Silhouette	D	NOD
0.54647	Silhouette	EC	NOD
0.51248	Silhouette	DV	ARD
0.50597	Silhouette	DV	NOD
0.50320	Silhouette	EC	ARD
0.49629	Silhouette	EC	SP
0.48934	SSBOverSSW	D	ARD
0.48225	SSBOverSSW	EC	NOD
0.47612	SSBOverSSW	D	NOD
0.47541	SSBOverSSW	DV	NOD

Table 5.5: The top 10 highest correlation scores with respect to NMI.

yet it is not in the top 10 of any of the external indices. This exemplifies the fact that deciding which criteria performs the best depends on the external index used to evaluate the results. As a result, we cannot declare a clear winner with regards to correlation scores. In addition, although some of the criteria expressed higher levels of correlation, such as the top 10 Adjusted Rand Index (ARI) results, none of them expressed enough correlation such that we can rely on their evaluations as a substitute for a ground truth.

As an aside, we also notice some agreement between the rankings for the Jaccard index and for the Alternative Normalized Mutual Information index. This trend continues throughout the data, revealing a possible weak correlation between these two indices. Furthermore, we notice that the Pearson Correlation Distance (PCD) does not show up in any of the results. Upon further investigation we discovered that the permutations involving this distance metric performed the worst amongst all permutations - indicating that this distance metric is not well suited for this task. Whereas all three of the other distance methods appear to be interchangeable, as no single method dominates the rankings. Also, we chose to omit the permutations that used the Closeness centroid method, as they were almost perfect duplicates of the permutations that used the EstimatedCloseness centroid method. This represents that for our task, the EstimatedCloseness measure represents a sufficiently accurate estimate of Closeness centrality.

5.2 Indicator Quality Analysis

In addition to this correlation analysis, we also considered all possible pairs of community mining results within each network. For each pair we labeled the result with the higher external index score as the most correct one. We then determined which pair was given a higher score by each of the criteria. The indicator quality counter of a criterion is incremented by 1 if it correctly gave the higher score to the same community mining result as the external index did. If it did not, then its total error rate is incremented by the absolute value of the difference in the external index scores for each community mining result. The following is a brief example of this

evaluation:

	ARI	Silhouette	Dunn
MiningResult1	0.8	0.7	0.5
MiningResult2	0.5	0.6	0.9

MostCorrect(ARI) = MiningResult1

MostCorrect(Silhouette) = MiningResult1

MostCorrect(Dunn) = MiningResult2

We can see that Silhouette and ARI agree, therefore Silhouette's 'indicator quality' counter is incremented by 1. Whereas Dunn does not agree with ARI, therefore its error rate is incremented by 0.3, which is the difference in the ARI values (abs(0.8-0.5) = 0.3).

From this example, and the explanation above, we can see that a high quality criterion should have a high indicator quality counter, and a low error rate. There are 156 networks, and six community mining methods, resulting in 2340 pairwise comparisons ($156 * \binom{6}{2}$). We normalize each indicator quality score by the number of such comparisons to learn the percentage of times that the criteria selected the most correct answer.

The top 10 indicator quality scores, across all external indices, are shown in Table 5.6. The lowest 10 error rates are shown in Table 5.7. To produce a unified ranking, we multiply the error rate by the inverse of the indicator quality score; thus a low unified score is desired. The top 10 unified scores are presented in Table 5.8. In addition, in Tables 5.9, 5.10, 5.11, and 5.12 we show the top 10 unified scores for each external index to highlight the disagreements between them.

From these results we can see that BallAndHall, Modularity, WG, and covWG dominate the rankings for both indicator quality and error rate. Modularity appears to perform slightly better than the other four, but as was the case for the correlation scores, the exact ranking depends on which external index we consider. Unfortunately, once more we notice that none of the criteria perform well enough for their results to be considered as ground truths. Overall, none of the criteria selected the correct community mining result more often than 57% of the time.

Indicator	Criteria	Centroid	Distance
0.57160	Modularity	-	-
0.55943	BallAndHall	DV	SP
0.55922	WG	D	ARD
0.55911	BallAndHall	D	ARD
0.55879	WG	DV	SP
0.55599	covWG	DV	SP
0.55545	BallAndHall	D	NOD
0.55491	McClainAndRao	-	ARD
0.55276	WG	D	NOD
0.55071	covWG	D	ARD

Table 5.6: The top 10 highest indicator quality scores, summed over all external indices.

Error	Criteria	Centroid	Distance
57.74007	Modularity	-	-
60.32266	WG	DV	SP
60.73679	BallAndHall	DV	SP
61.89298	covWG	DV	SP
67.43488	BallAndHall	D	NOD
68.14982	BallAndHall	D	ARD
68.50125	BallAndHall	EC	SP
68.58471	WG	D	NOD
69.44110	WG	D	ARD
70.22536	WG	EC	SP

Table 5.7: The top 10 lowest error rates, summed over all external indices.

Unified	Criteria	Centroid	Distance
0.04350	Modularity	-	-
0.04649	WG	DV	SP
0.04676	BallAndHall	DV	SP
0.04794	covWG	DV	SP
0.05229	BallAndHall	D	NOD
0.05249	BallAndHall	D	ARD
0.05344	WG	D	NOD
0.05348	WG	D	ARD
0.05433	BallAndHall	EC	SP
0.05596	WG	EC	SP

Table 5.8: The top 10 unified scores, summed over all external indices.

Unified	Criteria	Centroid	Distance
0.02264	McClainAndRao	-	SP
0.02426	McClainAndRao	-	NOD
0.02474	McClainAndRao	-	ARD
0.02713	WG	DV	SP
0.02742	BallAndHall	DV	SP
0.02780	covWG	DV	SP
0.02910	BallAndHall	EC	ARD
0.02925	Modularity	-	-
0.02995	BallAndHall	D	NOD
0.03169	WG	D	NOD

Table 5.9: The top 10 unified scores with respect to ARI.

Unified	Criteria	Centroid	Distance
0.04994	VarianceRatio	D	NOD
0.05434	VarianceRatio	EC	ARD
0.05661	VarianceRatio	EC	NOD
0.05662	VarianceRatio	DV	NOD
0.05788	$C/\sqrt{k}$	D	SP
0.05814	VarianceRatio	D	ARD
0.06030	VarianceRatio	DV	ARD
0.06089	Modularity	-	-
0.06392	$C/\sqrt{k}$	D	NOD
0.06680	PBM	D	NOD

Table 5.10: The top 10 unified scores with respect to Jaccard.

Unified	Criteria	Centroid	Distance
0.03803	Modularity	-	-
0.04535	WG	DV	SP
0.04605	BallAndHall	DV	SP
0.04856	covWG	DV	SP
0.05007	VarianceRatio	D	NOD
0.05293	VarianceRatio	EC	ARD
0.05395	WG	D	ARD
0.05510	BallAndHall	D	ARD
0.05615	VarianceRatio	D	ARD
0.05690	VarianceRatio	EC	NOD

Table 5.11: The top 10 unified scores with respect to ANMI.

Unified	Criteria	Centroid	Distance
0.04070	Silhouette	D	NOD
0.04278	McClainAndRao	-	ARD
0.04517	Silhouette	DV	ARD
0.04539	McClainAndRao	-	SP
0.04626	Modularity	-	-
0.04742	McClainAndRao	-	NOD
0.04911	WG	DV	SP
0.04938	BallAndHall	DV	SP
0.04989	covWG	DV	SP
0.05152	BallAndHall	D	ARD

Table 5.12: The top 10 unified scores with respect to NMI

5.3 Difficulty Analysis

For our final piece of analysis we wanted to group the networks based on some quantitative estimate of how hard it is to find the correct communities in a network. Unfortunately we could not find a suitable estimator. As an alternative, we grouped the networks by looking at the average score of their community mining results, across all of the external indices. Empirically it follows that networks with a low average score are more difficult than networks with a higher average. The label for each network depended on which range its average score fell into, as follows:

[0-0.29]: assigned a 'hard' label. There were 101 such networks.

[0.3-0.59]: assigned a 'medium' label. There were 39 such networks.

[0.6-1.0]: assigned an 'easy' label. There were 16 such networks.

We should note that many of the networks within the 'medium' and 'hard' ranges had scores well above 0.6 for some particular external index. However, to avoid being biased towards a particular external index, we only considered the average across all indices.

The top 10 correlations for the relative validity criteria for each level of difficulty are shown in Tables 5.13, 5.15, and 5.17. Furthermore the top 10 unified scores for each level of difficulty are shown in Tables 5.14, 5.16, and 5.18.

Based on our groupings we can see that both Modularity and BallAndHall dominate the rankings for 'easy' networks, with each criterion expressing very accurate

Correl.	Criteria	Centroid	Distance
0.91151	Modularity	-	-
0.85435	BallAndHall	DV	SP
0.82511	WG	DV	SP
0.81766	covWG	DV	SP
0.80033	BallAndHall	D	NOD
0.79927	WG	D	NOD
0.78670	BallAndHall	EC	SP
0.77330	BallAndHall	D	ARD
0.76513	WG	D	ARD
0.73923	BallAndHall	D	SP

Table 5.13: The top 10 correlation scores, across all external indices, for 'Easy' networks.

Unified	Criteria	Centroid	Distance
0.01309	BallAndHall	DV	SP
0.01313	Modularity	-	-
0.01421	WG	DV	SP
0.01829	covWG	DV	SP
0.03574	BallAndHall	D	NOD
0.03596	WG	D	NOD
0.04891	BallAndHall	EC	SP
0.05371	WG	EC	SP
0.05686	BallAndHall	EC	ARD
0.05817	BallAndHall	D	ARD

Table 5.14: The top 10 unified scores, across all external indices, for 'Easy' networks.

Correl.	Criteria	Centroid	Distance
0.80875	Modularity	-	-
0.73343	WG	D	ARD
0.73158	BallAndHall	DV	SP
0.72524	BallAndHall	D	ARD
0.72417	BallAndHall	D	SP
0.72375	BallAndHall	EC	SP
0.70535	WG	D	SP
0.70048	WG	DV	SP
0.69998	covWG	EC	SP
0.69980	WG	EC	SP

Table 5.15: The top 10 correlation scores, across all external indices, for 'Medium' networks.

Unified	Criteria	Centroid	Distance
0.02530	Modularity	-	-
0.03252	BallAndHall	D	ARD
0.03289	WG	D	ARD
0.03540	BallAndHall	D	SP
0.03631	WG	DV	SP
0.03688	BallAndHall	DV	SP
0.03831	WG	D	SP
0.03843	BallAndHall	EC	SP
0.03848	covWG	DV	SP
0.03886	covWG	D	SP

Table 5.16: The top 10 unified scores, across all external indices, for 'Medium' networks.

Correl.	Criteria	Centroid	Distance
0.13261	SSBOverSSW	D	SP
0.13023	SSBOverSSW	DV	SP
0.12719	Silhouette	D	ARD
0.12557	SSBOverSSW	EC	SP
0.11023	SSBOverSSW	DV	ARD
0.10888	CIndex	-	SP
0.08989	SSBOverSSW	EC	NOD
0.08771	SSBOverSSW	EC	ARD
0.07938	SSBOverSSW	D	NOD
0.07920	SSBOverSSW	D	ARD

Table 5.17: The top 10 correlation scores, across all external indices, for 'Hard' networks.

results. For networks that fall within this category we believe that either of these criteria could be used as a ground-truth substitute. It should be noted, however, that very few real-world networks have such clearly defined community boundaries.

In the 'Medium' networks we see a clear domination by Modularity, with strong performances from both BallAndHall and WG. The correlations indicate a reasonable level of accuracy, such that these criteria could be used as a ground-truth substitute if required. In addition, 'Medium' networks are most likely more common in the real world as they contain noisy, yet defined, community boundaries. Yet even in these realistic networks it is clear that these criteria are not suitable replacements for a ground truth.

Unified	Criteria	Centroid	Distance
0.04846	SSBOverSSW	D	ARD
0.04992	SSBOverSSW	EC	NOD
0.05046	Silhouette	D	NOD
0.05065	SSBOverSSW	EC	ARD
0.05067	Silhouette	EC	NOD
0.05107	SSBOverSSW	DV	NOD
0.05140	SSBOverSSW	D	NOD
0.05204	SSBOverSSW	DV	ARD
0.05370	McClainAndRao	-	SP
0.05474	SSBOverSSW	EC	SP

Table 5.18: The top 10 unified scores, across all external indices, for 'Hard' networks.

Finally, in 'Large' networks we see both Silhoutte and SSBOverSSW dominating the rankings. Unfortunately their correlation scores are very low, indicating that none of the criteria can be relied upon for such challenging networks. However, these rankings only tell the overall story across all external indices. There are in fact some criteria which perform quite well for some external indices, achieving indicator rates above 0.7 for these 'Hard' networks. We have included these rankings, and more, in the Appendix section.

In summary, we have shown that Modularity, BallAndHall, WG, and covWG, are the top performing criteria when considering all of the external indices. The top criterion for each particular external index can be determined by studying the tables of results for that index. Furthermore we've shown that although Modularity dominates 'Easy' and 'Medium' networks, other criteria become more accurate when considering more difficult networks. Overall, however, none of the metrics are accurate substitutes for a ground truth.

5.4 Appendix

Indicator	Criteria	Centroid	Distance
0.75624	McClainAndRao	-	SP
0.75065	McClainAndRao	-	ARD
0.72868	McClainAndRao	-	NOD
0.71921	Silhouette	D	SP
0.69208	Silhouette	D	NOD
0.68777	CIndex	DV	ARD
0.67743	BallAndHall	DV	SP
0.67657	WG	DV	SP
0.67485	covWG	DV	SP
0.67442	BallAndHall	D	ARD

Table 5.19: The top 10 indicator scores with respect to ARI.

Error	Criteria	Centroid	Distance
39.74751	McClainAndRao	-	SP
41.04314	McClainAndRao	-	NOD
42.61772	WG	DV	SP
43.11425	McClainAndRao	-	ARD
43.12828	BallAndHall	DV	SP
43.56082	covWG	DV	SP
45.25556	BallAndHall	EC	ARD
45.81193	Modularity	DV	SP
46.30270	BallAndHall	D	NOD
48.80712	WG	D	NOD

Table 5.20: The top 10 error scores with respect to ARI.

Indicator	Criteria	Centroid	Distance
0.69423	PBM	D	NOD
0.69208	PBM	DV	NOD
0.69078	PBM	EC	NOD
0.68691	VarianceRatio	EC	ARD
0.68648	$C/\sqrt{k}$	D	SP
0.68562	VarianceRatio	EC	NOD
0.68562	VarianceRatio	D	NOD
0.68303	VarianceRatio	DV	NOD
0.68260	PBM	D	ARD
0.67959	$C/\sqrt{k}$	D	NOD

Table 5.21: The top 10 indicator scores with respect to Jaccard.

Error	Criteria	Centroid	Distance
79.50711	VarianceRatio	D	NOD
82.45160	Modularity	-	-
86.67675	VarianceRatio	EC	ARD
88.99288	WG	DV	SP
89.13855	BallAndHall	DV	SP
89.53534	VarianceRatio	D	ARD
89.79338	VarianceRatio	DV	NOD
90.12495	VarianceRatio	EC	NOD
90.94655	covWG	DV	SP
92.26848	$C/\sqrt{k}$	D	SP

Table 5.22: The top 10 error scores with respect to Jaccard.

Indicator	Criteria	Centroid	Distance
0.72567	Silhouette	D	NOD
0.70198	Silhouette	EC	NOD
0.68906	McClainAndRao	-	SP
0.68691	Silhouette	DV	NOD
0.68346	SSBOverSSW	EC	ARD
0.68303	SSBOverSSW	D	ARD
0.68174	McClainAndRao	-	ARD
0.67959	Silhouette	DV	ARD
0.67959	Silhouette	EC	ARD
0.67916	SSBOverSSW	DV	NOD

Table 5.23: The top 10 indicator scores with respect to NMI.

Error	Criteria	Centroid	Distance
66.60758	Modularity	-	-
67.72124	McClainAndRao	-	ARD
68.57161	Silhouette	D	NOD
69.00123	WG	DV	SP
69.28181	BallAndHall	DV	SP
69.89901	covWG	DV	SP
71.28420	Silhouette	DV	ARD
72.18063	McClainAndRao	-	NOD
72.61985	McClainAndRao	-	SP
72.84549	BallAndHall	D	ARD

Table 5.24: The top 10 error scores with respect to NMI.

Indicator	Criteria	Centroid	Distance
0.52929	$C/\sqrt{k}$	D	SP
0.51335	$C/\sqrt{k}$	D	NOD
0.51034	VarianceRatio	EC	ARD
0.50689	$C/\sqrt{k}$	EC	NOD
0.50646	VarianceRatio	EC	NOD
0.50388	VarianceRatio	D	NOD
0.50345	PBM	D	NOD
0.50258	PBM	DV	NOD
0.50000	PBM	EC	NOD
0.49957	PBM	EC	ARD

Table 5.25: The top 10 indicator scores with respect to ANMI.

Error	Criteria	Centroid	Distance
36.08918	Modularity	-	-
40.67880	WG	DV	SP
41.39852	BallAndHall	DV	SP
43.16554	covWG	DV	SP
49.41485	WG	D	ARD
49.84470	BallAndHall	EC	SP
49.86442	BallAndHall	D	ARD
52.14300	BallAndHall	D	NOD
52.45674	WG	EC	SP
52.66312	WG	D	NOD

Table 5.26: The top 10 error scores with respect to ANMI.

Part III

Using Triads to Identify Local Community Structure in Social Networks

Chapter 6

Motivation

The challenge of exactly how to detect communities has been a central problem studied in social network analysis in the past two decades. This research has been fueled by a demand from numerous fields that hope to use community mining algorithms to discover protein communities in biology, crime factions in criminology, and groups of friends in telecommunication networks. To this end, researchers have proposed a variety of techniques that discover communities by considering the entire network structure, that is, they require global knowledge of the network [53, 51, **?**]. Unfortunately, they realized that these global techniques do not scale well when considering extremely large information networks, such as Facebook or the World Wide Web, which are becoming increasingly popular and contain hundreds of millions or billions of nodes [53].

To remedy this problem, researchers have recently proposed local methods that detect communities by only considering local information and therefore are not sensitive to the size of the network [13, 12, 56]. These local methods generally require some metric that determines the relative quality of a community, and indeed, many such metrics have been proposed. However, the existing metrics often suffer from poor outlier detection [13, 46] and the discovery of incorrect communities in simple ground truth networks [12, 60].

In this thesis we aim to solve both of these problems by presenting our T metric, which defines the relative quality of a community by considering the number of internal and external triads it contains. We apply our T metric within a modified version of Clauset's local framework to greedily discover communities while

achieving more accurate outlier and hub detection when compared to previous approaches [13]. We also show that our framework, combined with the T metric, leads to increased accuracy on a variety of ground truth networks when compared to the existing techniques.

Chapter 7

Related Work

In recent years researchers have proposed a variety of community mining techniques that employ either the divisive or agglomerative framework to detect communities using global information [**?**]. The most well-known of these approaches is Newman's Q-Modularity metric which considers the number of edges within a community minus the expected number of such edges in a random network [51]. However, Fortunato and Barthélemy have shown that Modularity-based metrics suffer from a resolution limit, in that they cannot detect communities smaller than some threshold [20]. Furthermore, it is unclear how to detect outlier nodes with Modularity-based methods.

In addition, researchers have realized that it is computationally intractable to consider global information for many of the large scale networks that they wish to analyze [53]. To address this concern, Clauset [13] introduced his local community mining framework that explores the network through local expansion and thus is not sensitive to the network size. His method requires a metric to determine the quality of each discovered community and a variety of such metrics have been proposed, including Clauset's own R metric, the M metric from Luo et al., and the L metric from Chen et al. [13, 46, 12]. All of these metrics evaluate a community by considering how edges are distributed within the community relative to outside of the community. These metrics, however, fail to accurately identify outliers and achieve low scores on many ground truth networks, as shown in our evaluation section.

Palla et al. [56] have also proposed their Clique Percolation Method which identifies communities by rolling a k-clique around the network until it is unable to

reach any unexplored nodes. The nodes covered while rolling are considered the discovered community and then the process continues on a different section of the network. Although their algorithm benefits from being local and intuitive, it is also very sensitive to the parameter choice for k, which determines the size of the clique and thus the algorithm can be difficult to apply in practice.

Other researchers have proposed using information theory to discover communities, such as in the method introduced by Rosvall and Bergstrom [60]. In their Infomap algorithm, Rosvall and Bergstrom view community mining as a code compression problem, where the goal is to describe a random walk of the network using as few bits as possible. The application of community structures allows the algorithm to assign the same unique code name to multiple nodes, provided they are in different communities. Thus a good community partitioning, where the random walk is more likely to stay within the community than leave it, can greatly reduce the number of unique codes required. It follows from their intuition, then, that finding the best compression, through simulated annealing, will also yield the best community partitioning. Their algorithm produced excellent results on synthetic networks, as shown by Lancichinetti and Fortunato [42], but did not perform as well against real-world networks, as shown in our evaluation.

We find that our approach lies somewhere between Clauset's Local Framework and the Clique Percolation Method, in that our T metric favours communities that contain triads (cliques of size 3), but it discovers these communities through local expansion.

Chapter 8

Our Approach

Our approach is a two stage algorithm that first detects communities by applying our T metric within the local community framework and then employs an additional stage to identify outliers/hubs in the discovered community.

The local community framework we apply in this thesis has been adapted from Clauset's local framework [13] and can be summarized as follows. First, we initialize the community with a single node and place all of its neighbours into the shell set. Then, for each iteration, we greedily select the node from the shell set that, when included in the community, maximizes the T metric. We add this selected node to the community and all of its neighbours to the shell set. This process continues until there are no nodes in the shell set which would further maximize the T metric. At this point, a community is discovered and the algorithm restarts on another node in the network. In order to prevent overlapping communities we ensure that all nodes which are assigned to one community cannot belong to any other community. A depiction of the shell set is shown in Figure 8.1. Our major deviation from Clauset's original framework is that we do not keep track of the boundary set. Also, as we will explain later, we have added an additional stage to the framework that detects both outliers and hubs.

It is important to note that the selection of the starting node for the local framework can have a dramatic effect on the accuracy of the algorithm. In particular, our evaluation in Section 4 shows that randomly selecting the starting node can result in very poor community structure and accuracy. We hypothesize that a good starting node will have a high degree because it allows for a large neighbourhood to be con-

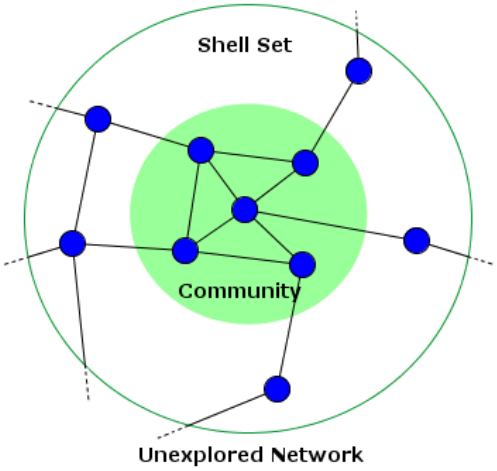


Figure 8.1: A depiction of our local framework.

sidered in the first iteration of the framework. However without global knowledge it is impossible to select the optimal starting node, as we do not know the degree of every node. To remedy this we propose a local approach, that first chooses a node at random and then explores its immediate neighbourhood and selects the node with the highest degree. This node becomes our starting node. Note that we have used the degree metric, instead of PageRank or Betweenness, because it can be computed locally without consulting the entire network. We briefly considered other metrics, such as Estimated Closeness [18], but they did not outperform the degree metric in our test cases.

8.1 Local Community Metric T

Given the local community framework we can see that the role of the T metric is to determine whether or not a node should be included in the community. Intuitively, our metric favours nodes that form many triads with nodes within the community and few triads with nodes outside of the community. We define these quantities as T_{in} and T_{ex}, respectively. We define a triad as a collection of three nodes that are fully connected, aka, a 3-node clique. Our intuition is that all members of a triad are tightly bonded together and thus are more likely to belong to the same community.

More formally, we present our T metric as:

$$T = T_{in} * T_{\mathit{diff}}$$

Where

$$T_{\mathit{diff}} = \begin{cases} T_{in} - T_{ex} & \text{if } T_{in} \geq T_{ex} \\ 0 & \text{otherwise} \end{cases}$$

$$T_{in} = \frac{1}{6} * \sum_{i \in C, j \in C, k \in C} A_{i,j} * A_{j,k} * A_{i,k}$$

$$T_{ex} = \frac{1}{2} * \sum_{i \in C, j \in S, k \in S} A_{i,j} * A_{i,k} * A_{j,k}$$

Where C is the set of nodes in the community, S is the set of nodes in the shell set, and A is the adjacency matrix such that $A_{i,j}$ is 1 if nodes i and j share an edge. We divide the T_{in} score by 6 to prevent double counting all permutations of the same triad, for example, 'ABC', 'ACB', 'BCA', 'BAC', 'CAB', and 'CBA' all refer to the same triad between nodes A, B, and C. For T_{ex} we only divide by 2 because the limitation that $i \in C$ reduces the number of permutations. Our intuition with the T formula is that we wish emphasize the internal triads more than the external because we believe that a community more strongly defined by what is inside of it, rather than outside.

We have bounded T_{diff}, and thus T, to be non-negative because all of the nodes in the initial stages of the community will belong to more external triads than internal ones. If left unbounded, this would result in a negative T_{diff} score that would penalize well connected nodes; yet these are the very nodes that we believe should be included first. Thus, we set the T_{diff} score to zero in these cases and let the tie-breaking step determine the best node.

This tie-breaking step is a critical part of the metric because there are many cases where multiple nodes result in the same T score, yet are qualitatively different. For example, consider a node X that when included in the community has a T_{in} score of 49, T_{ex} score of 48, and thus a T score of 49. Also consider a node Y that has a T_{in} score of 7, T_{ex} score of 0, and thus also a T score of 49. Clearly, node Y is a better choice to include in the community because it directly contributes to the

internal score without a negative influence on the external score. We capture this intuition by always selecting the node with the lowest T_{ex} score in the event of a tie.

It is important to note that we are not considering triads that have two nodes in the community and one node in the shell set. This is because such a triad could be classified as either external or internal depending on whether the target node is chosen to be included in the community, or placed back into the shell set. Thus it does not make intuitive sense to assign this triad to either set.

Furthermore we are aware that our metric is dissimilar from many of the existing approaches in that it does not try to maximize a ratio of internal to external scores. This is because we feel that the difficulty associated with dividing by zero results in a biased metric that favours nodes with no external relations. For example, consider a metric that counts the number of edges. Also consider two nodes in the shell set: one with 2 internal edges, 0 external edges, and one with 10 internal edges, 1 external edge. If the ratio of internal to external edges of the community is 100:10, then the first node will be included, but the second will not. We find this approach to be counter-intuitive, especially given that as the ratio score of the community increases, so does the idiosyncrasy of such examples.

8.2 Incremental Formula

Although the formulae given above are relatively simple, it would be computationally demanding to count the number of triads every time a node is considered, thus we also present an incremental formula for computing the T_{in} and T_{ex} scores based on the previous scores. More formally:

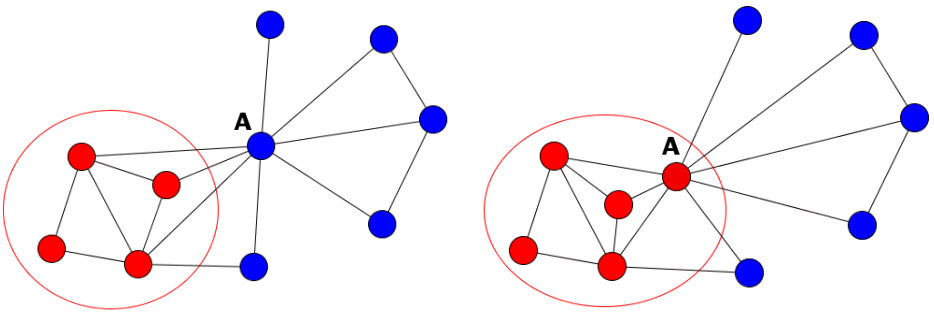
$$T_{in}\prime = T_{in} + \frac{1}{2} \sum_{\substack{i \in N(X) \\ j \in N(X) \\ i \neq j}} A_{j,i} * C_i * C_j$$

$$\begin{aligned} T_{ex}\prime = T_{ex} \\ &+ \frac{1}{2} \sum_{\substack{i \in N(X) \\ j \in N(X) \\ i \neq j}} A_{j,i} * (1 - C_i) * (1 - C_j) \\ &- \sum_{\substack{i \in N(X) \\ j \in N(X) \\ i \neq j}} A_{j,i} * (1 - C_i) * C_j \end{aligned}$$

Where T_{in} and T_{ex} are the scores before including node X in the community, $N(X)$ is the neighbourhood of node X, $A_{i,j}$ is the adjacency matrix, and C_n is 1 if n is in the community, 0 otherwise. Here, the last term in $T_{ex}\prime$ represents the number of triads that contain one node in the community and one node outside of the community. These triads are discounted because they were considered external triads prior to node X being included in the community, but now are considered uncounted triads. Note that we divide the second term in $T_{ex}\prime$ by 2 to avoid double counting both permutations of the same triad. A visual example of the incremental formula can be seen in Figure 8.2.

8.3 Outlier and Hub Detection

Although our T metric is used to identify communities, it does not directly solve the problems of pruning outliers from these communities or detecting hubs. The notion of an outlier can be summarized as a node that is weakly connected to the community but does not belong to any other community. Whereas a hub refers to a node that is strongly connected to many communities, without truly belonging to any individual community. To identify such nodes we have added an additional stage to the local framework that further processes each community after it has been



(a) Before including node A. T_{in} = 2, T_{ex} = 1.

(b) After including node A. $T_{in}\prime$ = 2 + 2 = 4, $T_{ex}\prime$ = 1 + 2 - 1 = 2.

Figure 8.2: An example of the incremental T calculation. Nodes within the circle are part of the community.

discovered. In particular, we iterate through the entire community and record the number of internal triads that each node belongs to. We then compute the average, T_{inAvg}, and standard deviation, T_{inStd}, of this score.

While iterating through the community we label a node as a hub if it participates in more external than internal triads; which follows from the observation that this node may belong to many other communities. However, it is not sufficient to detect a hub by only considering a single community. Thus, we allow nodes with the 'hub' label to join more than one community. This way if two or more communities label the same node as a hub, then it must be a true hub. On the other hand, if only a single community labels it as a hub then, by definition, it cannot be a hub and we remove its label.

To detect outliers we rely on the statistical distribution of the internal triads in the community. More specifically, a node is an outlier if it satisfies the following criteria:

$$T_{in}(X) < \lfloor T_{inAvg} - T_{inStd} \rfloor$$
$$T_{ex}(X) = 0$$

Where $T_{in}(X)$ is the number of internal triads that node X participates in and likewise for $T_{ex}(X)$. We believe that this definition best captures the intuitive un-

derstanding of an outlier, in that any node participating in significantly fewer triads than the average must be a weak member of that community. We have opted to use only one standard deviation based on our empirical analysis. We should point out that there are a variety of well-known statistical approaches to determine outliers, such as those proposed by Chauvenet, Grubbs, or Peirce [9, 28, 57]. Unfortunately we could not apply these methods as their assumption that the data is distributed normally does not hold in our scale-free social networks.

8.4 Time Complexity

Generally, it is challenging to capture the real-world time complexity of local methods because they are often bottlenecked by the costly data retrieval of node and edge structure from the large scale network. However, in terms of theoretical complexity, we see that the T metric must consider every node in the shell at each iteration and, for each node, it must count the number of triads that node is involved in. To count the triads we must iterate through the nodes neighbours, checking if they belong to the community and whether they share a neighbour with the node. The expected number of such neighbours is equal to the average degree of the network. Thus, the complexity of T can be formalized as $O(avg(|C|) * k^2)$, where k is the average degree of the network and $avg(|C|)$ is the average community size.

Chapter 9

Evaluation and Results

To rigorously evaluate our proposed framework we have compared it to a variety of popular community mining algorithms on a series of well-known ground truth networks. We have employed the Adjusted Rand Index (ARI) to compute a quantitative score that indicates how closely the results returned by each algorithm match that of the ground truth. More specifically, this index compares two sets of results and returns a score that ranges from 0, which indicates a completely random match, and 1, which indicates a perfect match.

We performed an evaluation against all known ground-truth networks, which are summarized in Table 9.1.

Ground Truth Network	Num. of Nodes	Num. of Edges	Num. of Coms.
Karate Club [66]	34	78	2
Strike [49]	24	38	3
Political Blogs [1]	1224	19087	2
Political Books [39]	105	441	3
Mexican Politics [26]	35	117	2
NCAA Football [27]	180	788	11 + outliers + hubs

Table 9.1: An overview of the ground truth networks used in our evaluation.

Of particular interest to us are the NCAA Football network, which contains many small communities and the Political Blogs network, which contains over 1000 nodes in two very large communities. We expect that many algorithms will have difficulty capturing both the small and large scale communities. Furthermore, we note that the Strike and Karate networks each contain one node that shares a single edge with both ground truth communities. Thus, using only the information in the

	MaxMin	CPM	Local L	Local R	Local M	Local T	Infomap
Karate Club	**1**	0.15	0.32	0.52	0.47	**1** / 0.9	0.7
Strike	**1**	0.36	0.37	0.71	0.76	**1** / 1	0.8
Political Blogs	-	-	0.06	0.62	0.66	**0.88** / 0.65	0.84
Political Books	0.64	0.63	0.22	0.55	0.57	**0.66** / 0.57	0.65
Mexican Politics	**0.36**	0.14	0.09	0.19	0.3	0	0.17
NCAA Football	0.15	0.983	0.96	0.28	0.28	**0.996** / 0.94	0.31

Table 9.2: Evaluation results. For the X/Y cells, X indicates the average score when selecting the starting nodes with the maximum local degree, and Y indicates the average score when randomly selecting the starting nodes. A dash indicates that the algorithm did not complete when processing the network.

	MaxMin	CPM	Local L	Local R	Local M	Local T	Infomap
Karate Club	2	3	6	3	3	2	3
Strike	3	6	5	5	3	3	4
Political Blogs	-	-	94	20	66	3	46
Political Books	2	4	10	7	4	3	5
Mexican Politics	3	1	3	3	2	1	4
NCAA Football	5	12	12	13	11	11	12

Table 9.3: The number of communities detected by each algorithm.

	MaxMin	CPM	Local L	Local M	Local T	Infomap
Karate Club	0.1	0.1	0.06	0.04	0.051	0.066
NCAA Football	0.55	0.28	0.26	0.23	0.12	0.12
Political Blogs	-	-	294.9	76	789.4	1.4
Political Books	0.27	0.24	0.35	0.15	0.07	0.08

Table 9.4: The runtime for each algorithm, measured in seconds. We have omitted the R metric because it is similar to the M metric from an efficiency standpoint.

network it is impossible to assign these nodes to the correct community every time. To prevent 'lucky' selections from biasing the results we have allowed these two nodes to belong to either of the communities without any penalty to the Adjusted Rand Index (ARI) score. The algorithms we compare our T metric against include:

MaxMin Modularity

This is an agglomerative algorithm proposed by Chen et al. [11] as an improvement over Newman's Q-Modularity based approach in that it also considers the number of unrelated node pairs within the community. This method requires global information about the network and contains no outlier detection. Note that we do not compare out method against Q-Modularity-

because Chen et al. have shown that their MaxMin algorithm outperforms Q-Modularity in previous evaluations [11].

Clique Percolation Modularity (CPM)

Please see the section on Related Work. For our evaluation we have selected the best result between the parameter value of K = 3, 4, and 5.

Local L

This is a local algorithm proposed by Chen et al. [12] that employs Clauset's local framework and the L metric to discover communities by maximizing the ratio of internal average degree over external average degree.

Local R

This is the original local algorithm proposed by Clauset that tries to maximize the number of edges leading from the boundary set of a community to its core and minimize the number of external edges [13]. This method contains no outlier detection.

Local M

This is also a local algorithm proposed by Luo et al. [46] that employs Clauset's framework and the M metric to discover communities by maximizing the ratio of internal edges over external edges. This method contains no outlier detection.

Infomap

Please see the section on Related Work. In a previous evaluation by Lancichinetti and Fortunato [42], this method outperformed many other algorithms on a set of synthetic networks. This method contains no outlier detection.

When evaluating our T metric we also want to determine what the optimal strategy is for selecting a starting node in the local framework. Thus we present two evaluations. In the first, our framework selects the starting node by exploring the local neighbourhood of a randomly selected node and choosing the one with highest degree. In the second, our framework simply selects a starting node at random.

To mitigate the effects of this randomness we have run each local algorithm ten times and reported the average score of these runs. We hope to show our proposed approach for selecting the starting node is significantly better than the random approach.

The results of our evaluation are summarized in Table 9.2, which contains the Adjusted Rand Index (ARI) scores, and Table 9.3, which contains the number of detected communities. The Normalized Mutual Information (NMI) scores do not differ in any significant way and thus we have not included them here.

As we can see in Table 9.2, our T metric matches or outperforms the existing algorithms on nearly every ground truth network, with the exception of the Mexican Politics network. In fact we notice that all of the algorithms perform very poorly on this network and upon further visual inspection it is unclear if there is any discernible community structure within the network. We have provided a sample visualization of the Mexican Politics network in Figure 9.1.

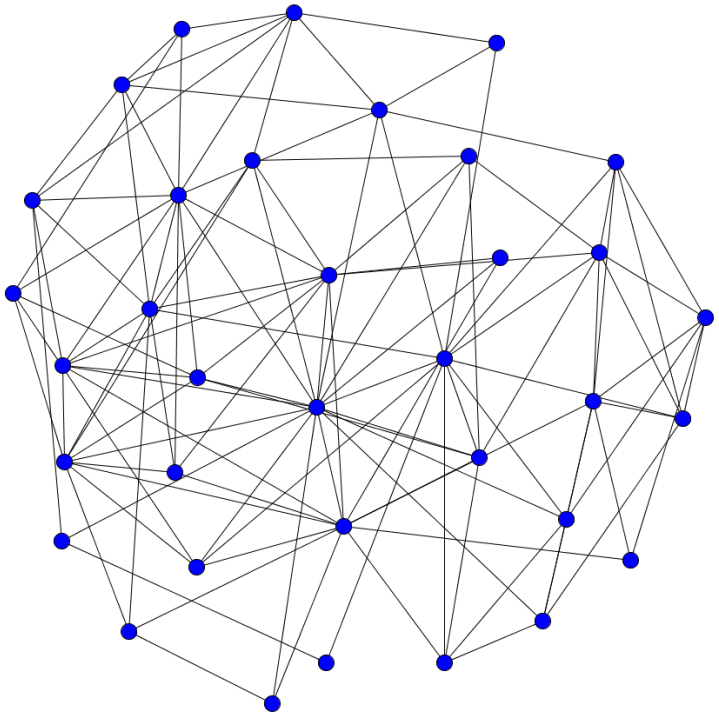


Figure 9.1: A visualization of the Mexican Politics network that reveals no obvious community structure.

Furthermore, in Table 9.3 we notice that our algorithm is the only method which

identifies the correct number of communities for a majority of the ground truth networks. We feel that this is an important evaluation tool in that ARI score can often be misleading when we don't consider the number of communities. This is exemplified by the fact that both the M metric and Infomap achieved a reasonably high ARI score in the Political Blogs network, even though they detected a multitude of extraneous communities. Additionally, we can see that our outlier detection method performed exceptionally well given that it accurately identified all of the outliers and all but one of the hubs in the NCAA Football Network. More importantly, contrary to Local L, our method did not identify any outliers or hubs in the other outlier-free networks.

Finally, our method of selecting the starting node appears to be somewhat better than the random approach when applied to our T metric. We can see noticeable improvements when using our approach on denser networks such as Political Blogs and Political Books. In addition to this evaluation we have also applied the L metric within our proposed framework to determine if it is our framework that provides the increased accuracy, or if it is the T metric itself. We hypothesized that perhaps our outlier and hub detection stage was responsible for our excellent results. This was not the case, and our results, which we do not present here for the sake of brevity, indicate that the L metric performs very poorly in our framework. Thus, we are more confident in claiming that the performance of our algorithm is largely attributable to our T metric.

As shown in Table 9.4, the increased accuracy of our algorithm comes at the cost of a longer runtime on networks with a high average degree, such as the Political Blogs network. To further explore this issue we have computed the runtime of our algorithm on a network of 1000 nodes and varied the average degree. The resulting chart is shown in Figure 9.2. It is clear that the time complexity of our approach can become prohibitively expensive for networks with very high average degree. We hope that future research into the efficiency of this approach will result in a clever heuristic for approximating the T score or improving upon its incremental computation for such networks. Alternatively, such research may lead to a suitable data structure that can significantly speed up the runtime, similar to what

FastModularity did for the Modularity based approaches [51].

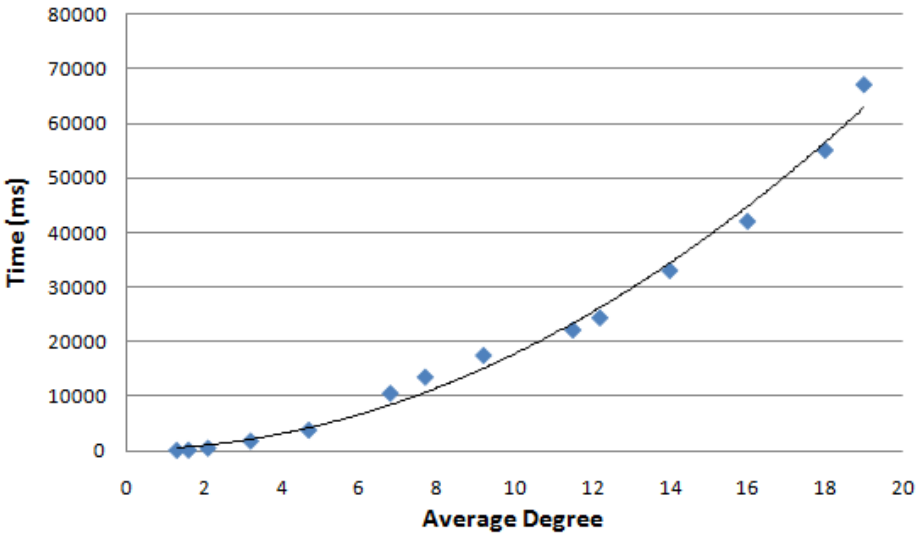


Figure 9.2: A graph of runtime versus average degree, for a 1000 node network.

Part IV

Community Centric Network Layouts

Chapter 10

Motivation

In previous parts we have seen a variety of community mining methods and metrics that have all claimed to accurately identify communities in social networks. However, there has been very little effort spent on evaluating the results of these algorithms in either a human-assisted or an automated sense. This is exacerbated by a lack of visualization methods that showcase either the structure of the communities or the relationships between them. Without these visuals clues it can be difficult for researchers to apply any intuition when evaluating their algorithms. Furthermore, such visualizations are beneficial for non-research users, who do not wish to scour through complex tables and matrices to get a big-picture view of the discovered communities.

In this part we address this issue by presenting two different layout algorithms. Our algorithms aim to generate aesthetically pleasing graph layouts which highlight the structure of each community and the relationships between the communities. We believe that our layouts allow a user to more easily discern the difference between a good community mining algorithm and poor one, when compared to the existing generalized layouts. Also, our layouts more accurately depict the relationships between the communities, allowing researchers and decision makers to benefit from this previously difficult to obtain information.

In order to accomplish these goals we first propose a number of modifications to the existing Fruchterman-Reingold (FR) algorithm [24], including support for vertices with a non-zero radius and the removal of visualization boundaries. We then introduce our first community centric layout algorithm, called COMmunity

Boundaries (COMB), which includes a two-stage process of generating representative vertices and employing circular bounding boxes to ensure the communities remain intact and isolated. We then present our second algorithm, named COMmunity Circles (COMC), which utilizes a slotting system to efficiently showcase the relationships between each community. Our third algorithm, called COMmunity Attraction (COMA), visually amplifies both the attraction between community members, and the repulsion between vertices that do not share a community. Finally, we also propose a very simple edge-bundling technique to reduce visual clutter in dense networks.

Our contributions in this part are as follows:

1. Adding support for non-zero radius vertices to the existing FR algorithm and removing the bounding frame.

2. COMB, our two stage algorithm for generating an efficient community-centric layout that highlights the relationships between communities.

3. COMC, our algorithm which generates community centric layouts using a highly efficient slotting system to showcase the relationships between communities.

4. COMA, our algorithm which generates community centric layouts with intuitive modifications to the attraction and repulsion functions of the FR algorithm.

Chapter 11

Related Work

The basis of our research comes from a well-known layout algorithm by Fruchterman and Reingold that aims to produce layouts which are both aesthetically pleasing and contain uniform edge lengths. To accomplish this, the Fruchterman-Reingold (FR) [24] layout employs a force-directed model where each vertex in the graph both attracts its neighbours and repels all other vertices. The strength of this attraction/repulsion is regulated by the current temperature of the system; high temperatures produce strong attraction/repulsion forces and low temperatures produce weak forces. These forces determine how far away a vertex should move if it is repelled, or how close it should come if it is attracted.

The FR algorithm has three main functions: the attraction function, the repulsion function, and the position calculation function. At each iteration the FR algorithm computes the attraction function between each end point of an edge, and the repulsion function between all pairs of vertices. These forces are summed by the position calculation function and each vertex is moved so that it becomes closer to its neighbours and/or further away from its non-neighbours. The temperature is then reduced and the next iteration starts; the algorithm completes when the temperature reaches zero or the number of iterations reaches a threshold. We note here that because FR operates on edge end points it is able to natively support multigraphs by accumulating the attraction for all of the shared edges. We also note that although the temperature system described is comparable to simulated annealing algorithms, the layout itself does not do any hill-climbing due to the time complexity required to reach the optimal solution. Instead, the authors assume that the discrete

iteration-based movement of vertices in the system will be enough to overcome the local optima.

Other force-directed layouts have also been proposed, including the well-known Kamada and Kawai layout. In their approach, Kamada and Kawai (KK) [36] model the edges between vertices as springs and employ Hooke's law to compute the attractive forces between neighbouring vertices. Their algorithm then aims to minimize the total tension in the system; while also finding an 'ideal' distance between non-neighbouring vertices. As far as we know there has been no conclusive evaluation that determines whether the FR algorithm or the KK algorithm produces better layouts. We have chosen the FR layout as a basis for our algorithm because of its popularity amongst the research community.

There have also been numerous community or cluster based layout algorithms proposed by the research community, including the energy model proposed by Truong et al. [63]. In their model the authors aim to contain each cluster of vertices within a convex shape and to ensure that clusters do not overlap; very similar goals to the COMB algorithm we propose in this thesis. They do this by introducing a weighting factor to the attraction and repulsion forces in the FR algorithm to ensure that some optimal inter and intra-cluster distance is maintained. This optimal distance appears to be provided by the user and thus we feel their approach cannot easily be applied to general graphs without some previous knowledge of what the optimal distances should be. Furthermore, their approach is tailored to support clusters of disconnected vertices by adding 'phantom' edges between these vertices. We believe our algorithms are set apart from this energy based model because they do not require any user-specified parameter.

In addition, Frishman and Tal [23] have proposed a layout algorithm to visualize clusters over time. Their approach places emphasis on ensuring that the clusters remain in the same position across each timeframe and that the cluster sizes should be proportional to the number of vertices they contain. Even with very different goals, their method is strikingly similar to our own COMB algorithm in that they employ 'fake' vertices to set the position of each cluster and assign it a specific visual size based on the number of vertices within. However, because they favour

dynamic networks, their layout leaves much to be desired in aesthetic qualities such as uniform edge length and an inability to showcase intra-cluster structures.

Other methods have also been proposed to present hierarchical cluster results, where clusters can contain sub-clusters, including the approach demonstrated by Lancichinetti [40]. However his algorithm is unpublished and thus we cannot comment on the intuition or methodology that goes into the layout. Instead, we simply present a sample layout from the algorithm in Figure 11.1. Finally, Balzer and Deussen [6] have also proposed a method to visualize large networks containing numerous hierarchical clusters. Given a layout, their approach produces a level-of-detail visualization that displays everything from generalized cluster shapes, right down to the individual vertices within each cluster, depending on the amount of detail the user wants to see. Their approach combines edge-bundling, surface generation, and translucent filters to accurately depict the network structure without producing visual clutter.

Our proposed methods differ from previous approaches in that we do not require any user-defined parameters to produce our community layouts, nor do we focus on any specific aesthetic quality. Rather, we attempt to maximize the aesthetic quality of all aspects in the networks, such as the inter-community qualities, intra-community qualities, and the network as a whole. We believe this generalized approach produces more meaningful visualizations and, contrary to previous methods, we provide a detailed evaluation to showcase this.

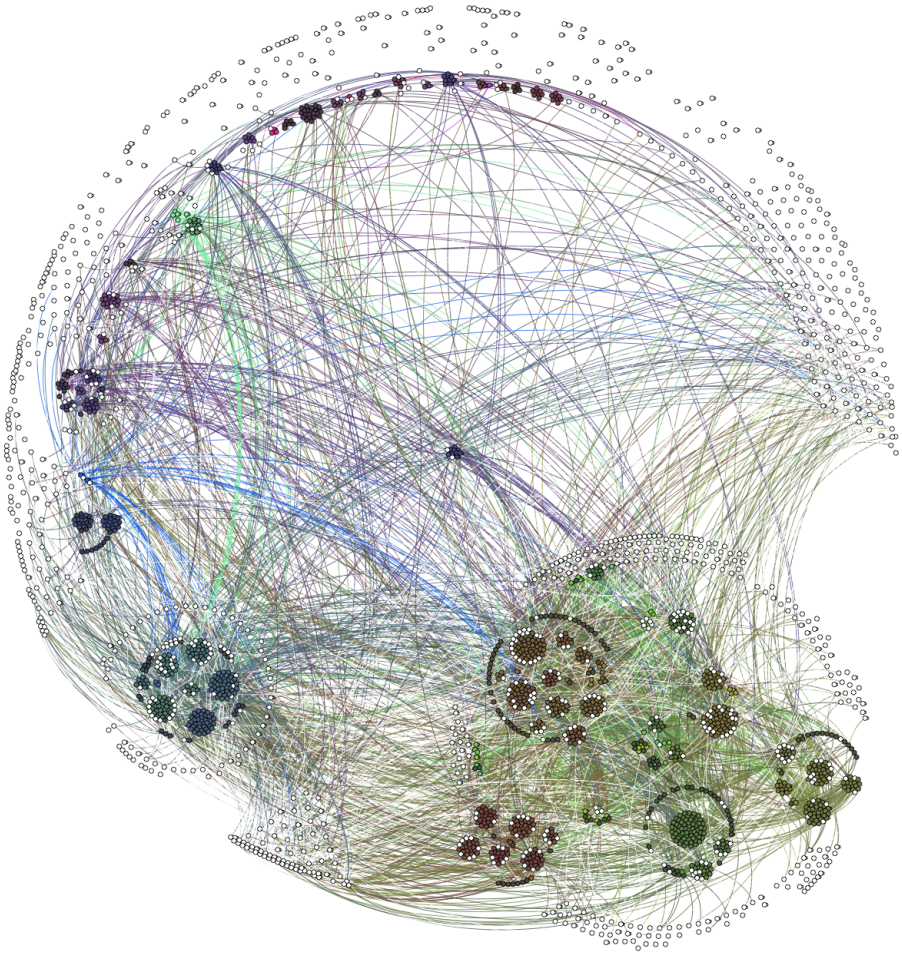


Figure 11.1: A visualization of the communities in the Protein-Protein Interaction Yeast Network using Lancichinetti's [40] layout algorithm.

Chapter 12

Fruchterman-Reingold Enhancements

12.1 Sized Vertices

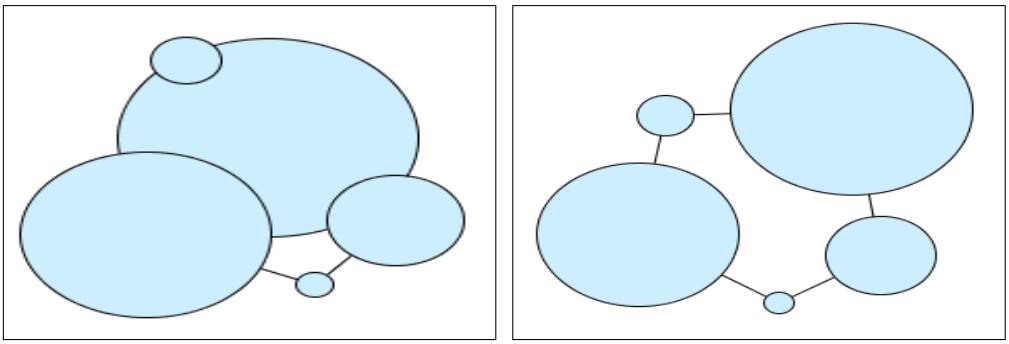
Before proposing our layout algorithms we feel it prudent to introduce two basic modifications to the existing FR algorithm. The first modification addresses the practical need to visualize networks that contain vertices with heterogeneous sizes. Such networks are common in real-life scenarios where vertex size conveys some meaningful information to the viewer. Unfortunately, the existing FR algorithm assumes that all vertices are infinitesimally small points and thus the resulting layouts often end up with large vertices overlapping or completely obscuring smaller vertices. An example of such a case is depicted in Figure 12.1(a). This overlap occurs because the repulsion function in the existing FR algorithm repels each pair of vertices by a force inversely proportional to the distance, d, between their centres. As d nears zero, the force becomes incredibly strong to ensure that vertices do not clump up.

Yet this repulsion is insufficient in the case when each vertex has a non-zero radius, because the vertices will be overlapping long before the distance between their centres nears zero. To solve this problem we apply an intuitive solution of measuring the distance, d, between the edges of the vertices, instead of the centres. This way the value d accurately reflects the distance between vertices of any size, and the extremely strong repulsive forces will occur before the vertices begin to overlap. We should note that although our solution is independently conceived, it

has been previously proposed by Harel and Koren [29]. More formally, we set the distance d equal to:

$$d = \max(dist(u, v) - (radius(v) + radius(u)), \epsilon)$$

Where $dist(u, v)$ is Euclidean distance between the centres of vertex u and vertex v, and $radius(x)$ is the radius of the vertex x. We include ϵ as a small positive constant to ensure that overlapping vertices do not result in a negative distance, d. The resulting layout is shown in Figure 12.1(b).



(a) Original FR algorithm. (b) FR with support for sized vertices.

Figure 12.1: An example of our modification to support sized vertices in the FR algorithm.

12.2 Boundary-Free Layout

Our second modification focuses on the FR algorithm's requirement of a user-defined length and width for the frame, or bounding box, that will be used to contain the resulting layout. The original authors argued that a layout is aesthetically pleasing when it conforms to a frame and is distributed evenly throughout the frame [24]. We, however, feel that this requirement is largely unintuitive. We do not often force a picture to fit a specific picture frame, but rather seek a frame that is large enough to contain our picture. Thus we opted to remove the need for a frame in our modified FR algorithm.

In doing so we were forced to redefine two of the constants in the existing FR algorithm; namely, the force constant, k, and the starting temperature, $temp$. These

constants were previously defined based on the width and height and, as stated in the related works section, they strongly influence the distance a vertex is able to move in a single iteration. Instead, we have provided the following alternative definitions:

$$k = \sqrt{(|V| + |E|)}$$
$$temp = |V| + |E|$$

Where $|V|$ is the number of vertices in the multigraph and $|E|$ is the number of edges. We have selected these definitions based on some intuition about how much movement should occur relative to the size of the network. Large networks require larger vertex movements in order to escape a local minima and thus their k and $temp$ values should be higher than smaller networks. We have chosen a square root function to ensure that very large networks do not result in extreme movements that end up accomplishing little else other than wasting an iteration.

Chapter 13

Community Boundaries

In this section we present our first layout algorithm, COMB, which consists of two stages. In the first stage we compute the boundary size for each community and then determine its optimal position in the layout. In the second stage we place each vertex within its community boundary and attempt to minimize the inter-community edge lengths.

Our algorithm operates on a multigraph $G = \{V, E, C\}$ where V is the set of vertices, $V = \{v_1, v_2, v_3...\}$, E is the set of edges, $E = \{e_1, e_2, e_3...\}$, and C is the set of pre-computed communities, $C = \{c_1, c_2, c_3...\}$. We define each community $c_x = \{V_{c_x}, E_{in_x}, E_{out_x}\}$ where V_{c_x} is the set of vertices in the community, E_{in_x} is the set of edges between two vertices within the community, and E_{out_x} is the set of edges leading out of the community.

13.1 Representative Vertices

We begin the algorithm by generating a single representative vertex for each community in the network. These representative vertices will ultimately determine the position of the communities in the final layout. As such, each of these representatives must be structurally equivalent to the community they represent in terms of both visual size on the screen and connections to other vertices.

To address the connection equivalence we generate a set of representative vertices, $R = \{r_{c1}, r_{c2}, ..., r_{cn}\}$, such that set of edges that each r_{cx} participates in, $E_{r_{cx}}$, is exactly equal to E_{out_x}, which is the set of edges its representative commu-

nity participates in. We should note that as per our definition it is very likely that each pair of representative vertices will be connected by more than one edge; thus the need for a multigraph structure.

To compute the on-screen size of a community we employ one of two techniques. The first technique is the obvious solution of generating a separate layout for each community using the Fruchterman-Reingold (FR) [24] algorithm and computing the convex hull of the layout. This approach gives us fairly accurate estimate of how much visual space the community will take up in the final layout, however it is costly to run the FR algorithm ($O(|V^2| + |E|)$) for each community in the network. An alternative technique is to estimate of the amount visual space required using only the number of internal edges ($|E_{in_x}|$), vertices ($|V_{cx}|$) within each community. We have defined our estimate function as:

$$radius = (\max(10, \sqrt{|V_{cx}|}) + \max(10, \sqrt{|E_{in_x}|})) * lm$$

Where lm is a constant derived from the average radius of each vertex. We employ the max functions to ensure that very small communities are allotted some meaningful visual space. It is important to note that any other estimate function could also be used in our algorithm; we simply chose one that makes some intuitive sense and has been empirically valid in our experiments.

By using either the estimate or the separate layout technique we can then set the radius of each representative vertex, r_{cx}, to be equal to the computed radius of the bounding circle for community x. An example of generating representative vertices from a set of communities is presented in Figure 13.1. The pseudo-code to generate representatives is available in Algorithm 1.

13.2 Initial Layout

After generating the representative vertices we can produce an abstraction of the network, $G_{abstract} = \{R + V_{no-comm}, E\}$, where R is our set of representatives and $V_{no-comm}$ is the set of vertices that do not belong to any community and thus are not captured by the representatives. Please see an example of $G_{abstract}$ in Figure

Input: C, a set of communities c_1,c_2,... where $c_i = \{V, E\}$, the set of vertices and edges belonging to the community c_i. $g = \{V_{all}, E_{all}\}$, the network to layout.

Output: A multi-graph G with the representative nodes and associated edges.

R = ();

G = new $Multigraph()$;

foreach $v_i \in V_{all}$ **do**

```
/* Include all outlier vertices in the
   abstraction.                                  */
```

if *getCommunity*(v_i) = $null$ **then**

G.addVertex(v_i);

end

end

foreach $c_i \in C$ **do**

```
/* Add sized representative nodes to the
   abstraction.                                  */
```

r = new $Vertex()$;

$r.size = (\max(10, \sqrt{c_i.|V|}) + \max(10, \sqrt{c_i.|E|})) * lm$;

$r.community = c_i$;

R.push(r);

G.addVertex(r);

end

foreach $r \in R$ **do**

```
/* Connect representative nodes with the edges
   their respective communities share.           */
```

c = $r.community$;

foreach $e \in E_{all}$ **do**

if $e.startVertex \in c.V$ *and* $e.endVertex \notin c.V$ **then**

$targetRep = getRep(getCommunity(e.endVertex))$;

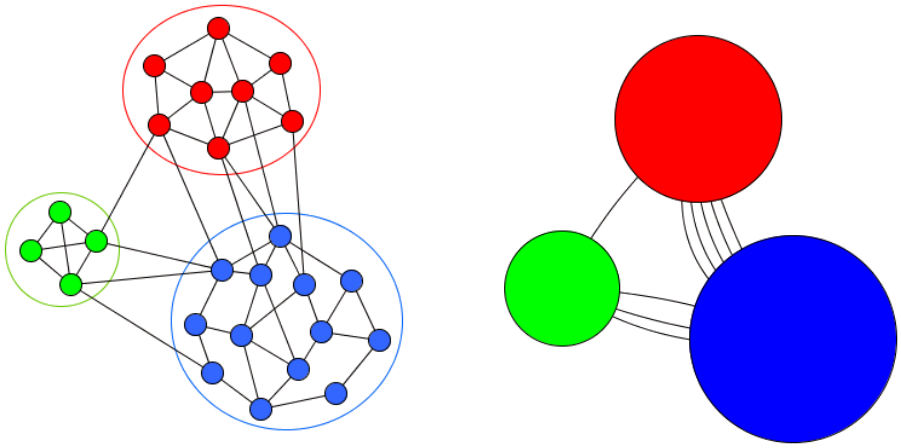
$G.addEdge(r, targetRep)$;

end

end

end

Algorithm 1: Generating an abstract network in COMB.



(a) Three communities (b) The generated Representative Vertices

Figure 13.1: An example of generating Representative Vertices.

13.1(b). We then feed this abstract network into our FR algorithm that has been modified to support vertex sizes.

The resulting layout contains representative vertices placed at their 'optimal' position, given the goal of uniform edge length. These positions are recorded and will be used to determine where each community should reside in the final layout. Note that we do not record the position of the vertices in $V_{no-comm}$. These vertices were only added to the abstraction to ensure that their edges are considered when finding an optimal position for the representatives.

13.3 Vertex Placement

In the second stage of COMB we focus on refining our initial placement by setting the location of each vertex in the network. We begin this stage with the initial layout generated in stage 1 that contains the representative vertices. We then replace each representative vertex with a bounding circle, centred at the same location that the vertex once occupied. These bounding circles will ensure that the final layout has clearly defined boundaries for each community. As such, each circle must have a radius equal to that of the representative vertex it replaced.

Once the bounding circles are appropriately sized we begin inserting the vertices from the original network. Each vertex is placed at a uniformly selected ran-

dom position within the circle that represents its community. After the placement is complete we have an initial layout where all vertices belonging to the same community are contained within the same bounding circle. When we feed this this initial placement into our modified FR algorithm we expect the vertices to move around within their circle, but they can not leave the circle. This limited freedom of movement should allow those vertices with many connections to other communities to migrate towards the perimeter of their respective circle. In the following section we define these bounding circles.

13.4 Implementing Bounding Circles

To enable bounding circles we have modified the position calculation function int he FR algorithm so that each vertex is checked against the perimeter of its bounding circle in each iteration. We determine if the vertex is outside the perimeter by calculating the distance between the vertex and the centre of the circle. Any vertex with a distance greater than the radius must be outside of the perimeter and needs to be relocated so that it is inside the perimeter once more. Before relocating the vertex we wish to record its angle to the centre of the circle. This angle should point towards the optimal position (as determined by the FR algorithm) that the vertex was trying to reach and thus it may be the optimal angle within the bounding circle as well.

Therefore we first compute the angle between the out-of-bounds vertex and the centre of the circle. More formally:

$$angle(v, c) = atan2(c.y - v.y, v.x - c.x)$$

Where v is the vertex, c is the bounding circle that contains the vertex, and x/y refers to the x and y coordinates of a point. $c.x$, for example, would indicate the x coordinate for the centre point of the circle c.

Once the angle is computed we can now move the vertex within the circle, but how far inwards? We noticed in our early experimentation that moving the vertex exactly to the perimeter produced poor layouts, as the perimeters of the circles be-

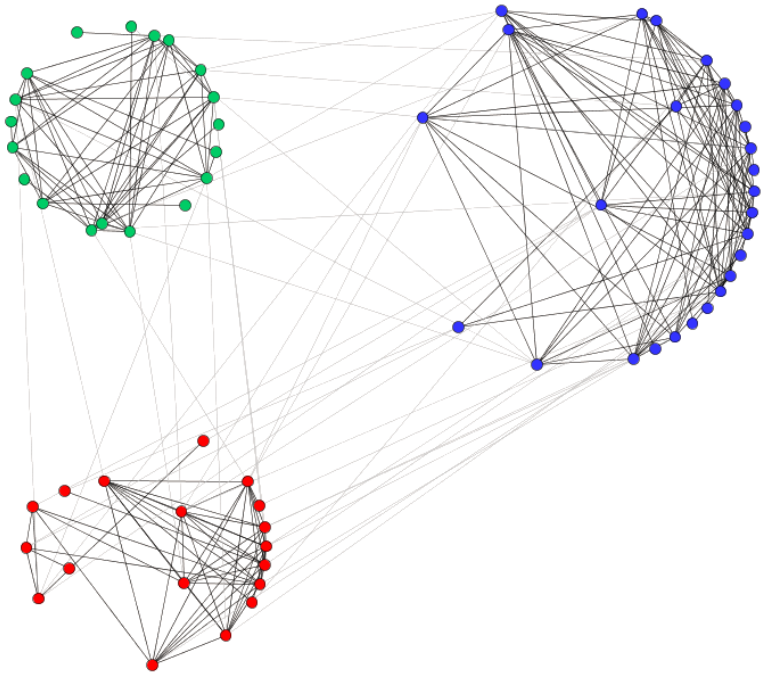


Figure 13.2: Bounding Circles without randomized bounce-back.

came extremely crowded and obfuscated the structure of the community itself. See Figure 13.2. To combat this problem we applied some randomness to the movement. More specifically, each offending vertex is moved to a randomly selected position between the perimeter and the centre of the circle, while keeping the same angle. This randomness follows a uniform distribution.

However, the randomness alone is not satisfactory because a vertex may be incorrectly moved to the centre of the circle on the very last iteration of the algorithm, even though it just barely slipped outside of the bounding circle. Thus we also take into account the temperature of the system. When the temperature is near its maximum, the offending vertices may be moved all the way to the centre of the circle; but as the temperature cools, the movements do not stray far from the perimeter. We capture this intuition in the following formulae:

$$
\begin{aligned}
mlen(c) &= c.radius * \left(1 - \left(rand() * \frac{curTemp}{maxTemp}\right)\right) \\
length &= mlen(c) \\
v.x &= (\cos(angle(v,c)) * length) + c.x \\
v.y &= (\sin(angle(v,c)) * length) + c.y
\end{aligned}
$$

Where c is the bounding circle for the vertex v, $angle()$ is previously defined, $rand()$ returns a uniformly random number [0,1], $curTemp$ is the current temperature, and $maxTemp$ is the initial temperature.

As the system cools, $curTemp$ decreases, causing the $length$ to increase, which places the vertices (on average) further away from the centre of the circle and closer to its perimeter. This ensures that vertices which barely slip out of the circle in the final iterations of the algorithm are still able to remain near the perimeter.

13.5 Final Layout

Now that we have defined the bounding circles we merely provide the entire network, G, to our modified FR algorithm. We then feed the initial placement of the vertices, and the definitions of each bounding circle, into the algorithm. The resulting layout contains all of the features of the original FR algorithm along with a well-defined visual boundary around each community. An example of a final layout is depicted in Figure 13.3. The pseudocode for our modified FR algorithm is available in Algorithm 2.

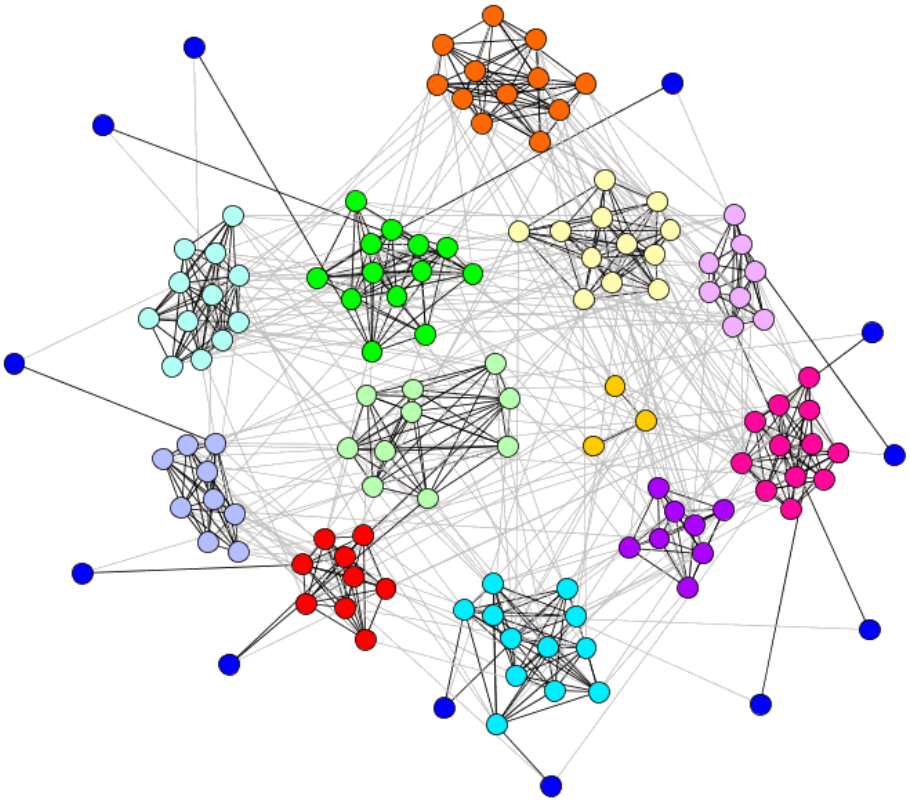


Figure 13.3: An example of a layout produced by COMB. Vertices in blue are outliers.

Input: A multi-graph G = {V,E}, where $v \in V$ has a $.community$ attribute set for the $centre$ and $radius$ of the community's bounding circle.
Output: A position for each vertex in G.

$k = \sqrt{(|V| + |E|)}$;
$temp, temp_{initial} = |V| + |E|$;
$f_a(z) = z^2/k$; $f_r(z) = k^2/z$;

```
/* Initialize a random position for each vertex.  Bounded
   vertices are forced into their circles.                  */
```
foreach $v \in V$ **do**
  if $v.community \neq null$ **then**
    $v.pos = getRandomPos(v.community)$;
  else
    $v.pos = getRandomPos()$;
  end
end

foreach $iteration$ **do**
  `/* Calculate repulsion function.                          */`
  foreach $v \in V$ **do**
    foreach $u \in V$ **do**
      if $u \neq v$ **then**
        $\Delta = v.pos - u.pos$;
        $d = \max(|\Delta| - (v.radius + u.radius), \epsilon)$;
        $v.disp = v.disp + ((\Delta/d) * f_r(d))$;
      end
    end
  end
  `/* Calculate attraction function.                         */`
  foreach $e \in E$ **do**
    $\Delta = e.start.pos - e.end.pos$;
    $d = \max(|\Delta| - (e.start.radius + e.end.radius), \epsilon)$;
    $e.start.disp = e.start.disp - ((\Delta/d) * f_a(d))$;
    $e.end.disp = e.end.disp + ((\Delta/d) * f_a(d))$;
  end
  `/* Limit the maximum displacement via the temperature.   */`
  foreach $v \in V$ **do**
    $v.pos = v.pos + \frac{v.disp}{|v.disp|} * \min(v.disp, temp)$;
    if $v.community \neq null$ **then**
      `/* Ensure bounding circles are enforced.             */`
      if $dist(v, v.community.centre) > v.community.radius$ **then**
        $angle = getAngle(v.pos - v.community.centre)$;
        $tempMod = 1 - \frac{temp}{temp_{initial}}$;
        $disp = v.community.radius * (1 - (rand() * tempMod))$;
        $v.pos.x = (\cos(angle) * disp) + v.community.centre$;
        $v.pos.y = (\sin(angle) * disp) + v.community.centre$;
      end
    end
  end
  `/* Cool the system according to a user defined cool().   */`
  $temp = cool(temp)$;
end

Algorithm 2: The COMB algorithm, with sized vertices, bounding circles, and removal of borders. Parts of this pseudo-code are adapted from the FR paper [24].

Chapter 14

Community Circles

The layouts produced by COMB contain clearly defined boundaries and reveal both the internal structure and relationships of the communities. Highlighting this internal structure, however, has an efficiency cost as the layout algorithm needs to retain all of the functions from the original FR algorithm, resulting in an $O(|V|^2 + |E|)$ time complexity. This is comparable with existing layout algorithms, but it is not fast enough to be considered a truly interactive visualization. Thus we propose our COMmunity Circles (COMC) algorithm, which maximizes efficiency by using circular layouts to showcase only the community relationships. Our goal with COMC is to represent each community as a circle and place the vertices on the perimeters of each circle such that inter-community edge length is minimized. This algorithm is similar to COMB, in that it is based on FR, and shares many of the same steps, including the generation of representative vertices and the resulting vertex replacement.

14.1 Sizing Representative Vertices

Unlike in COMB, we do not need to scale the size of the representative vertices according to the number of edges within the community, as these edges are essentially ignored by the COMC layout. Instead, we need to ensure that the perimeter of each representative is large enough so that we can place each vertex on it without overlap. Thus we modify the radius function:

$$circumference = (avgsize(V_{cx}) + padding) * |V_{cx}|$$
$$radius = \frac{circumference}{2\pi}$$

where $avgsize()$ is the average radius of all the vertices in V_{cx}, and $padding$ is a user-defined constant that describes how much space there is between each vertex on the circle. Our preferred setting $padding = 15$.

14.2 Perimeter Slots

Once the representatives are generated and placed using COMB, we can record the perimeter and location of each representative and use this information to define a new data structure. This structure contains both the center and radius of the circle along with an array of "slots," one for each vertex in the community. During the course of the COMC algorithm, each vertex is assigned to the closest slot on its respective community circle, according to the Euclidean distance between the circle and the current position of the vertex. If the closest slot is already occupied by another vertex, we compute the total length of all inter-community edges for both vertices. The vertex with the smallest total edge length gets the slot, and the other vertex must move to the second closest slot. If that slot is also occupied, then the procedure repeats itself until there is a suitable unoccupied slot. We can guarantee that this process terminates because the set of inter-community edge lengths has a total order.

Before assigning the slots, we determine the position of each vertex by leveraging the attraction and repulsion functions of the FR algorithm. In particular, we only compute the attraction between two vertices if they belong to different communities. This is because we do not need to consider the attraction between members of the same community; they are all tied to the perimeter of the circle and cannot move closer to each other. In addition, we do not need to compute the repulsion function for any vertex that belongs to a community. We can avoid this step because a pair of vertices within a community cannot possibly get further away than the diameter of circle and we not interested in showcasing the internal structure of each community.

The attraction function sets the position of each vertex by drawing it closer to its neighbours in other communities. Once this position is set, each vertex is assigned to the nearest slot on its community circle as described above. At the end of each iteration, the vertices are moved to the position dictated by their respective slot, and then the slots are cleared for the next iteration. The algorithm continues until the system has cooled according to a cooling function specified in the FR algorithm [24].

14.3 Efficiency Gains

By removing the repulsion function and limiting the attraction function, we have greatly reduced the time complexity of the algorithm. In each iteration, we need only loop over the set of inter-community edges to compute the attraction function and then assign the slots. This reduces the worst-case time complexity from $O(|V^2| + |E|)$ down to $O(|E_{out}| + |V| * avg(|C|))$, where $avg(|C|)$ is the average number of vertices in each community. In practice, the slot assignments tend to stabilize after the first few iterations of the algorithm, and thus the time complexity is dominated by $|E_{out}|$. A sample visualization of COMC is provided in the Evaluation section.

Chapter 15

Community Attraction

Although our COMmunity Boundaries (COMB) algorithm meets its goals of creating a stark visual boundary around each community, it sacrifices some of the community's structural aspects to do so. Not every community can fit comfortably within an appropriately sized circle and many vertices end up being squished together with the bulk of the community. This is especially troubling for outlier vertices that are pushed towards the middle, even though they have only a single edge and would look better on the outskirts of the community. This squishing also causes each community to appear more dense than it actually is, which can mislead the viewer. Unfortunately adding more room to each circle does not solve the problem, as this leads to large patches of white space forming between the communities and takes away from the aesthetic qualities of the full layout.

For cases when a stronger emphasis should be placed on community structure, we present our COMmunity Attraction (COMA) layout algorithm, with slightly different goals. Instead of enforcing a strong visual boundary between communities, the COMA algorithm merely encourages visual boundaries, and rather focuses on highlighting the structural elements of each community. To accomplish these goals we have modified the two major forces in the FR algorithm: repulsion and attraction. We believe that if two vertices belong to the same community they should have a stronger attraction than if they belong to different communities. Likewise, if two vertices belong to different communities they should have a stronger repulsion than if they belonged to the same community. We note that this idea is similar to the community attraction and repulsion method presented by Truong et al. [63].

However, their method considers some user-defined optimal distance between vertices in a cluster; we have no such requirement. Also, their top priority is to ensure that communities are positioned in a convex zone [63], which greatly contradicts our goals for COMA.

15.1 Repulsion Function

In order to ensure that two vertices belonging to different communities are repelled, we apply a weighting factor to the repulsion function in the FR algorithm. We want to ensure that the repulsion is strong enough to keep somewhat of a defined boundary between communities, but not too strong as to cause unnecessary white space in the layout. Also, our approach should be scalable for very small networks, and very large networks, without requiring the user to set parameters.

We address these needs by proposing a weighting factor based on the density of each community. More specifically, the repulsion between vertex u and vertex v is determined by the density of communities that contain vertex v and u. This follows from our intuition that communities with high density can afford to have their vertices repulsed a considerable distance, as they will be dragged back into the community by the attraction function. Likewise, communities with lower densities cannot afford a strong repulsion as they have relatively few edges that connect to the repulsed vertex, and thus may not be able to draw the vertex back in.

Assuming that $f_r(z)$ is the repulsion function and that vertices v and u belong to different communities, we apply the weighting factor as:

$$force = f_r(dist(u,v))^{(1+density(u_c)^2)}$$

Where $density(u_c)$ is the density of the community that vertex u belongs to. We define $density$ as $\frac{|E|}{|V|*|V-1|}$. Note that the repulsion function is performed for all unordered pairs of vertices, thus $f_r(dist(v,u))$ will also be computed. We apply the weighting factor as a exponent because we want to amplify the repulsion in very large networks, where the force constant is larger, more so than in smaller networks.

Also, we have squared the density so that extraordinarily dense communities do not cause massive movements in the vertices with which they interact.

15.2 Attraction Function

Much like in the repulsion function, we also apply a weighting factor to the attraction function to ensure that vertices belonging to the same community are strongly attracted to each other. Once again, we want to ensure the attraction is strong enough to keep the community together, but not so strong as to prevent the outliers from drifting towards the outskirts.

In this function, the weighting factor is determined by the density of the network as a whole. We believe that if the network is very dense, then there is a high probability that many inter-community edges are attached to the community. These sum of the attraction forces in these edges may pull the community apart; thus the intra-community attraction must be quite strong to hold the community together. Likewise, if the network has low density, then there are likely few inter-community edges and thus the intra-community attraction need not be as strong.

Assuming that $f_a(z)$ is the attraction function and that vertices v and u belong to the same community, we apply the weighting factor as:

$$force = f_a(dist(u,v))^{(1+\sqrt{density(G)})}$$

Where $density(G)$ is the density of the network as a whole. Once again we apply the weighting factor as a exponent because we want to amplify the strength of the attraction in very large networks more so than in smaller networks. We have square-rooted the density so that even in very sparse networks the weighting factor is still meaningful.

The pseudo-code for COMA can be extrapolated from the pseudo-code for COMB. Simply remove the bounding circle modification and adjust the f_a and f_r functions according to the definitions provided above. A visualization of a COMA layout is provided in Figure 15.1.

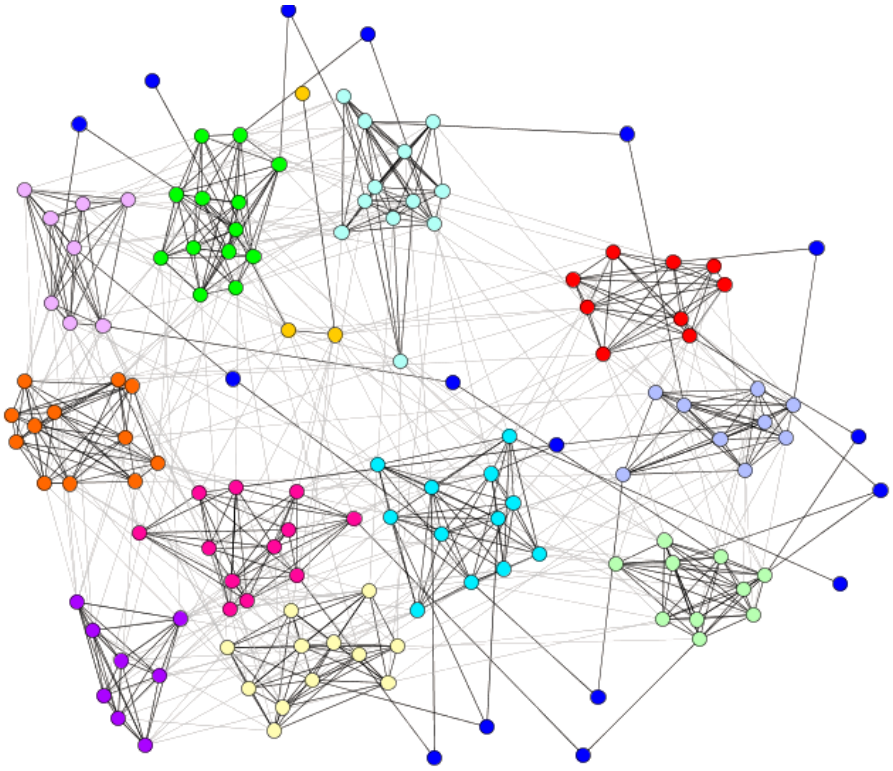


Figure 15.1: An example of a layout produced by COMA. Vertices in blue are outliers.

Chapter 16

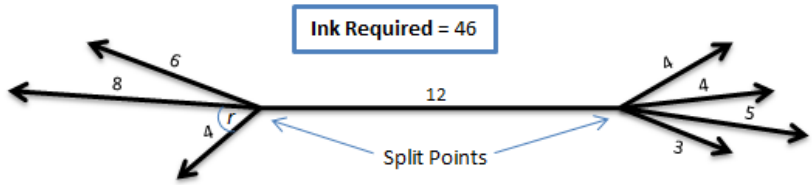
Edge Bundling

As you can see in the final layout examples for both COMB and COMA, the visualization becomes polluted by the sheer number of edges between each community. The higher the average degree of the network, the worse this problem becomes. To address this issue we explored the usage of well-known edge bundling techniques proposed by Holten [31], and by Gansner et al. [25]. The proposed techniques aim to reduce the visual clutter by bundling together edges that are heading in roughly the same direction. This greatly reduces the number of visible edges without degrading the information those edges convey to the viewer. However, both of the proposed techniques proved to be too slow for our highly interactive analysis software, Meerkat [10]. Also, they were designed for a general graph and were unable to leverage the knowledge that each community must have its own edge bundle.

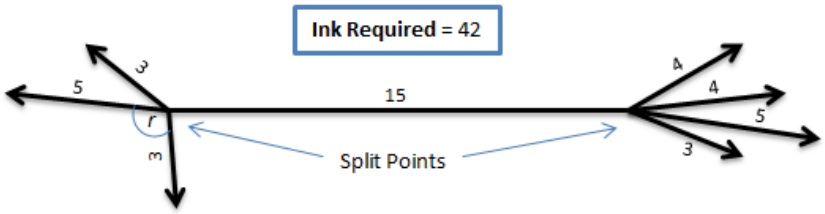
Instead of these approaches, we briefly present our own bare bones edge bundling technique extracted from the algorithm provided by Gansner et al. [25]. Our goal is relatively simple; we wish to create a separate edge bundle for each pair of communities. That way we can greatly reduce the visual clutter by having at most $\frac{|C|*|C-1|}{2}$ visible inter-community edges, where $|C|$ is the number of communities. Deciding which edges to bundle is rather trivial; we simply take all edges with an end point in both Community X and Community Y and bundle them together into the XY bundle.

Once bundled we must decide where each end of the bundle should split back into its constituent edges. Split too early and we risk reintroducing the visual clutter we were trying to remove; split too late and the constituent edges will fan out into

an undesired circle. To determine the optimal splitting points we follow the method proposed by Gansner et al. [25] and record the amount of ink required to draw the bundle, given two splitting points. The ink can be thought of as equal to the distance your pen would cover if you were to manually draw the bundle itself and all of its constituent edges after the splits. An example for two different split points is shown in Figure 16.1.



(a) Split points with a small angle r.



(b) Split points with a large angle r.

Figure 16.1: An example of split points.

More formally, the ink required to draw a bundle, given the split points, is provided in the following formula:

$$Ink(B, sp1, sp2) = d(sp1, sp2) + \sum_{e \in B} (\min(d(e.start, sp1), d(e.end, sp1)) + \min(d(e.start, sp2), d(e.end, sp2)))$$

Where $d()$ is the Euclidean distance between two points, B is the set of edges in the bundle, $sp1$ and $sp2$ are the first and second split points, $e.start$ is the first end point of the edge, and $e.end$ is the second end point.

This definition of $Ink()$ can further be refined by considering the max angle, r, between a split point and any of its constituent edges. We agree with Gansner et

al. [25] that split points with high r values are not ideal because they lead to sharp angles that reduce the smoothness of the bundling and lead to a poor visualization. An example of high and low r values is depicted in Figure 16.1. To compute the r value we merely find the edge with the maximum angle between it and the split point, and record this angle. We then modify our Ink formula to take this angle into account:

$$AInk(B, sp1, sp2) = Ink(B, sp1, sp2) * \left(1 + \frac{\cos(r)}{p}\right)$$

Where $Ink()$ is defined above, r is the maximum angle between an edge and its associated split point, and p is a dampening factor. Larger values of p indicate that less emphasis should be put on reducing the maximum angle r and smaller values indicate more emphasis. For our experiments we set $p = 5$.

Once we can compute the required ink we need only iterate over all possible split points to find those which require the minimum amount of ink. However, this is very costly; thus we employ the golden section search (GSS) algorithm, also suggested by Gansner et al. [25], to efficiently locate the optimal split point positions. We should note that the GSS algorithm is univariate; thus we cannot search for both split points at the same time. Instead, we must hold one split point constant while we search for the optimal location of the other split point, and vice versa. To initialize the algorithm we provide the centroids of each community as the starting split points.

Finally, we scale the rendered width of each edge bundle by the square root of the number of edges within that bundle. This offers a non-intrusive way to convey the approximate number of bundled edges to the viewer. An example of our edge bundling algorithm is shown in Figure 16.2.

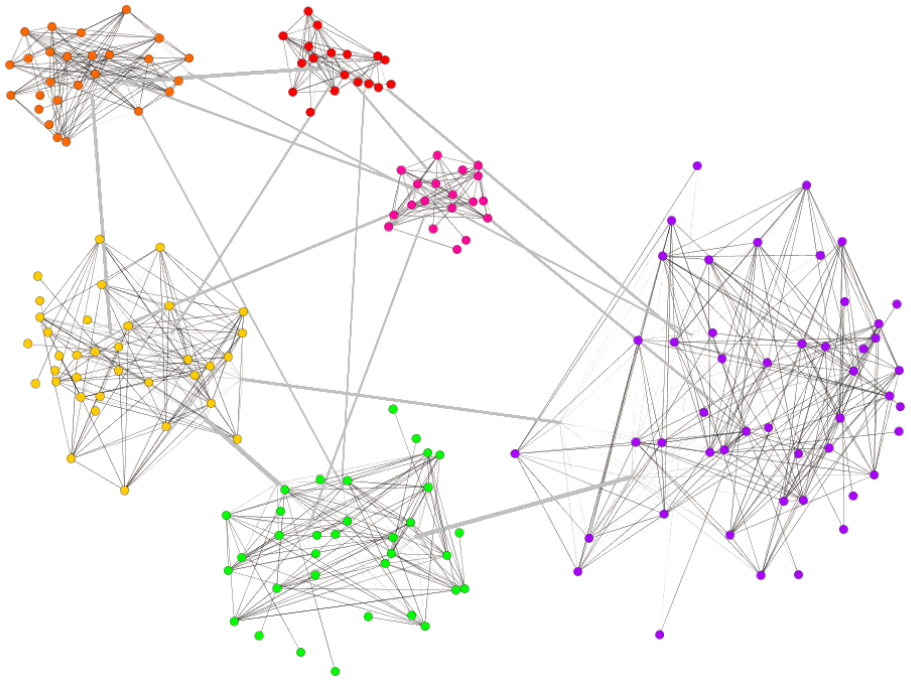


Figure 16.2: An example our simplified edge bundling technique.

Chapter 17

Evaluation

17.1 Visual Juxtaposition

Research in visualization evaluation (e.g., [58, 54]) suggest an evolution towards the design of cognitive experiments to establish measures of visualization effectiveness. But a useful precursor to such formal evaluation is to provide informal arguments of visualization method preferences by comparing alternative renderings of the same kind of base data. In this way we at least begin the development of intuitive measures that provide the basis for visualization method preference. We have adopted this approach.

In our comparison of alternative methods for community clustering visualization, we are unable to include any of the related work algorithms because we could not find implementations and the examples provided in the respective papers were trivial at best. We have also opted to not include edge bundling in these visualizations because our bundling algorithm is not easily extended to non-community centric approaches and thus would give our algorithms an unfair advantage.

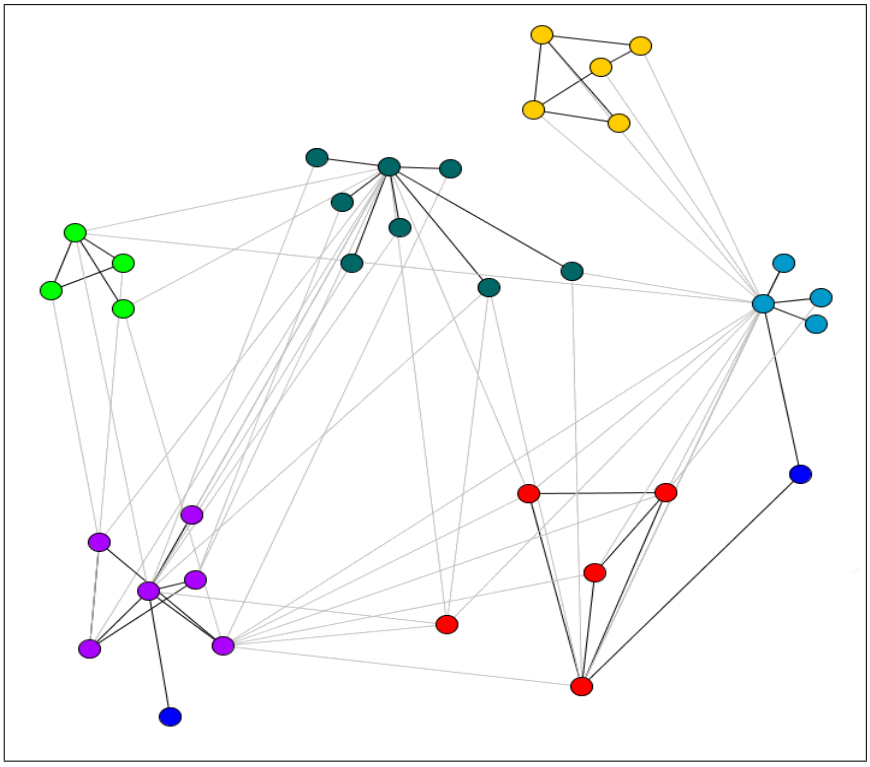
In Figures 17.3, 17.6, and 17.9 we present visualizations for a variety of well-known social networks, including the NCAA Football network [27], Zachary's Karate Club [66], and the Political Books network[39]. For each network we have run an existing Community Mining algorithm, such as Fast Modularity [14], and generate a visualization using COMB, COMC, COMA, the Kamada Kawai (KK) algorithm [36], and the general Fruchterman-Reingold algorithm [24]. We note that although the KK and FR algorithms are not specifically tailored for commu-

nities they are nonetheless the most popular methods used to visualize community mining results. To generate the visualizations we have used the FR and KK implementations provided by the JUNG framework [47]. Note that the colour of a vertex specifies the community to which it belongs.

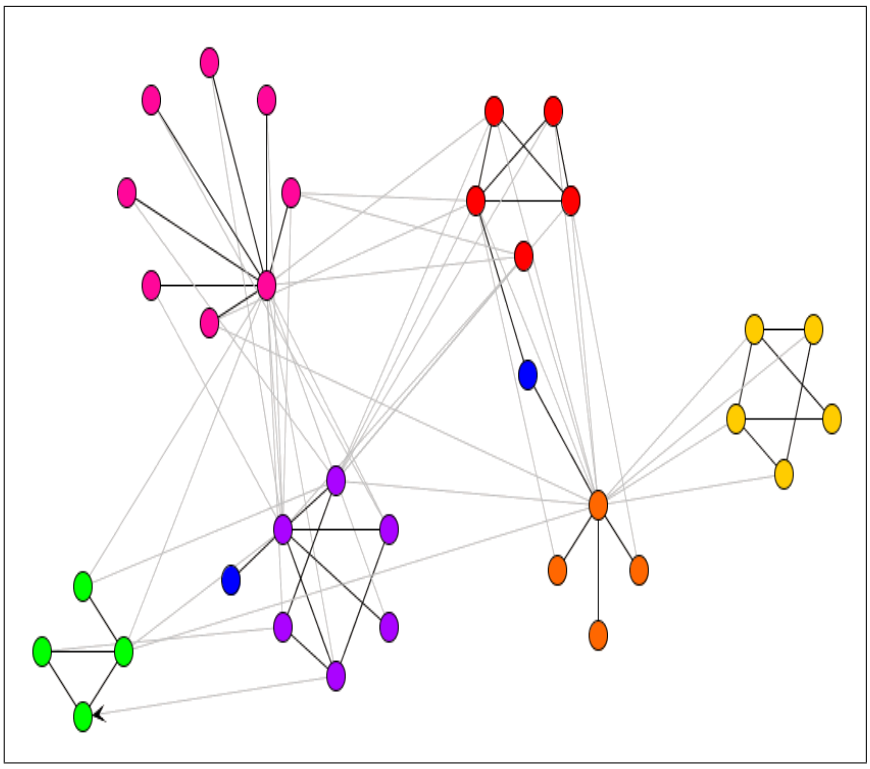
For the rather trivial Karate Club network depicted in Figure 17.3 we can already see a clear distinction between our proposed algorithms and the generalized layouts. In the COMB, COMC, and COMA layouts we can easily see the relationships between the communities and the structure of each community. This insight would allow the viewer to recommend that perhaps the yellow and teal communities should be merged. Similar observations are much harder to make in the generalized layouts because it is difficult to tell how each community is related.

For the Political Books network shown in Figure 17.6 we can see that the COMB, COMC, and COMA layouts highlight the tight coupling between the orange, pink, and purple communities, whereas one can only assume such a coupling exists in the FR and KK based layouts. Furthermore the COMA layout clearly shows that the structure of the yellow community is rather weak; the triad in the middle of the network only has single tie between it and the main body of the community. A similar observation is not as obvious in the other layouts.

Finally, in the NCAA Football network, shown in Figure 17.9, we can see clear community boundaries and the interaction between these communities in the COMB, COMC, and COMA layouts. Furthermore, both the FR and COMA layouts reveal that each of the communities contain numerous outliers (the single edge vertices), indicating that perhaps a community mining algorithm with support for outliers should be used instead. Unfortunately, the FR layout provides no other information as the communities seem to be completely overlapping each other, making it difficult to determine the structure of any community. The same can be said for the KK layout.

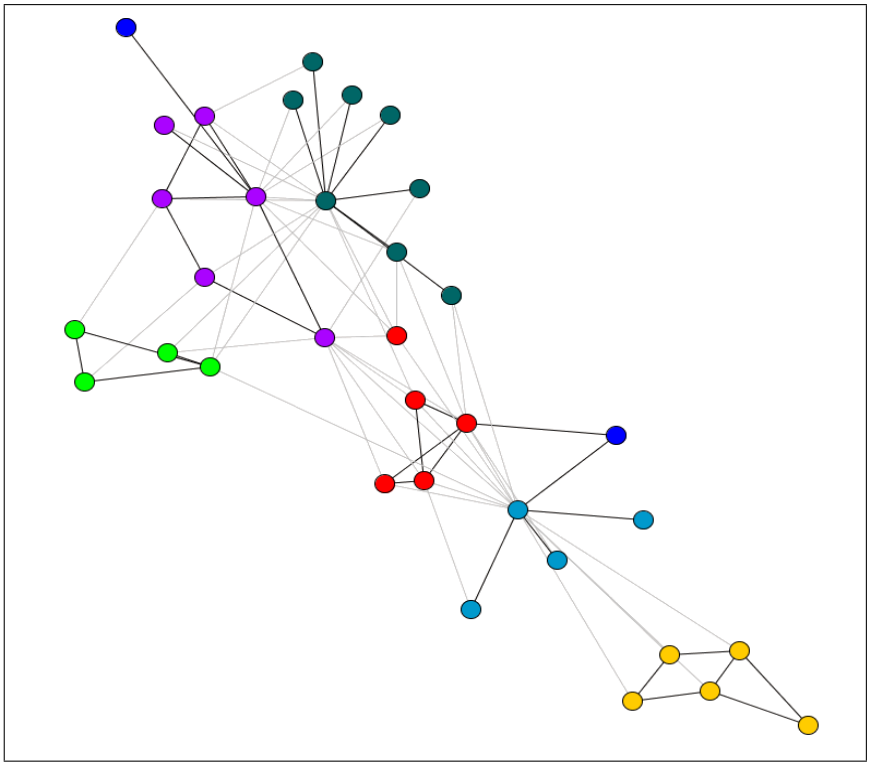


(a) COMB

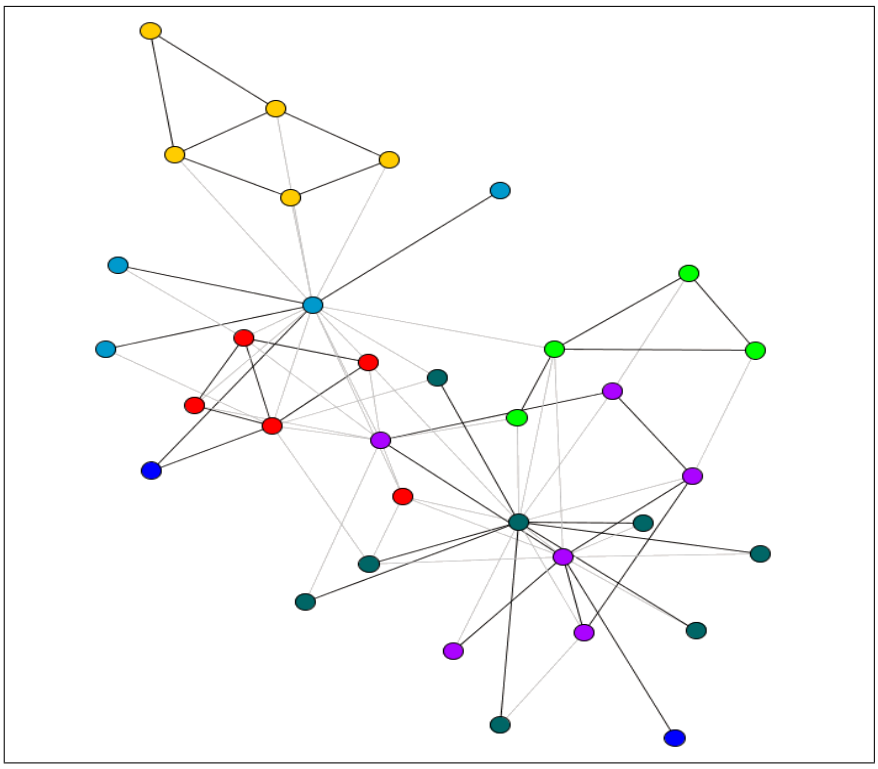


(b) COMC

Figure 17.1: Zachary's Karate Club Network.

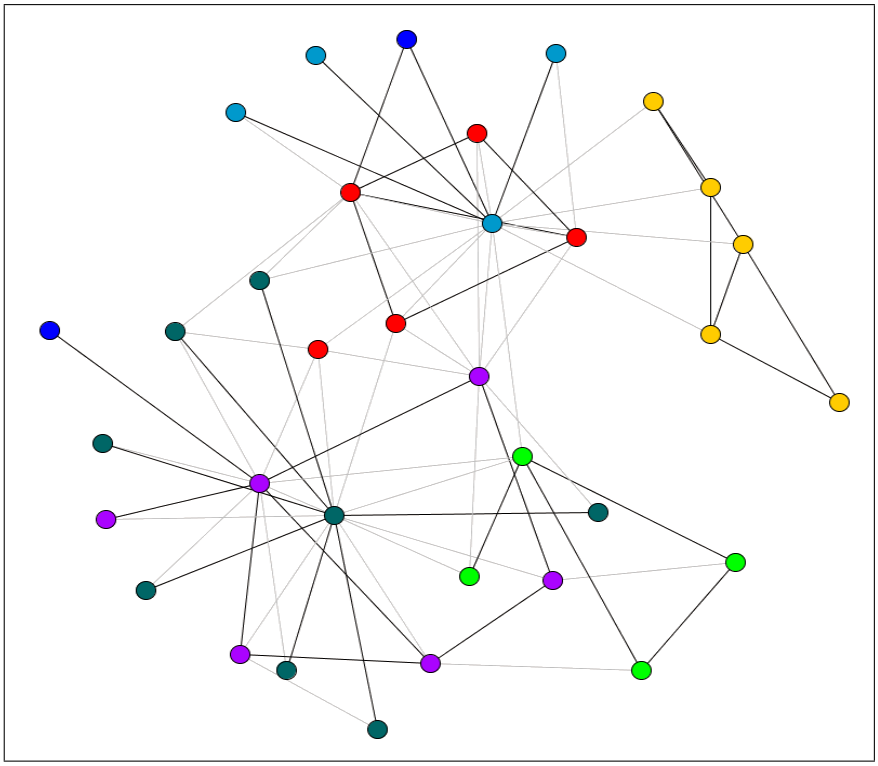


(a) COMA



(b) Fruchterman-Reingold

Figure 17.2: Zachary's Karate Club Network.



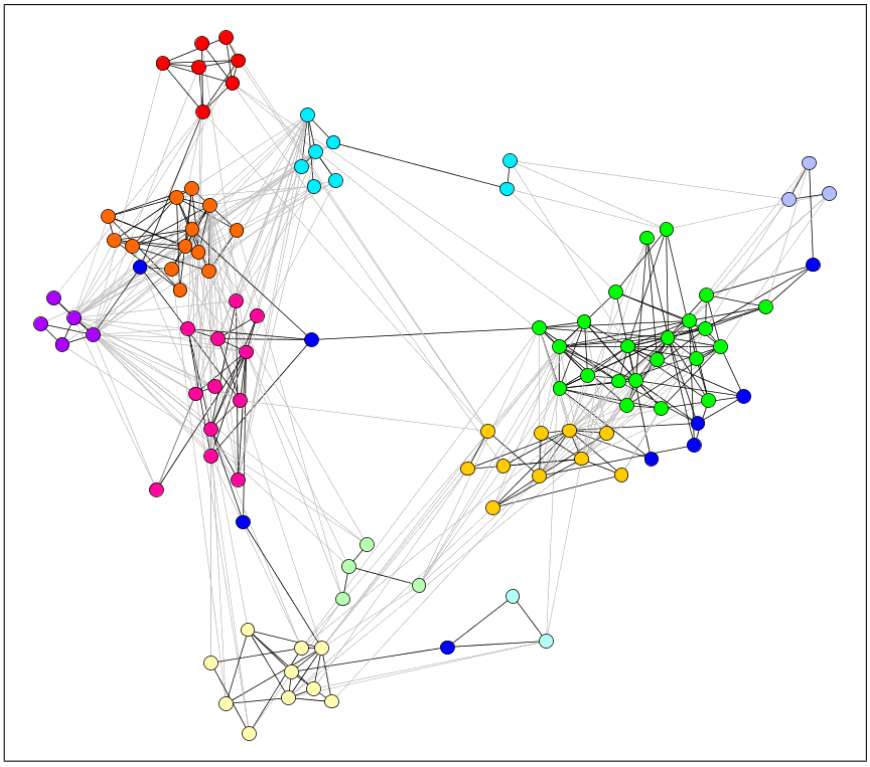
(a) Kamada-Kawai

Figure 17.3: Zachary's Karate Club Network.

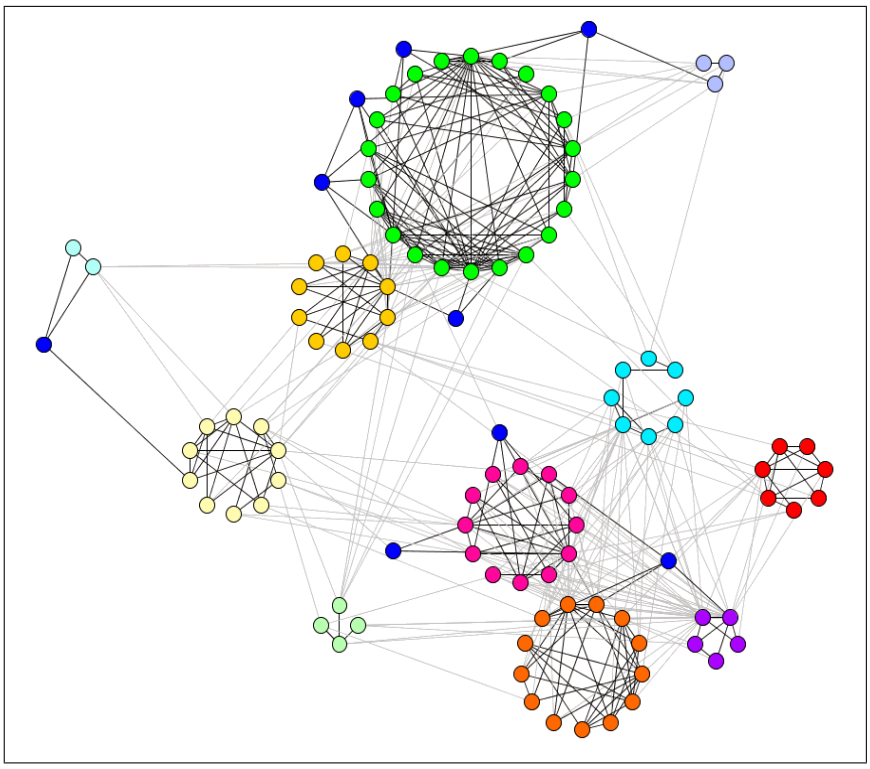
17.2 Efficiency

In addition to the visual evaluations, we also present a comparison between the efficiency of the algorithms. The COMB, COMA, FR, and KK algorithms each employ a force-based approach that requires computing a force between every pair of vertices, and the end points of each edge, in each iteration. Thus the time complexity of each iteration can be thought of as $\Theta(|V|^2 + |E|)$; our modifications to the FR algorithm do not add any significant time complexity.

Computing the final time complexity requires determining how many iterations are possible; yet this is a notoriously difficult question in systems with cooling schedules. To avoid this issue the authors of previous papers have simply set a limit to the number of iterations, say 100. We have adopted this approach and determined that the final time complexity of each algorithm is $O(|V^2| + |E|)$. The COMC algorithm does not need to compute the pair-wise forces and thus its worse case time complexity is $O(|E_{out}| + |V| * avg(|C|))$.

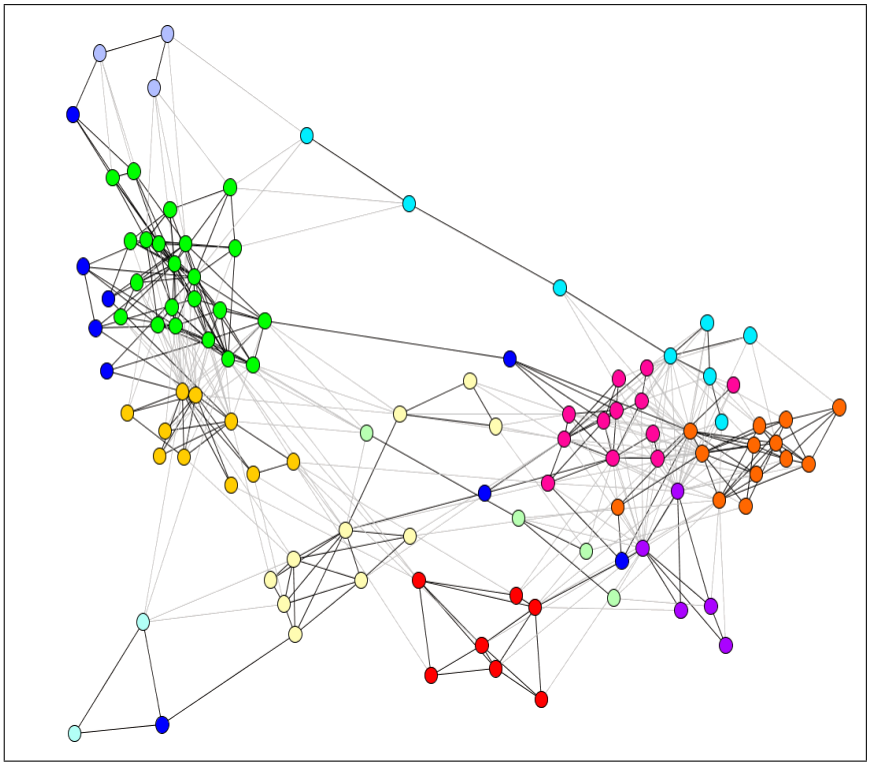


(a) COMB

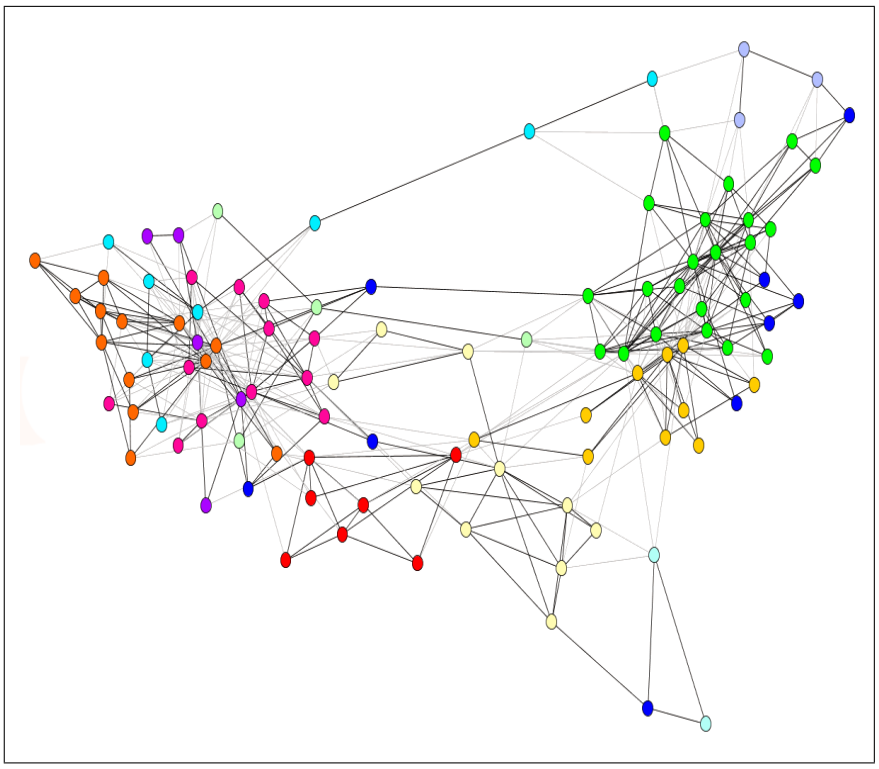


(b) COMC

Figure 17.4: Political Books Network.

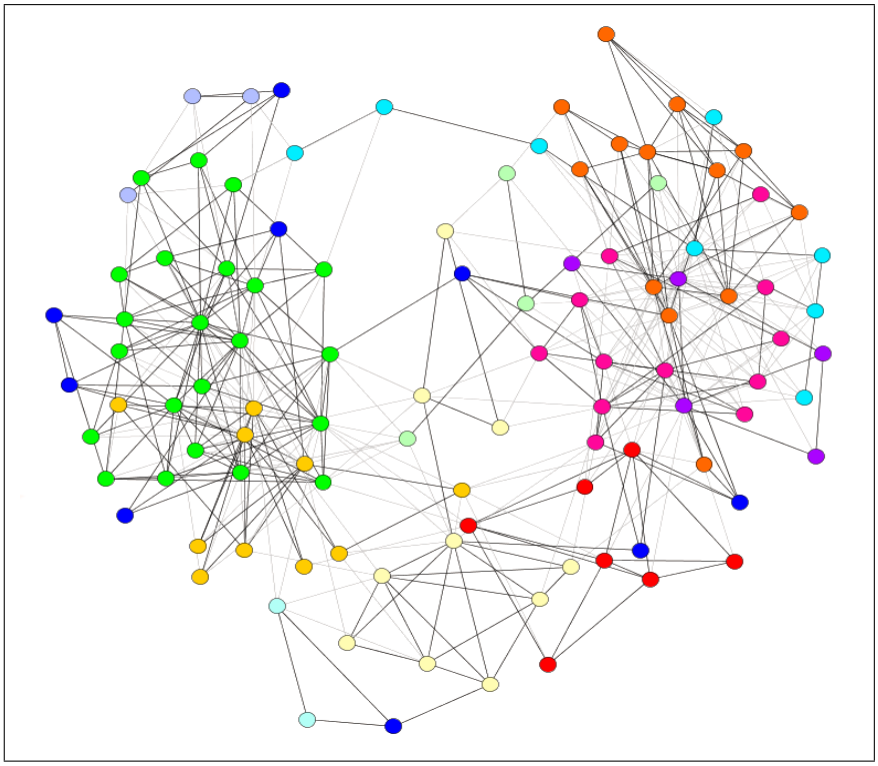


(a) COMA



(b) Fruchterman-Reingold

Figure 17.5: Political Books Network.

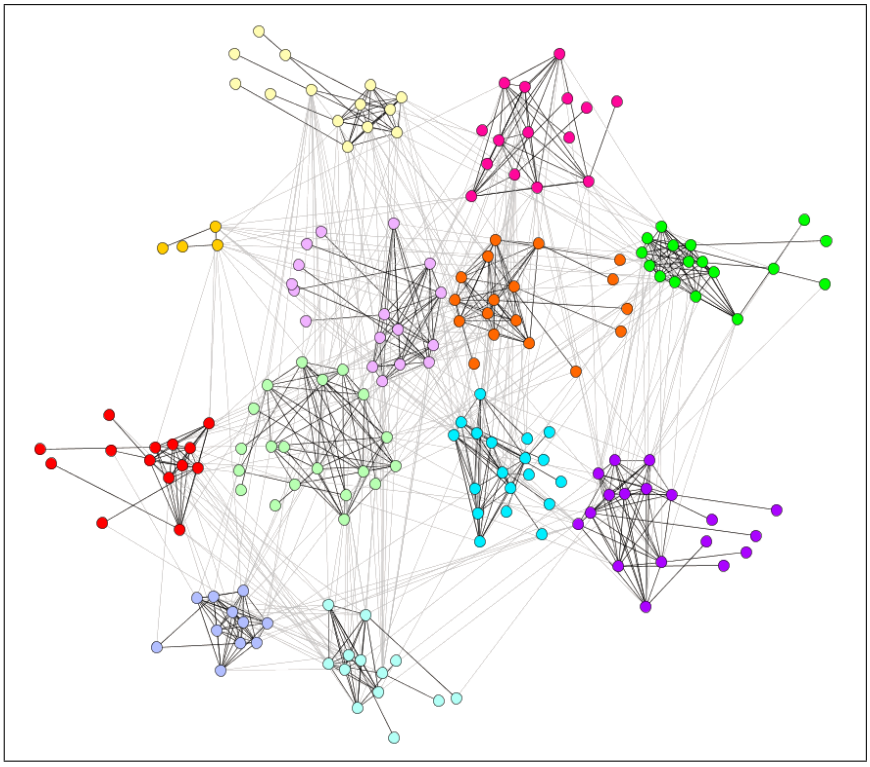


(a) Kamada-Kawai

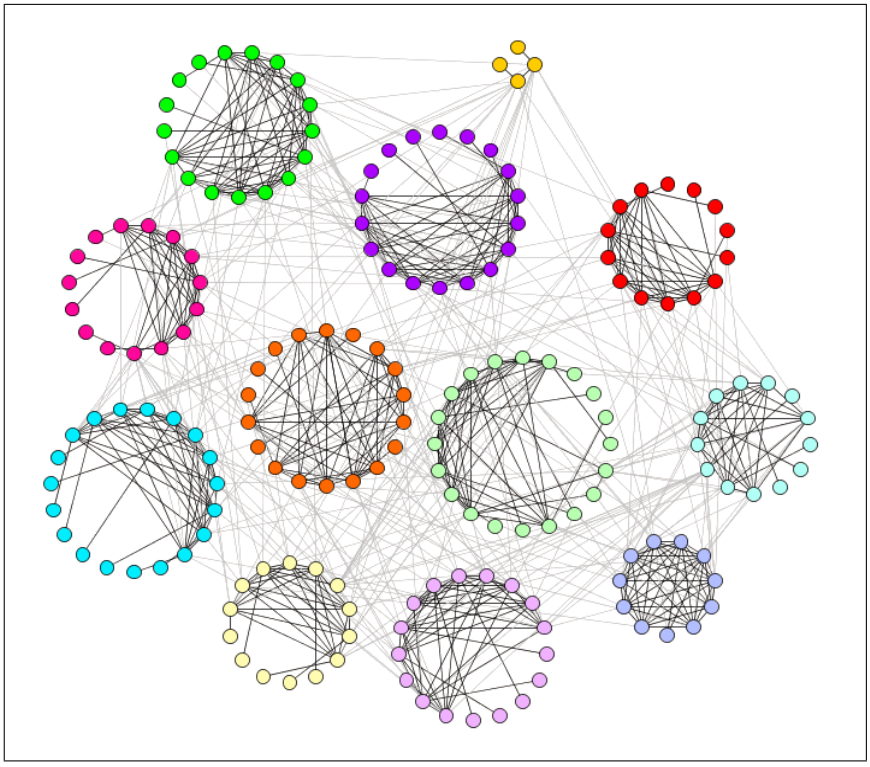
Figure 17.6: Political Books Network.

To include our edge bundling technique into the complexity we only need to note that each edge must be traversed once to determine which communities it connects. Thus our final time complexity remains at $O(|V^2| + |E|)$.

A list of the running times for each algorithm, computed on an Intel i7-2540m with 8GB of RAM, is shown in Table 17.1. We can see that COMB and COMA are equivalent to the existing FR algorithm in terms of efficiency, and that COMC is significantly faster than the other layouts. We believe this efficiency allows one to use the layout in an interactive setting, such as displaying the evolution of communities in dynamic networks. We should note that the Kamada-Kawai algorithm appears to be much slower because it spends a considerable amount of time making minor adjustments to the layout.

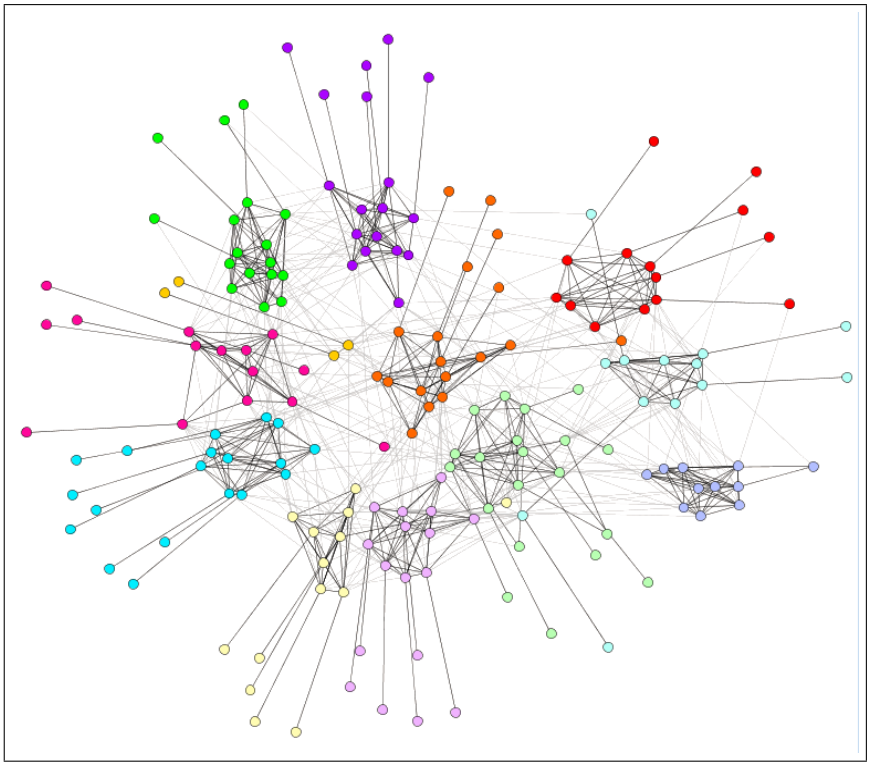


(a) COMB

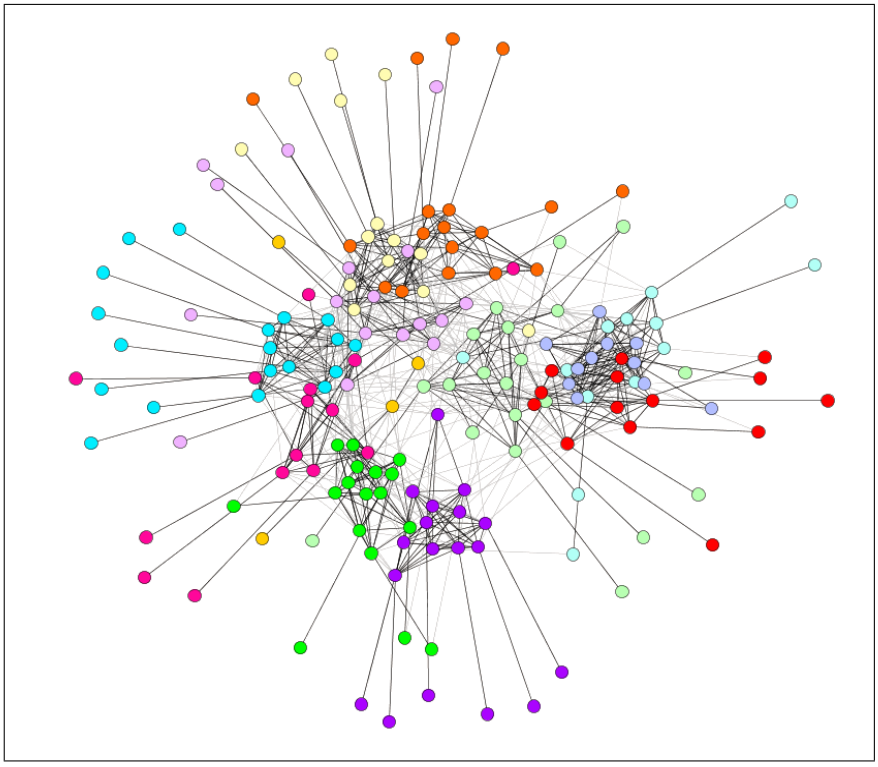


(b) COMC

Figure 17.7: NCAA Football Network.

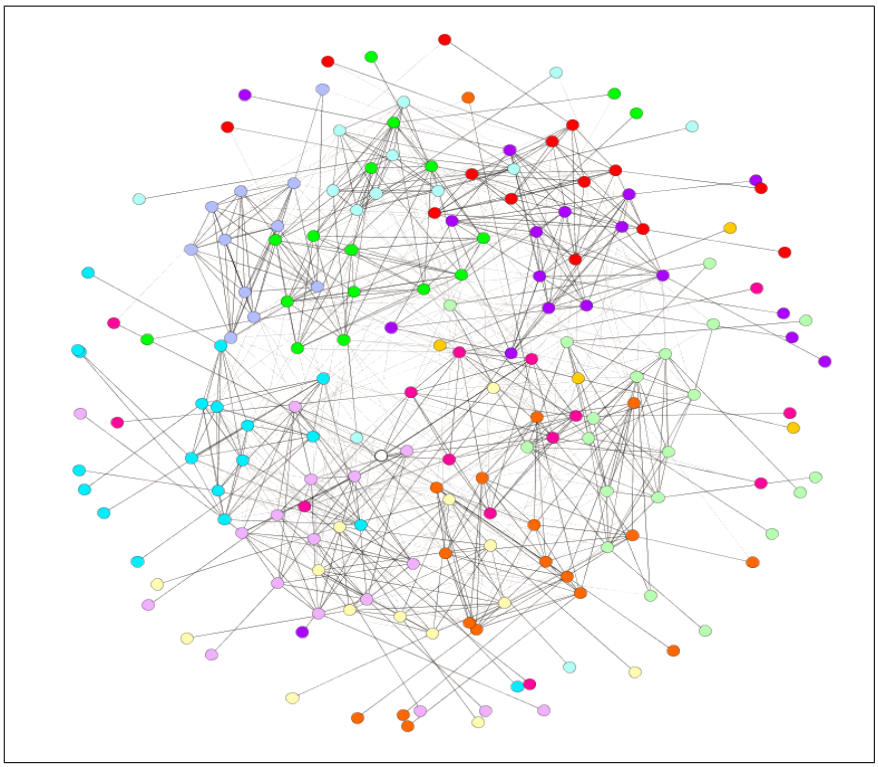


(a) COMA



(b) Fruchterman-Reingold

Figure 17.8: NCAA Football Network.



(a) Kamada-Kawai

Figure 17.9: NCAA Football Network.

Network	V	E	COMB/COMA	COMC	FR	KK
Karate Club	34	77	143 ms	85 ms	130 ms	110 ms
Politcal Books	105	441	7567 ms	375 ms	7181 ms	19752 ms
NCAA Football	180	787	13585 ms	411 ms	14258 ms	22530 ms

Table 17.1: Runtimes of each layout algorithm averaged over 10 runs.

Part V

Conclusion and Open Problems

Chapter 18

Conclusion and Open Problems

18.1 Conclusion and Summary

In this thesis we have proposed three novel algorithms that employ visual or structural analysis to extract meaningful information from social networks. These algorithms were motivated by a bevy of real-world applications that will benefit from being able to more accurately identify and evaluate community structure. In each section we addressed one or more of our thesis statements and contributed a survey, an evaluation, or a new algorithm, to the field of social network analysis.

In part one of our thesis we motivated our work by presenting a brief survey of the existing static and dynamic community mining algorithms. We also discussed the daunting evaluation challenges facing the field of social network analysis.

In part two we addressed our first thesis statement by transforming and enabling criteria from the field of data clustering to work in the context of social network analysis. The list of criteria we considered includes the well-known Silhouette Width Criterion, the Variance Ratio Criterion, and the Dunn Index. We also included the well known Modularity metric from social network analysis. We performed an experiment to determine which criteria could best predict the accuracy of a community mining result without needing to consult the ground truth.

Our results showed that overall, Modularity was the most accurate, followed closely by the Ball and Hall criterion. We further analyzed the results by grouping our experimental networks into three difficulty levels and showed that some criteria perform better on 'easy' networks than on 'medium' networks. Finally, we argued

that none of criteria can serve as suitable replacement to a ground truth on anything but the easiest of networks. Thus our first thesis statement proved to be false.

In part three of the document we explored our second and third thesis statements by presenting a novel community mining algorithm based on our T metric. In addition, we also proposed modifications to Clauset's local framework in order to achieve improved outlier detection and better starting node selection when compared to previous approaches. We then performed a rigorous evaluation against a variety of existing community mining algorithms and showed that our method outperforms all of these algorithms on a variety of ground truth networks.

Furthermore, we showed that our starting node selection method is superior to the current practice of randomly selecting a node, and that our outlier detection can correctly detect outliers, or the absence of, in all of the ground truth networks we evaluated. Thus, we can conclusively say that by utilizing triads, as per our second thesis statement, and by including outlier detection, as per our third thesis statement, we have proposed a community mining algorithm that is significantly more accurate than the current state-of-the-art algorithms.

Finally, in part four, we addressed our fourth thesis statement by presenting our COMmunity Attraction (COMA), COMmunity Circles (COMC), and COMmunity Boundary (COMB) layout algorithms. These algorithms aim to generate aesthetically pleasing network layouts while highlighting both the structure of each community and the relationships between the communities. To accomplish this goal we modified the existing Fruchterman-Reingold (FR) [24] algorithm by removing the need for bounding borders, adding support for vertices with non-zero size, and either enabling the definition of bounding circles within the layout or modifying the repulsion/attraction functions. Furthermore we borrowed concepts from Gansner et al. [25] and presented our simplified edge bundling technique to reduce visual clutter in the final visualization.

Together, these modifications allowed us to create stark visual borders around each community (COMB), highlighted the structure of the communities (COMA), or reduced the overall time-complexity (COMC), while still generating an aesthetically pleasing layout. In our evaluation section we argued that COMB, COMC, and

COMA offer more insight into the structural components of each community, and the relationships between the communities, when compared to the existing techniques. We believe that our visualizations will allow researchers and analysts to more quickly identify communities of interest without needing to scour endless tables of statistics, thus confirming our fourth thesis statement.

18.2 Contributions

This thesis makes the following contributions:

1. A thorough discussion of data clustering criteria in the context of evaluating community mining results in social networks. Our evaluation revealed that although there are similarities between the fields, the criteria themselves do not outperform the existing social network analysis metrics nor do they act as suitable replacement for ground truth networks.

2. A novel community mining algorithm based on our T metric that uses a local framework to accurately identify outliers and hub nodes in the network. Our evaluation shows that our algorithm achieves better accuracy than the current state-of-the-art.

3. Our COMmunity Boundary (COMB) algorithm, which generates aesthetically pleasing layouts of networks and highlights the relationships between the discovered communities.

4. Our COMmunity Circles (COMC) algorithm, which showcases inter-community edges while ensuring a reduced time-complexity.

5. Our COMmunity Attraction (COMA) algorithm, which reveals community structure without sacrificing the aesthetic quality of the layout.

18.3 Open Problems

In this thesis we proposed a number of new algorithms for the area of community mining in the burgeoning field of social network analysis. However, there are still

many open problems remaining in this area that, once solved, may further enhance the quality or evaluation of the identified communities.

For example, most of the current methods in community mining focus on identifying communities that do not overlap. Thus a node cannot be a member of more than one community. Yet, there are a variety of real-world applications where overlapping communities are desired, such as in friendship or disease networks. Many algorithms, including our own T metric, can be trivially extended to support overlapping communities, but there is no known ground truth for overlapping networks. As such, it is extremely difficult to provide a convincing evaluation of any overlapping algorithm and thus authors shy away from publishing these results. Proposing a conclusive and convincing overlap evaluation framework would greatly increase the visibility and attention that the overlap problem receives.

Furthmore, as we discussed in Part 3, we believe that deciding how to select a starting node in a local community mining framework is an entirely open problem that warrants further study. In particular, one could be interested in discovering what structural properties of node result in it being a good seed node for a community and how these are properties related to existing metrics, such as PageRank, Degree, or Betweenness. Perhaps possible insights can be gained by studying how database researchers have solved the similar problem of selecting cluster centroids; but it is not immediately obvious that these techniques could easily be applied within the context of community mining.

Finally, we are completely unsatisfied with the current lack of methods/metrics that facilitate the evaluation of visualization algorithms. A good starting point would be to identify methods or metrics that determine how a ‘good’ visualization layout would differ from a ’bad’ one. Perhaps one could use some measure of edge length, visual density, or a yet to be discovered metric. We find that it is a daunting task to present a new layout algorithm when there is no agreed upon methodology to determine if some research is producing better or worse results than the existing algorithms.

Bibliography

[1] L. Adamic and N. Glance. The political blogosphere and the 2004 us election. In *in Proceedings of the WWW-2005 Workshop on the Weblogging Ecosystem*, 2005.

[2] J.M. Anthonisse. The rush in a directed graph. *Stichting Mathematisch Centrum*, 1971.

[3] S. Asur, S. Parthasarathy, and D. Ucar. An event-based framework for characterizing the evolutionary behavior of interaction graphs. In *KDD '07*, pages 913–921, New York, NY, USA, 2007. ACM.

[4] F. B. Baker and L. J. Hubert. Measuring the power of hierarchical clustering analysis. *J Am Stat Assoc*, 70(349):31–38, 1975.

[5] G. H. Ball and D. J. Hall. Isodata, a novel method of data analysis and pattern classification. 1965.

[6] M. Balzer and O. Deussen. Level-of-detail visualization of clustered graph layouts. In *APVIS'07*, pages 133–140, 2007.

[7] J. C. Bezdek and N. R. Pal. Some new indexes of cluster validity. *IEEE Trans Syst Man Cybern B*, 28(3):301–315, 1998.

[8] R. B. Calinski and J. Harabasz. A dentrite method for cluster analysis. *Commun Stat*, 3:1–27, 1974.

[9] W. Chauvenet. A manual of spherical and practical astronomy v. ii. In *Technometrics 11*, 1863.

[10] J. Chen, J. Fagnan, R. Goebel, R. Rabbany K., F. Sangi, M. Takaffoli, E. Verbeek, and O. Zaïane. Meerkat: Community mining with dynamic social networks. *Data Mining Workshops (ICDMW)*, pages 1377–1380, 2010.

[11] J. Chen, O. Zaïane, and R. Goebel. Detecting communities in social networks using max-min modularity. *SIAM International Conference on Data Mining (SDM'09)*, 2009.

[12] J. Chen, O. Zaïane, and R. Goebel. Local community identification in social networks. In *ASONAM*, pages 237–242, 2009.

[13] A. Clauset. Finding local community structure in networks. *Phys. Rev. E*, 72(2):026132, Aug 2005.

[14] A. Clauset, M.E.J. Newman, and C. Moore. Finding community structure in very large networks. *Phys. Rev. E*, 70(066111), 2004.

[15] L. Danon, A. Daz-Guilera, J. Duch, and A. Arenas. Comparing community structure identification. *J Stat Mech*, 9(8):P09008, 2005.

[16] D. L. Davies and D. W. Bouldin. A cluster separation measure. *IEEE Trans Pattern Anal Mach Intell*, 1:224–227, 1979.

[17] J. C. Dunn. Well separated clusters and optimal fuzzy partitions. *J Cybern*, 4:95–104, 1974.

[18] D. Eppstein and J. Wang. Fast approximation of centrality. *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, 2001.

[19] S. Fortunato. Community detection in graphs. *Physics Reports*, 486:75–174, 2010.

[20] S. Fortunato and M. Barthélemy. Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, 104(1):36–41, January 2007.

[21] L. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41, 1977.

[22] H. P. Friedman and J. Rubin. On some invariant criteria for grouping data. *J American Stat Assoc*, 62:1159–1178, 1967.

[23] Y. Frishman and A. Tal. Dynamic drawing of clustered graphs. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 191–198, 2004.

[24] T. M. J. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Software: Practice and Experience*, 21:1129–1164, 1991.

[25] E.R. Gansner, H. Yifan, S. North, and Scheidegger C. Multilevel agglomerative edge bundling for visualizing large graphs. *Pacific Visualization Symposium (PacificVis), 2011 IEEE*, pages 187–194, 2011.

[26] J. Gil-Mendieta and S. Schmidt. The political network in mexico. In *in: Social Networks 18*, volume 4, pages 355–381, 1996.

[27] M. Girvan and M.E.J. Newman. *Proc. Natl. Acad. Sci.*, 99:7821–7826, 2002.

[28] F. E. Grubbs. Procedures for detecting outlying observations in samples. In *Technometrics 11*, pages 1–21, 1969.

[29] D. Harel and Y. Koren. Drawing graphs with non-uniform vertices. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '02, pages 157–166, 2002.

[30] J. A. Hartigan. Clustering algorithms. 1975.

[31] D. Holten. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *Visualization and Computer Graphics, IEEE Transactions on*, 12(5):741 – 748, 2006.

[32] L. Hubert and P. Arabie. Comparing partitions. *J Classif*, 2:193–218, 1985.

[33] L. J. Hubert and J. R. Levin. A general statistical framework for assessing categorical clustering in free recall. *Psychol Bull*, 10:1072–1080, 1976.

[34] D. Huffman. A method for the construction of minimum re-dundancy codes. In *Proc. Inst. Radio Eng*, volume 40, pages 1098–1101, 1952.

[35] P. Jaccard. tude comparative de la distribution florale dans une portion des alpes et des jura. *Bulletin de la Socit Vaudoise des Sciences Naturelles*, 37:547–579, 1901.

[36] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31(1):7–15, 1989.

[37] L. Kaufman and P. Rousseeuw. Finding groups in data. 1990.

[38] R. R. Khorasgani, J. Chen, and O. R. Zaiane.

[39] V. Krebs. http://www.orgnet.com, January 2012.

[40] A. Lancichinetti. https://sites.google.com/ site/andrealancichinetti/cvis, January 2012.

[41] A. Lancichinetti and S. Fortunato. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Phys. Rev. E*, 80(1):016118, Jul 2009.

[42] A. Lancichinetti and S. Fortunato. Community detection algorithms: a comparative analysis. *Phys. Rev. E*, 80:056117, 2009.

[43] A. Lancichinetti, S. Fortunato, and J. Kertsz. Detecting the overlapping and hierarchical community structure of complex networks. *New Journal of Physics*, 11(033015), 2009.

[44] A. Lancichinetti, S. Fortunato, and F. Radicchi. Benchmark graphs for testing community detection algorithms. *Phys. Rev. E*, 78(4):046110, Oct 2008.

[45] J. Leskovec, J. Lang, and M. W. Mahoney. Empirical comparison of algorithms for network community detection. In *WWW*, August 2010.

[46] F. Luo, J. Z. Wang, and E. Promislowl. Exploring local community structures in large networks. In *WI 06: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 233–239, 2006.

[47] J. Madadhain, D. Fisher, P. Smyth, S. White, and Y.B. Boey. Analysis and visualization of network data using jung. *Journal of Statistical Software*, 10:1–35, 2005.

[48] J. O. McClain and V. R. Rao. Clustisz: a program to test for the quality of clustering of a set of objects. *J Mark Res*, 12:456–460, 1975.

[49] J.H. Michael. Labor dispute reconciliation in a forest products manufacturing facility. In *Forest Products Journal*, volume 47, pages 41–45, 1997.

[50] G. W. Milligan and M. C. Cooper. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50(2):159–179, 1985.

[51] M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Phys. Rev. E*, 69(6):066133, Jun 2004.

[52] M. E. J. Newman. Modularity and community structure in networks. *Proc. Natl. Acad. Sci.*, 103(23):8577–8582, 2006.

[53] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phys. Rev. E*, 69(2):026113, Feb 2004.

[54] Chris North. Toward measuring visualization insight. *IEEE Comput. Graph. Appl.*, 26(3):6–9, 2006.

[55] M. K. Pakhira, S. Bandyopadhyay, , and U. Maulik. Validity index for crisp and fuzzy clusters. *Pattern Recognit Soc*, 37:487–501, 2004.

[56] G. Palla, I. Derenyi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435:814, Jun 2005.

[57] B. Peirce. Criterion for the rejection of doubtful observations. In *Astronomical Journal II 45*, 1852.

[58] Catherine Plaisant. The challenge of information visualization evaluation. In *Proceedings of the working conference on Advanced visual interfaces*, AVI '04, pages 109–116. ACM, 2004.

[59] D. A. Ratkowsky and G. N. Lance. A criterion for determining the number of groups in a classification. *Aust Comput J*, 10:115–117, 1978.

[60] M. Rosvall and C. T. Bergstrom. In *Proc. Natl. Acad. Sci*, volume 105, pages 1118–1123, 2008.

[61] P. J. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *J Comput Appl Math*, 20:53–65, 1987.

[62] M. Takaffoli, F. Sangi, J. Fagnan, and O. Zaïane. A framework for analyzing dynamic social networks. In *ASNA*, September 2010.

[63] Q.D. Truong, T. Dkaki, and P.J. Charrel. An energy model for the drawing of clustered graphs. In *Proceedings of Veme colloque international VSST*, 2007.

[64] L. Vendramin, R. Campello, and R. Hruschka. Relative clustering validity criteria: A comparative overview. *Statistical Analysis and Data Mining*, 3(4):209–235, 2010.

[65] L. Wan, J. Liao, and X. Zhu. Cdpm: Finding and evaluating community structure in social networks. In *Advanced Data Mining and Applications*, volume 5139, pages 620–627. Springer Berlin / Heidelberg, 2008.

[66] W.W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33:452–473, 1977.