Leaders, Followers, and Community Detection

by

Dhruuv Parthasarathy

B.S., Electrical Engineering, MIT, 2013

Submitted to the Department of Electrical Engineering and Computer Science in Partial Fulfillment of the Requirements for the Degree of Master of Engineering in Electrical Engineering and Computer Science at the Massachusetts Institute of Technology

[JUNE 2014]

May 2014

Signature of Author: Signature redacted

Dhruv Parthasarathy
Department of Electrical Engineering and Computer Science
March 18, 2015

Certified by: Signature redacted

Devavrat Shah
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by: Signature redacted

Prof. Albert R. Meyer
Chairman, Masters of Engineering Thesis Committee

Leaders, Followers, and Community Detection

by Dhruv Parthasarathy

Submitted to the Department of Electrical Engineering and Computer Science

May 16, 2014

In Partial Fulfillment of the Requirements for the Degree of Master of Engineering in Electrical Engineering and Computer Science

Abstract

Communities in social interaction networks or graphs[1] are sets of well-connected, and very often overlapping vertices. Formally, we view any maximal clique of the social network graph as a community. The problem of finding maximal cliques is known to be computationally hard. The goal of this work is to identify structural conditions in social network graphs that lead to efficient identification of maximal cliques, i.e. overlapping communities.

We propose an evolutionary model called *sequential community graphs* for community formation in social networks. In a sequential community graph, each node enters the graph by either joining an existing community, or creating its own. To discover communities, i.e. maximal cliques, in such graphs, we present the non-parametric Iterative Leader-Follower Algorithm (ILFA). We establish that the ILFA finds all the communities/maximal cliques correctly in the sequential community graph model in polynomial time in the number of vertices in the graph.

To scale to very large data sets, we propose a minor simplification of the ILFA, called the fast leader-follower algorithm (FLFA) which effectively runs in linear time in the input data size, and finds all communities correctly for sequential community

[1] In this paper, we shall use terms social network and graph interchangeably.

graphs with an additional constraint. Empirically, the FLFA and IFLA perform nearly the same in terms of accuracy, but the FLFA runs nearly three orders of magnitude faster.

We find that the sequential community graph model is a good fit for a wide variety of social networks where users can be modeled as entering the graph by joining existing communities or creating their own. In such social networks, we demonstrate that the FLFA and ILFA outperform other state of the art algorithms both in terms of speed and accuracy. For example, in the Internet Movie Database (IMDB) graph where communities naturally correspond to actors in the same movie, our algorithms finds nearly all ground truth communities correctly while all other known community detection algorithms do very poorly. Similar empirical results are found for various other social data sets. This supports our hypothesis that we can model many social graphs as sequential community graphs and accurately detect their communities using the ILFA or FLFA.

Acknowledgments

I'd like to acknowledge all the people who have been fundamental in helping me through this process over the last year. In particular, Professor Shah and Professor Zaman have provided incredible guidance and support in helping me every step of the way. Additionally, multiple fellow students such as Bonny Jain, Pranav Ramkrishnan, and Kuang Xu were always willing to listen to the latest developments and provide incredibly constructive feedback. I'd also like to thank MIT for giving me the opportunity and support to carry out this research and expand my knowledge. This has been a wonderful experience and I'm truly thankful for it.

Contents

Abstract 3

Acknowledgments 5

List of Figures 9

1 Introduction 11

2 Prior Work 15

3 Community Detection 19

3.1 Community Detection 19

4 Leader Follower Algorithms 27

4.0.1 Fast Leader-Follower Algorithm 27

4.0.2 Iterative Leader-Follower Algorithm 30

4.0.3 Performance Guarantees 34

5 Empirical Evaluation 41

5.0.4 Data Description 43

5.0.5 Experimental Results 44

Accuracy 45

Runtime 46

Robustness 47

6 Conclusion **49**

Bibliography **51**

List of Figures

3.1 A latent community bipartite graph (top left) and its corresponding observation graph (top right). The follower vertices are colored black and the communities in the observation graph are circled with dashed lines. (bottom) Example of spurious communities created by an unobserved edge in the observation graph. 19

3.2 Illustration of two sequential community graph constructions (and the sequential construction of the corresponding observation graph) resulting in the same observation graph. In each construction the member vertices are listed in order of addition to the graph with the newest vertex at the bottom. 22

4.1 Application of FLFA to a graph with three communities. (top) The figures show each new community that is detected. (bottom) The list of degree ordered vertices has (multi)colored rectangles showing the (possible multiple) community membership of the vertices as new communities are detected. The seeds of each new community are indicated in the vertex lists. 28

4.2 Application of ILFA to a graph with four communities. The first iteration detects the peripheral communities, while the second iteration detects the community at the core of the graph. 30

5.1 Plot of average F1 score versus the average runtime per edge of each algorithm. The dashed line indicates the 0.5 power set score. 44

5.2 Plot of F1 score (top) and number of communities found (bottom) by FLFA on IMDB graph versus edge deletion fraction The dashed line indicates the 0.5 power set score. 46

Chapter 1

Introduction

THE understanding of community structure is an important problem in the analysis of social network graphs. Communities represent a latent structure that is manifested through densely connected vertices. For example, a latent social group such as people in the same college class may show up as a set of densely connected people in a social network such as Facebook. Members of the same community should be connected to each other. Therefore, the fundamental structure of a community in a graph is a maximal clique. However, in many instances there may only be noisy observations of the graph where certain edges are missing, forcing observed communities to no longer be exact cliques. Therefore, we view community detection as equivalent to maximal clique detection in the presence of noise.

In general graphs, the maximal clique enumeration problem is known to be computationally hard. The number of maximal cliques in a graph can be exponential in the number of vertices [10]. Even finding the size of the largest maximal clique is known to be NP-complete [7]. The question of interest in this paper is *whether finding (nearly) maximal cliques is more tractable in social network graphs* that are generated through simple, modellable, human interactions. We find that maximal cliques/communities in social interaction networks can indeed be found efficiently and correctly for a class of graphs where social interactions can be modeled by a Sequential Community Graph.

Our contributions. To answer this question, we start by hypothesizing a simple,

natural model for community formation in a social interaction network which we call the *sequential community graph model*: starting from one vertex in the graph, vertices are added sequentially; each vertex either joins one or more existing communities in the graph, or forms a new community.

To detect communities or maximal cliques in such a model, we present iterative leader-follower algorithm (ILFA) which finds communities in time polynomial in number of vertices in the graph. Precisely, the running time is bounded by $O(|V||E|(|E| + |V|\log|V|))$ for graph with vertex set V (see Theorem 4). We establish that ILFA finds all communities in the sequential community graph model correctly (see Theorem 2).

For the algorithm to scale for very large data sets, we propose a minor simplification of the ILFA, called the fast leader-follower algorithm (FLFA) which runs in time $O(|E|+|V|\log|V|)$ for graphs with edge set E and vertex set V, i.e. FLFA is effectively linear in the input data size (see Theorem 3). We establish that the FLFA finds all communities correctly for sequential community graphs with an additional constraint (see Theorem 1).

We do an extensive empirical evaluation of our algorithm for various social data sets where *ground truth* communities are known so that we can verify the performance of the algorithm. Concretely, we report the performance on three social interaction networks in which communities are generated due to a shared activity. First, we study the IMDB dataset where actors are vertices, edges between two actors indicate that they have acted in one or more movies together, and a community corresponds to a movie. Next, we look at the Les Miserable data set where characters in the novel Les Miserbales are vertices, edges between two actors indicate that they were in the same chapter of the novel, and a community corresponds to a chapter. Finally, we also use the MIT cultural show dataset where show participants are vertices, edges between two participants indicate that they were in one or more performances together, and a

community corresponds to a performance.

For all of these datasets, we compare the performance of our algorithms (ILFA and FLFA) with representative known algorithms. Namely, modularity optimization [11], CESNA [19], and BigClam [18]. We compare with the classical spectral clustering (and a few other algorithms), but we do not report the results here as for all of these datasets they perform poorly compared to these other representative algorithms.

We find that, our algorithms (IFLA/FLFA) find all or nearly all communities correctly for all of the three social datasets. The ILFA algorithm performance is slightly better than FLFA in terms of its accuracy in finding communities. However, the FLFA algorithm, runs orders of magnitude faster than ILFA, as expected based on theoretical bounds on their running times. In contrast, modularity optimization, BigClam and CESNA perform poorly both in terms of accuracy, and in terms of speed (quantified precisely in Section ??).

In summary, the ILFA and FLFA outperform other known algorithms for a class of social interaction data sets that can be modeled as Sequential Community Graphs. These graphs are generated when individuals join a social graph by either joining existing communities or creating their own. As we shall see, this is a fairly broad, and applicable range of graphs. For such social data settings, the ILFA and FLFA perform exceptionally well both in terms of speed and accuracy.

A few final remarks. First, structurally the sequential community graph model is very similar to what we shall call the *prime number graph.* In this graph, the integers from 2 to $N > 2$ are vertices, edges between two integers indicate that hey share a prime number as a common factor (e.g. 14 and 21 have an edge since they have 7 as a common factor), and a community corresponds to a prime number in the sense that it is a collection of integers all of which have a given prime as their factor (e.g. all integers that contain 7 as a factor). It can be easily established that the ILFA and

FLFA will find all communities (prime numbers) correctly for such a graph. And in a sense, this graph captures the essence of the sequential community graph for which it is easy to identify the overlapping communities using the ILFA and FLFA in an entirely non-parametric manner. Second, not all social interaction graphs need to arise through mechanisms in the sequential community graph model. For such social datasets, the ILFA and FLFA may not be good fits. Therefore, like any other community detection algorithm, a degree of caution needs to be exercised in using the ILFA and FLFA by carefully thinking about the reasons why a social interaction graph exists and what is the underlying nature of the community structure.

Chapter 2

Prior Work

COMMUNITY detection in a graph is a specific instance of the more general problem of clustering. The goal of a good clustering algorithm is to create clusters of data points such that the points within each cluster are close to each other and points in different clusters are far from each other in an appropriate metric. In community detection, the data points are the vertices in a graph and the metric is the graph distance.

One approach to community detection is to cut the graph into disjoint communities of reasonably large size. This is done by minimizing functions such as RatioCut [6] and the normalized cut [13], which are graph-cuts weighted by the size of the communities. These functions are known to be NP-hard, but they can be approximately minimized using spectral clustering [13], [6],[15], [16],[9]. Spectral clustering maps the vertices of a graph into points in a finite dimensional space and then uses common clustering algorithms such as k-means to separate the vertices in this new space. In this way, it groups together vertices that are located close to each other in the finite-dimensional space.

Another approach to community detection is known as modularity optimization [11]. Modularity is a function which maps a partition of vertices in a graph to a number. The intuition for modularity is that if two vertices are in a community, then they are more likely to be connected than two vertices in a random graph. Optimizing modularity is

known to be NP-hard. However, fast approximation algorithms for it do exist [2]. In addition, it has been found that modularity has difficulty resolving small communities [8].

Most of the above describes algorithms that produce communities or clusters as disjoint sets. However, in social setting, one expects communities to be overlapping as a single node may be part of multiple communities. To that end, modularity optimization was modified by [2] to produce a hierarchy of overlapping communities.

When community is defined as a clique, as in this paper, a natural method that comes up is the k-clique percolation [12] where the goal is to find overlapping cliques using input parameter k and the output is produced by maximizing an appropriate objective.

Multiple community membership can also be viewed as a 'mixture probabilistic model' and this led to a statistical inference framework for finding overlapping communities [1]. A model-based overlapping community detection algorithm is proposed in [18] which only uses the graph structure. Vertex features and graph structure are combined to find overlapping communities in [19]. Such methods can suffer in terms of scaling with data size due to the inference task, but in [18] and [19] approximations are made to scale up to larger datasets. In addition to computational issues, the validity of the mixture model needs verification.

The algorithms in [2],[18],[19] are shown to scale to very large graphs, which is a crucial requirement for any practical community detection algorithm. This is the primary reason why we use them as candidate representative algorithms to compare against.

It should be noted that another requirement is to learn the community structure without any input parameters. The method of [2] is non-parametric and finds the number of communities naturally from the graph structure. However, in [18], [19] the

number of communities is an input parameter and different values are tried until a good fit to the data is found.

Chapter 3

Community Detection

3.1 Community Detection

The problem. We are given an undirected graph $G = (V, E)$ where $V = \{1, \ldots, n\}$ represents vertices and $E \subset V \times V$ represents edges between them. Any maximal clique of G is defined as a community of G. The goal is to find all communities or maximal cliques of G. Recall that a subset C of vertices V is called clique if $\{(i, j) : i \neq j \in C\} \subset E$; it is a maximal clique if no $C' \subset V$ such that $C \subset C'$ is a clique as well.

The way to think of G is as observed interactions, represented by E, between vertices in V due to an underlying *latent* or *unobserved* community structure in G. Precisely, let $\mathcal{G} = (V, \mathcal{C}, \mathcal{E})$ represent the latent community bipartite graph, where one partition of vertices is V, the vertices we observe in G and the other partition of vertices is $\mathcal{C} = \{c_1, \ldots, c_m\}$ that represents the m communities. The edges $\mathcal{E} \subset V \times \mathcal{C}$ are between these two partition, i.e. $\mathcal{G}$ is bipartite. The edges of $\mathcal{E}$ represent membership of vertices of V in communities of $\mathcal{C}$: $(i, c) \in \mathcal{E}$ if vertex i belongs to community c.

Finally, the observation graph $G = (V, E)$ is a *projection* of $\mathcal{G}$: $(i, j) \in E$ if and only if vertices $i, j \in V$ share one or more community in $\mathcal{G}$, i.e. there exists $c \in \mathcal{C}$ such that $(i, c), (j, c) \in \mathcal{E}$. We illustrate these graphs in Figure 3.1.

It is well known that the problem of finding maximal cliques in any G is computationally hard, as noted earlier. Note that for any given G, it is feasible to find a $\mathcal{G}$ so

that G becomes the corresponding projection of $\mathcal{G}$. In that sense, the above view is not restrictive and hence the problem of find communities in G even when G is projection of a latent community bipartite graph is equally hard.

The question of interest: *are there prevalent social phenomenon generating $\mathcal{G}$ for which finding communities in G is easy?* To answer this question, we shall present the sequential community graph model next.

Few remarks are in order before we present the model. First, our problem formulation as well as the latent community bipartite graph has been considered before, cf. [3], [14], [4], [17]. Second, In practice there may be missing edges or noise in an observation graph. In this case, communities would no longer be cliques. For instance, if an edge is removed from a clique with n vertices, then it becomes the union of two overlapping cliques each with $n-1$ vertices. Therefore, noise or missing edges in an observation graph will result in the creation of spurious community vertices in the corresponding latent community bipartite graph. We illustrate this in Figure 3.1. For the purposes of establishing theoretical results, we shall assume that G is perfectly observed. However, as we shall see, our algorithms work for noisy data as well.

Sequential Community Graph Model. Here we present the sequential community graph model for the evolution of a community bipartite graph $\mathcal{G}$. This should be treated as a *social hypothesis* applicable to a class of social scenarios. In particular, this model is relevant to settings where individuals enter a social graph by either joining existing communities or creating their own. We now present the model in detail.

Let $\mathcal{G}_n = (V_n, \mathcal{C}_n, \mathcal{E}_n)$ denote the community bipartite graph with n observed vertices, i.e. $|V_n| = n$. This is generated sequentially as follows. Initially, $n = 1$ and $V_1 = \{1\}$, $\mathcal{C}_1 = \{c_1\}$ and $\mathcal{E}_1 = \{(1, c_1)\}$. Given $\mathcal{G}_i$, the $\mathcal{G}_{i+1}$ is generated by adding vertex $i+1$ to V_i, i.e. $V_{i+1} = V_i \cup \{i+1\} = \{1, \ldots, i+1\}$. For $\mathcal{C}_{i+1}, \mathcal{E}_{i+1}$, one of the two choices listed below is exercised arbitrarily:

Choice 1. Choose a single community, $c \in \mathcal{C}_i$; add edge $(i+1, c)$ to $\mathcal{E}_i$ to obtain $\mathcal{E}_{i+1}$.

Choice 2. Add a new community vertex c' to $\mathcal{C}_i$ to obtain $\mathcal{C}_{i+1}$ and add a new edge $(i+1, c')$ to $\mathcal{E}_i$ to obtain $\mathcal{E}_{i+1}$. Then select any one other community vertex $c \in \mathcal{C}_i$. Let $V_c = \{v \in V : (v, c) \in \mathcal{E}_i\}$ be the neighbors of c and let $V_c' \subset V_c$ (V_c' can also be the empty set). Add edges $\{(v, c') : v \in V_c'\}$ to $\mathcal{E}_{i+1}$.

In a sequential community graph $\mathcal{G}_n$ there can be a maximum of n community vertices because a new community vertex is added only if a new observation vertex is added. Also note that the construction of a sequential community graph is not unique. There can be multiple sequences of vertices that produce a given sequential community graph. We illustrate this with an example in Figure 3.2.

Interpretation. The sequential community graph model corresponds to social phenomena where new members join society by either joining a set of communities or creating a new community. Thus, new communities are only created when new members join the graph. This captures the idea is that each community in society is often created by a single individual who brings together different people who were previously in different communities.

Consider the following example where a sequential community graph might form in real life. Imagine a college campus where communities are defined by clubs/organizations. When new individuals arrive at the college, they either have the opportunity of joining an existing club or creating their own club from scratch. When they create their own club, they recruit existing students to join their club. Note that this directly corresponds to our idea of a sequential community graph where each time a node enters the graph, it must either join a subset of existing communities, or create its own community using a subset of the vertices in the graph. There are many other such simple interactions that can lead to a sequential community graph, and the main characteristic is

that new members must enter a graph either through joining a set of communities or by creating their own.

The sequential community graph model motivates us to divide the vertices in an observation graph into two types: those that belong to a single community and those that belong to multiple communities. We define these vertex types as follows.

Definition 1. *A **leader** is a vertex which belongs to only one community. All other vertices are **followers**.*

We call vertices which belong to a single community leaders because they are the individuals in our model who create communities from scratch and bring together different people. In the college campus example, they are the presidents or the founders of the clubs. Everyone else in the graph is naturally deemed to be a follower.

The construction of a sequential community graph naturally incorporates our notions of leaders and followers. A new community can only be generated by a leader. Followers do not generate new communities when they join the graph. Instead they simply join existing communities. Also, as a sequential community graph evolves, the roles of vertices can change. In particular, leaders can become followers if they join newly generated communities that other leaders have created.

Sequential community graphs are a subset of the more general latent community bipartite graphs. They possess important properties, which have already been discussed informally, which allow us to efficiently recover all of their communities. The properties are stated formally below.

Proposition 3.1.1. *In every sequential community graph, there is always at least one leader.*

Proof. The last community vertex added to a sequential community graph was generated by a leader. The leader will remain a leader because no further communities will be generated.

■

The following important property holds as well.

Proposition 3.1.2. *Let $G = (V, E)$ be the observation graph for a sequential community graph $\mathcal{G} = (V, \mathcal{C}, \mathcal{E})$. If we delete any leader node from G, we obtain another graph $G' = (V', E')$ that is the observation graph of a sequential community graph, $\mathcal{G}' = (V', \mathcal{C}', \mathcal{E}')$.*

Proof. We will show that we can construct another sequential community graph $\mathcal{G}'$ for which G' is the observation graph. Let l be the leader vertex we wish to delete from G to create G'. There are three cases we need to consider:

1. l entered $\mathcal{G}$ by creating a new community c_l and l is the only leader of c_l.
2. l entered $\mathcal{G}$ by creating a new community c_l but l is not the only leader of c_l.
3. l entered $\mathcal{G}$ by joining an existing community c_l.

Case 1

In the first case, l entered $\mathcal{G}$ at step i by creating it's own community c_l and l is the only leader in this community. Let $O_l = \{v \in \mathcal{V} \backslash l \mid (v, c_l) \in \mathcal{E}\}$.Now if $\exists\ c \in \mathcal{C} \backslash c_l$ s.t. $\forall v \in O_l$, $(v, c) \in \mathcal{E}$, then we can construct $\mathcal{G}'$ as follows. Follow the exact same steps in the construction of $\mathcal{G}$ except for the following change:

1. At step i, simply avoid adding l and c_l to the graph and do nothing.

Since $\exists c \in \mathcal{C} \backslash c_l$ s.t. $\forall v \in O_l$, $(v, c) \in \mathcal{E}$, every pair of member nodes in $\mathcal{C}$ correctly share an edge in G' as required. It is easy to see therefore that G' is the observation graph for $\mathcal{G}'$.

Now if this condition does not hold and there is no such c, then we know immediately that $\exists v_1, v_2 \in O_l$ s.t. $\nexists c \in \mathcal{C} \backslash c_l$ s.t. $(v_1, c)(v_2, c) \in \mathcal{E}$. This implies that either v_1 or

v_2 entered the graph at a time step greater than i. Otherwise, if they both entered at the earliest possible time i, then by the SCG construction rules, they would have had to be a subset of the same community. Without loss of generality, say v_1 entered the graph at time step $j > i$. Moreover, let $V_i \subset O_l$ be the subset of O_l that entered $\mathcal{G}$ at time step i. In order to construct $\mathcal{G}'$, we now follow every step in the construction of $\mathcal{G}$ except for the following

1. At time step i, add v_1 to $\mathcal{G}'$ and have v_1 create community node c_l. Moreover, for each $v \in V_i$, add (v, c_l) to $\mathcal{E}'$.

2. At time step j, when v_1 was supposed to enter $\mathcal{G}$, do nothing.

This ensures all nodes in O_l still share a community, c_l, and therefore all pairs of nodes in O_l correctly share an edge in G'. It is therefore simple to verify that G' is the exact observation graph for $\mathcal{G}'$.

Case 2

Now we look at the case where l is not the only leader in c_l. Let l' be another leader of c_l and let i' be the step at which it is added to $\mathcal{G}$. We can construct a valid sequential community graph $\mathcal{G}'$ as follows. Follow the exact same process in the creation of $\mathcal{G}$ with the following changes. First, at time step i, instead of adding in nodes l and c_l as we did in the construction of $\mathcal{G}$, add nodes l' and c_l. Next, skip step i', when we were supposed to add in l'. Since all the steps used in the construction of $\mathcal{G}$ are valid steps in the construction of a sequential community graph , and the two steps we introduced are also valid steps in the construction of a sequential community graph , all steps used to construct $\mathcal{G}'$ are valid steps and $\mathcal{G}'$ is in fact a sequential community graph. It is again apparent that G' is the observation graph for $\mathcal{G}'$ as the structure of $\mathcal{G}'$ is identical to that of $\mathcal{G}$ with the only difference being that l is no longer in the graph.

Case 3 Finally, we consider the case where l joined an existing community, c_l upon

entering the graph at time step i. In this case, we can create $\mathcal{G}'$ by following the exact same steps used in the construction of $\mathcal{G}$, with the only change being that we skip step i and do not add l to the graph. It is clear that $\mathcal{G}'$ is a valid sequential community graph and moreover that G' will be the observation graph for $\mathcal{G}'$ for the same reasons used in the earlier cases.

Hence, G' is indeed the observation graph for an SCG $\mathcal{G}'$, as desired.

■

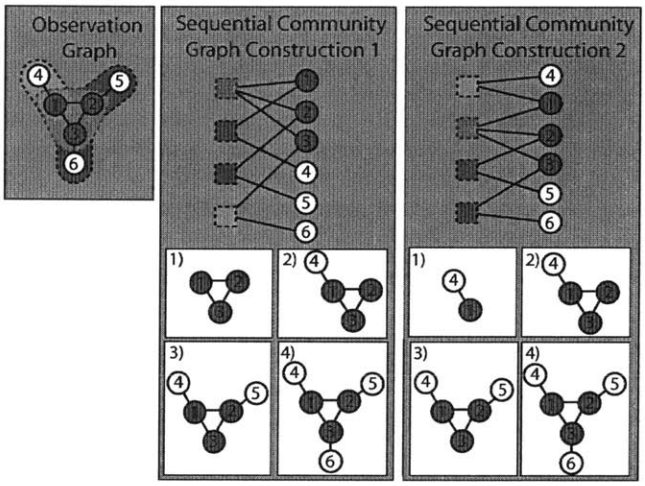

Figure 3.2. Illustration of two sequential community graph constructions (and the sequential construction of the corresponding observation graph) resulting in the same observation graph. In each construction the member vertices are listed in order of addition to the graph with the newest vertex at the bottom.

Chapter 4

Leader Follower Algorithms

We use the notion of followers and leaders discussed in Section 3.1 to develop two community detection algorithms: the fast leader-follower algorithm (FLFA) and the iterative leader-follower algorithm (ILFA). The FLFA is a simple procedure which can detect communities very quickly. The ILFA is an iterative procedure which involves running the FLFA and then removing certain vertices from the observation graph. The ILFA can find more communities than FLFA because it is applied iteratively to a transformed observation graph. However, we will see in practice that both algorithms have very similar performance in terms of accuracy, even though the FLFA has a strong advantage in terms of speed.

■ 4.0.1 Fast Leader-Follower Algorithm

We first provide some intuition to how the FLFA works. FLFA uses that fact that each community in an SCG can be identified by finding its leader, or the node whose only community is the given one. Since this leader is only part of a single community, we can just look at the neighbors of the leader to find all the nodes in a given community. Thus, finding the leader associated with a community allows us to find all the members of the community.

But how can we find these leaders? To answer this question, FLFA makes use of the fact that the degree of a leader must be less than or equal to the degree of its

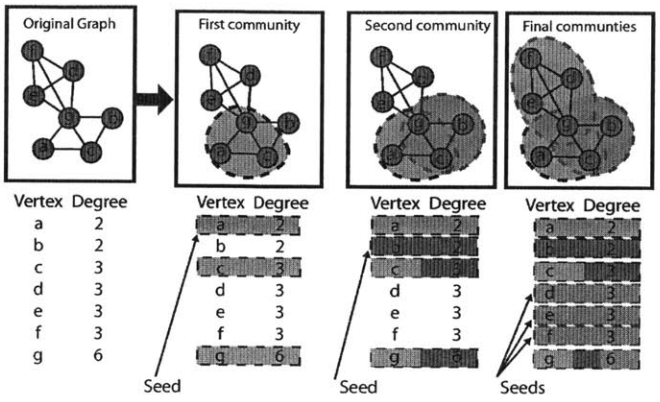

Figure 4.1. Application of FLFA to a graph with three communities. (top) The figures show each new community that is detected. (bottom) The list of degree ordered vertices has (multi)colored rectangles showing the (possible multiple) community membership of the vertices as new communities are detected. The seeds of each new community are indicated in the vertex lists.

neighbors, due to the fact that a leader only has connections to people within a single community, whereas its neighbors can be connected to all members within potentially many communities. Thus, to find leaders, FLFA simply attempts to find the nodes of lowest degree within each community. Once it finds these leaders, it looks at their neighbors to determine the underlying communities in the graph.

FLFA uses a heuristic of node degree to find leaders in a graph quickly. It uses a degree ordered list to sort the nodes in its graph and since leaders have a lower degree than followers, leaders will naturally appear earlier in the list. It then iterates through the list and finds the first node that hasn't been marked visited yet and picks it to be a leader. It marks the given leader, which we shall call a seed from here on out, and all its neighbors as visited, and places them together into a community. Following this, it continues through the list until all nodes have been visited. Note that not all the seeds selected to create communities in this way are necessarily leaders. They instead represent an approximation for what the leaders may be in the graph. As such, the communities that we create in this way are not necessarily cliques.

Since it uses such approximations, FLFA is able to find communities in the graph

extremely quickly using just a single pass. Moreover, it is also succinct and simple in its description and implementation. Lastly, as we shall see in the results section, it still is able to resolve communities with a relatively strong accuracy, despite taking a fraction of the time of some of the other potential algorithms.

The steps of the FLFA are specified below.

procedure FLFA(G)

$i \leftarrow 0$

$communities = []$

$rankedList \leftarrow$ list of vertices sorted by ascending degree

$visited = \{\}$

while $i < length(rankedList)$ **do**

$node \leftarrow rankedList[i]$

if *node* not in *visited* **then**

$C_i \leftarrow node \cup Neighbors(node)$

add C_i to *communities*

add each node in C_i to *visited*

end if

$i \leftarrow i + 1$

end while

return *communities*

end procedure

We illustrate the application of the FLFA to an example graph in Figure 4.1.

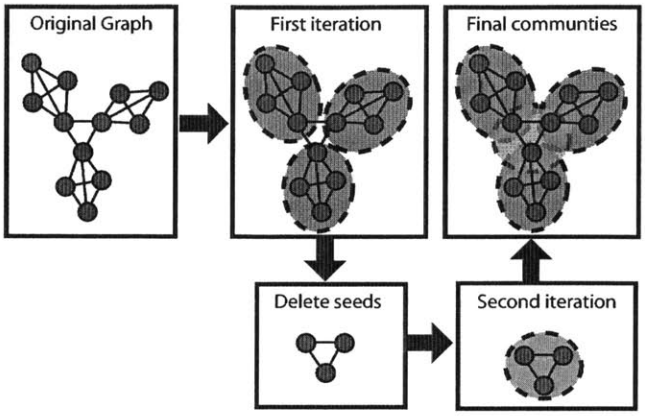

Figure 4.2. Application of ILFA to a graph with four communities. The first iteration detects the peripheral communities, while the second iteration detects the community at the core of the graph.

■ 4.0.2 Iterative Leader-Follower Algorithm

For some graphs the FLFA is not able to find all the communities. For instance, in the graph in Figure 4.2,the central three vertex community will not be found by the FLFA, as this community has no leaders. Every vertex here is a member of another community. If we could modify the graph in a way that transformed some vertices in the undetectable community into leaders, then the community could be detected by the FLFA. This motivates what we call the *iterative leader-follower algorithm* (ILFA) for community detection. This algorithm is designed to detect communities which cannot be found by simple application of the FLFA. The steps of the ILFA are specified below.

procedure ILFA(G)

$communities = []$

$G_M \leftarrow G$

while $G_M \neq \emptyset$ **do**

$newCommunities \leftarrow$

FLFA-M(G_M)

```
        if newCommunities = ∅ then
            break
        else
            for C in newCommunities do
                Remove lowest degree nodes of C from G_M
                if C not subset of any comm in communities then
                    add C to communities
                end if
            end for
        end if
    end while
end procedure
```

procedure FLFA-M(G)

$i \leftarrow 0$

$communities = []$

$rankedList \leftarrow$ vertices of G sorted by ascending degree

$visited = \{\}$

while $i < length(rankedList)$ **do**

$node \leftarrow rankedList[i]$

if $node$ not in $visited$ and $node \cup Neighbors(node)$ is clique **then**

$C_i \leftarrow node \cup Neighbors(node)$

add C_i to $communities$

add each node in C_i to $visited$

end if

$i \leftarrow i + 1$

end while

return $communities$

end procedure

We illustrate the application of the ILFA to a sample graph in Figure 4.2. Essentially, ILFA is an iterative algorithm that peels away the layers of the graph to reveal all hidden communities that FLFA previously could not detect. At each iteration, ILFA runs a modified version of FLFA on the graph, and then removes all leaders from the graph to reveal previously undetectable communities. It then continues to repeat these steps until the graph has no more nodes or there are no leaders in the given iteration.

To see an example of this, consider the graph in Figure 4.2. This graph has a core community of three vertices, each of which is a member of a four vertex community. The core community has no leaders, but the three peripheral communities do. The first

pass of ILFA detects the three peripheral communities. However, it does not detect the three vertex community at the core of the graph because it has no leaders. When we delete the seeds, the resulting graph is a three-clique and all the vertices are leaders. A second iteration of the ILFA detects this community.

The key here is that the reason FLFA wasn't able to detect some communities earlier, is because they had no leaders. As we saw during the construction of the SCG, this occurs when new leaders enter the graph and cause leaders of a given community to become followers. So the key to discovering a leader for these hidden communities is to remove the leaders that caused the leaders of the given community to convert from leaders to followers. This is the underlying intuition behind ILFA.

One important point to note is that ILFA uses a modified version of FLFA, FLFA-M, in its iterations. This modified version has a few key differences. First, it only accepts communities that are actual cliques. Next, it only accepts communities that are not sub cliques of previously detected cliques. In this way, FLFA-M ensures that we only detect true maximal cliques. These precautions are necessary as there are cases when FLFA picks a node to be a seed when it is a follower and not a leader. This can occur when a community has no leaders at all and hence the first node FLFA finds in it is a leader. By checking to see if the community produced is a clique or not, we avoid adding communities from such cases. We also need to ensure that we don't add sub cliques, as sometimes, in subsequent iterations of ILFA we may detect a subset of a previously detected community again. This occurs when we remove the leaders of a given community, but subsequently detect the set of leaders that haven't been removed as an independent community.

With these steps, we obtain a robust algorithm that, as we will see, can exactly discover all the communities in any Sequential Community Graph.

■ 4.0.3 Performance Guarantees

We will next establish theoretical performance guarantees for the ILFA and FLFA. The main results presented here concern the performance of the algorithms in terms of accuracy and speed.

Accuracy. Recall that in the observation graph for a latent community graph, the communities are maximal cliques. This makes community detection equivalent to finding all maximal cliques. The ILFA and FLFA were designed to find these types of communities and their performance is strongest in graphs where communities take this form. In particular, they have very strong performance on sequential community graphs.

We first present our result for the FLFA. There are examples of sequential community graphs where the FLFA cannot find all communities, such as that in Figure 4.2. Therefore, FLFA cannot detect communities on all sequential community graphs. However, there is a subclass of sequential community graphs where the FLFA will detect all communities. Our result is as follows.

Theorem 1. *Let $G = (V, E)$ be the observed graph of a sequential community graph. If every maximal clique of G has a leader, the output of FLFA will be the exact set of maximal cliques in G.*

Proof. We will prove this using a loop invariant. First, let us define $\mathcal{C}_i$ as the set of maximal cliques in G that have at least one leader at position $i-1$ or earlier in *rankedList*, the list of vertices of G sorted by ascending degree. Similarly, let $\mathcal{C}'_i$ be the set of communities identified by the FLFA by the end iteration $i-1$ or earlier.

Invariant

At the beginning of iteration i,

1. $\mathcal{C}'_{i-1} = \mathcal{C}_{i-1}$
2. For any $v \in V$, $v \in$ *visited* iff $v \in C$ where $C \in \mathcal{C}_{i-1}$.

Initialization

Before the algorithm starts, $\mathcal{C}_{i-1} = \emptyset$ (as there are no leaders in the empty list), and indeed $\mathcal{C}'_{i-1} = \emptyset$ as well. Moreover, *visited* is empty, as required.

Maintenance

At iteration i, we are given that $\mathcal{C}'_{i-1} = \mathcal{C}_{i-1}$. Let v be the node at $rankedList[i]$.

If $v \in visited$, then $v \in C$ for some $C \in \mathcal{C}_{i-1}$ and v is not a leader for any maximal clique not in $\mathcal{C}_{i-1}$. Hence, $\mathcal{C}_{i-1} = \mathcal{C}_i$. Moreover, since the FLFA does nothing when v is in visited, $\mathcal{C}'_i = \mathcal{C}'_{i-1}$. So $\mathcal{C}'_i = \mathcal{C}_i$ and the first part of the invariant is preserved in this case. The second part of the invariant is also clearly preserved as we did not add v to *visited* and $\mathcal{C}_{i-1} = \mathcal{C}_i$.

On the other hand, if $v \notin visited$, then it is not a part of any maximal clique in $\mathcal{C}'_{i-1}$. Let C^v be one maximal clique to which v belongs. We now show that v must be a leader for this previously undetected maximal clique C^v. Imagine that v were a follower for C^v. Then the leader for C^v appears at position $i-1$ or earlier as the leader has a lower degree than v, and therefore $C^v \in \mathcal{C}_{i-1}$. But as $v \notin visited$, $C^v \notin \mathcal{C}'_{i-1}$. Hence, $\mathcal{C}'_{i-1} \neq \mathcal{C}_{i-1}$, which contradicts our invariant and is therefore not possible. Thus v must be a leader for C^v.

We now set $C_{new} = v \cup Neighbors(v)$. Since v is a leader and belongs to a single maximal clique, $v \cup Neighbors(v)$ is a maximal clique in G. We then set $\mathcal{C}'_i = \mathcal{C}'_{i-1} \cup C_{new}$. Since C_{new} has a leader at position i or earlier, we see that $C_{new} \in \mathcal{C}_i$, and that $\mathcal{C}_i = \mathcal{C}_{i-1} \cup C_{new}$. Thus, at the end of the iteration, $\mathcal{C}'_i = \mathcal{C}_i$ as required.

Finally, as we place all vertices of C_{new} into *visited*, we see that the second part of the invariant is also preserved.

Termination

Since i increases by 1 after each iteration, i will eventually equal the length of *rankedList* and the algorithm will terminate. When it terminates, $\mathcal{C}_i$ will just be the

set of maximal cliques in G as every maximal clique in G will have a leader at position $length(rankedList)$ or earlier. Thus, as $\mathcal{C}'_i = \mathcal{C}_i$, $\mathcal{C}'_i$, the set of maximal detected by FLFA, will be the exact set of maximal cliques in G.

■

FLFA has correct detection on the subclass of sequential community graphs where each community has a leader, but in many sequential community graphs leaders become followers as the graph evolves. To achieve correct detection for the general class of sequential community graphs we require the ILFA. Our formal result is the following.

Theorem 2. *Let $G = (V, E)$ be the observed graph of a sequential community graph. The output of ILFA will be the exact set of maximal cliques in G.*

Proof. Before proving the theorem, we present a useful definition. Let G_i be the graph G_M, as defined in the pseudocode for the ILFA, at the beginning of iteration i. We now present a useful proposition that will help us prove the theorem.

Proposition 4.0.3. *At the end of iteration i, ILFA deletes exactly the set of leaders in G_i to create G_{i+1}.*

Proof. Let the set of nodes that the ILFA deletes at iteration i be L_i and let $\mathcal{V}_i$ be the set of member nodes assigned to a community by the FLFA-M at iteration i. From the ILFA procedure, L_i is given by $L_i = \{l \in \mathcal{V}_i |\ l$ is a minimum degree node of C for some $C \in \mathcal{C}_{FLFA}\}$, and $\mathcal{C}_{FLFA}$ is the set of communities returned by the FLFA-M. We show that L_i is the exact set of leaders in G_i, and therefore the ILFA deletes the exact set of leaders from G_i to obtain G_{i+1}.

First, we show that every node in L_i represents a true leader in G_i. Choose any $l \in L_i$. Since $l \in L_i$, l is the lowest degree node of some community C returned by FLFA-M. Since C was returned by FLFA-M, C must be a clique, and must moreover possess a leader as FLFA-M only detects cliques that have a leader (the proof for this

is identical to the proof for this statement in the FLFA). Since C possesses a leader (a node that is only part of clique C), and l is the minimum degree node in C, l must be a leader of C. Hence l is a leader in G_i.

Next, we show that every leader in G_i is in L_i. Let l_{true} be a leader in G_i, and let C_{true} be the single clique in G_i to which l_{true} belongs. Using an almost identical proof as that used for the FLFA, we see that FLFA-M will detect every community that has a leader in the graph, and therefore $C_{true} \in \mathcal{C}_{FLFA}$, where $\mathcal{C}_{FLFA}$ is the set of communities detected by FLFA-M. Since l_{true} is a leader of C_{true}, it follows that l_{true} is the lowest degree node of C_{true} and hence $l_{true} \in \{l \in \mathcal{V}_i, \mid l$ is a minimum degree node of C for some $C \in \mathcal{C}_{FLFA}\}$. Thus $l_{true} \in L_i$ and the proof is complete.

■

We now prove the main theorem. We will need a few terms in this proof. First, for any $n \in V$, let the order of n be the iteration i where n is a leader in G_i. Note that n is only a leader once (as it is deleted from the graph after it is found to be a leader), and hence the order of a node is unique. Note that the all leaders in the original graph, G, have order 0. Let the depth of a maximal clique be the order of the minimum order node in the maximal clique. Let $\mathcal{C}_i$ be the set of all maximal cliques in G of depth $i-1$ or lower. Finally, let $\mathcal{C}'_i$ be the set of communities found by ILFA at the beginning of iteration i. We now prove the main statement using a loop invariant.

Loop Invariant

At the beginning of iteration i, $\mathcal{C}'_i = \mathcal{C}_i$.

Initialization

When the algorithm initializes, $\mathcal{C}'_0 = \emptyset$, as no communities have been detected yet, and $\mathcal{C}_0 = \emptyset$ as well as there are no maximal cliques of depth less than 0.

Maintenance

Given that $\mathcal{C}'_i = \mathcal{C}_i$ at the beginning of iteration i, we show that $\mathcal{C}'_{i+1} = \mathcal{C}_{i+1}$ by

the beginning of iteration $i+1$. Let M_i' be the set of communities detected by ILFA during iteration i. Similarly, let m_i be the set of maximal cliques in G with a depth of i. We show that $M_i' = M_i$ and hence $C_{i+1}' = C_{i+1}$.

$M_i \subset M_i$

Take any community $m \in M_i'$. m is a clique as it is detected by ILFA and ILFA only accepts cliques as seen in the pseudocode. m is also not a subset of an community in C_i' as ILFA explicity only accepts commmunities that are not subsets of already detected communities. We show that m is a maximal clique of depth i.

Assume m is not maximal. Then it is a subclique of some maximal clique $m*$. Since m was detected by ILFA, $m = n \cup N(n)$, for some $n \in G_i$, and where $N(n)$ is the set of neighbors of n in G_i. Since $m * nm \neq \emptyset$, $\exists\ l \in m*$ s.t $l \notin G_i$ as otherwise $l \in N(n)$ and hence $l \in m$. Since $l \notin G_i$, and $l \in G$, l must have been deleted at iteration $j < i$ of ILFA. As shown by Proposition 4.0.3, this implies l was a leader in G_j. So $m*$ has depth less than or equal to j, and hence $m* \in C_i$. Since $C_i' = C_i$, $m* \in C_i'$. But as $m \subset m*$, m must then have been a subset of some community in C_i'. This is a contradiction. Thus m must be maximal.

We can also show m is of depth i. Since $m = n \cup N(n)$ and n is a leader of G_i, the depth of maximal clique m is less than or equal to i. If the depth were less than i, then $m \in C_i$ and hence $m \in C_i'$, and so m would be a subset of a community in C_i'. This is once again a contradiction. So the depth of m must be i. Hence $m \in M_i$, and since m was arbitrary, $M \subset M_i$.

$M_i \subset M_i'$

Choose any maximal clique $p \in M_i$. p has a depth of i. Since p is a maximal clique with a leader in G_i, the $FLFA - M$ call in $IFLA$ will detect p, using a nearly identical proof to the one shown in Theorem 1. Therefore, p will not be in M_i' only if p is a

subset of some community in C'_i and is hence not added to M'_i by $ILFA$. p can't be a proper subset of any community in C'_i as every community in C'_i is a clique, and p is a maximal clique. So the only way p could be a subset is if $p \in C'_i$ already. But as $C_i = C'_i$, every community in C'_i has depth less than i and so $p \notin C'_i$. Hence p cannot be a subset of any community in C'_i and so $p \in M'_i$. As p was arbitrarily chosen, $M_i \subset M'_i$.

Hence, we have shown that $M_i = M'_i$ and since $C'_{i+1} = C'_i \cup M'_i$, and $C_{i+1} = C_i \cup M_i$, $C'_{i+1} = C_{i+1}$ at the beginning of iteration $i+1$. So the invariant is preserved.

Termination

At the end of every iteration i, $ILFA$ deletes all leaders from G_i. As seen in Proposition 3.1.1, G_i is an SCG at every iteration. Since any non empty SCG has at least one leader, we delete at least one node of G at every iteration i. Since there are finitely many nodes in G, after finitely many iterations, say k, we will have deleted all nodes from G and $G_k = \emptyset$. Hence, $ILFA$ terminates.

Since every node is deleted by iteration k, every node has order less than k. This means that every maximal clique in G has depth less than k. Thus, C_k represents the exact set of maximal communities in G. Since we have, by our invariant, that $C'_k = C_k$, the set of communities $ILFA$ detects by the time it terminates, C'_k, is exactly the set of all maximal cliques in G. Hence we have shown the correctness of $ILFA$.

■

Runtime. We now analyze the runtime of the FLFA and ILFA. Our first result concerns the runtime of the FLFA.

Theorem 3. *For an input graph $G(V, E)$, the FLFA will terminate in $O(|E|+|V|\log(|V|))$ time.*

Proof. The first step of the FLFA is to calculate the degree of each vertex and sort the vertices by degree. Calculating the degree involves counting every edge in the

graph at most twice which takes $O(|E|)$ time. Sorting the $|V|$ vertices can be done in $O(|V|\log(|V|)$ time. The second step is to go through the degree sorted list and assign each unvisited vertex and its neighbors to a community. This can be done in $O(|E|)$ time. Combining these steps, we find that the a total runtime of the FLFA is $O(|E| + |V|\log(|V|))$.

■

We have the following result for the ILFA runtime.

Theorem 4. *For an input graph $G(V, E)$, the ILFA will terminate in $O(|V||E|(|E| + |V|\log(|V|)))$ time.*

The runtime of the ILFA is determined by the number of iterations it requires to terminate. While the worst case bound in Theorem 4 can be potentially large, we will see in Section **??** that in practice FLFA and ILFA have very similar runtimes on large graphs because not many iterations of FLFA are needed.

Proof. Each iteration of the ILFA involves running the FLFA, checking if each community is a clique, and then deleting seed vertices in the clique communities. Checking if a community is a clique requires counting the edges in the subgraph induced by the vertices in the community. Clearly, checking if a community is a clique will require at most $|E|$ operations. Using this, we find that each iteration of the ILFA will require $O(|E|(|E| + |V|\log(|V|)))$ steps. For a graph of $|V|$ vertices, the maximum number of iterations is $|V|$. Therefore, the worst case runtime of the ILFA will be $O(|V||E|(|E| + |V|\log(|V|)))$. ■

Chapter 5

Empirical Evaluation

We now compare the performance of the ILFA and FLFA to other state of the art community detection algorithms on several real graphs. We compare the performance of the algorithms in terms of accuracy and speed on graphs with known ground truth communities. The algorithms we compare against include the method for fast modularity optimization [2], CESNA [19], and BigClam [18].

To assess the accuracy of the algorithms, we first define a score to compare sets of communities. For any two sets of communities $\mathcal{C}$ and $\mathcal{C}'$ of an observation graph, we define their score as

$$d(\mathcal{C}, \mathcal{C}') = \frac{1}{2}\left(s(\mathcal{C}, \mathcal{C}') + s(\mathcal{C}', \mathcal{C})\right) \tag{5.1}$$

where we have defined $s(\mathcal{C}, \mathcal{C}')$ as

$$s(\mathcal{C}, \mathcal{C}') = \frac{1}{|\mathcal{C}|} \sum_{C \in \mathcal{C}} \max_{C' \in \mathcal{C}'} \delta(C, C')$$

and $\delta(C, C')$ is a similarity measure between communities. There are a variety of similarity measures we can choose, but we will follow the approach of [19] and use the F1 score which is used commonly in binary classification to measure the accuracy of

a test. For two communities C and C', we define the precision $p = |C \bigcap C'|/|C'|$ and the recall $r = |C \bigcap C'|/|C|$. The F1 score is given by the harmonic mean of p and r: $\delta(C, C') = 2pr/(p + r)$. For two identical community sets, the F1 score is one and the minimum value of the F1 score is zero for two disjoint communities.

The quantity $s(\mathcal{C}, \mathcal{C}')$ finds the best match in $\mathcal{C}'$ for every community in $\mathcal{C}$. It then calculates the average similarity score of this matching. Note that multiple communities in $\mathcal{C}$ are allowed to match to the same community in $\mathcal{C}'$ to allow for the possibility that communities in $\mathcal{C}$ are subsets of the same community in $\mathcal{C}'$. The overall score, $d(C, C')$ is simply the average of $s(\mathcal{C}, \mathcal{C}')$ and $s(\mathcal{C}', \mathcal{C})$. To see why our score needs both $s(\mathcal{C}, \mathcal{C}')$ and $s(\mathcal{C}', \mathcal{C})$, consider the case where $\mathcal{C}' = \{a\}$ and $\mathcal{C} = \{a, b, c, d, e, f\}$. If our score only accounted for $s(\mathcal{C}, \mathcal{C}')$, we would obtain a score of 1, even though the communities are quite clearly different. The quantity $s(\mathcal{C}', \mathcal{C}) = 1/5$. Hence, we need to account for the both $s(\mathcal{C}, \mathcal{C}')$ and $s(\mathcal{C}', \mathcal{C})$ to obtain an informative score for two sets of communities.

To understand what constitutes a good value of this score, we consider the set of communities which is the power set of the vertices $\mathcal{P}(V)$. This consists of every possible subset of V. This is an extremely naive community set and provides no information about community structure. We have the following result about our score and the power set communities.

Lemma 5.0.1. *Consider a graph with a ground-truth set of communities $\mathcal{C}$ and power set $\mathcal{P}(V)$. Then we have that $d(\mathcal{C}, \mathcal{P}(V)) \geq 0.5$.*

This shows that the most uninformative community set will score at least 0.5. We will refer to this as the *power set score*. For a community detection algorithm to have good accuracy, it should produce communities which score greater than this value.

Proof. Every set in $\mathcal{C}$ matches exactly with one set in $\mathcal{P}$ and will have an F1 score of one. Therefore $s(\mathcal{C}, \mathcal{P}(V)) = 1$ which immediately leads to $d(\mathcal{C}, \mathcal{P}(V)) \geq 0.5$. ∎

■ 5.0.4 Data Description

Our dataset consist of several graphs for which we have accurate ground truth communities. We describe these graphs below. All properties of the graphs are shown in Table 5.1.

Graph	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert \mathcal{C}\rvert$
Prime number graph	999	195,309	168
Culture show 2010	153	1802	13
Culture show 2011	138	3626	10
Les Miserables	71	244	80
IMDB	382,219	15,038,083	127,823

Table 5.1. Graph properties: Number of member vertices $|V|$, number of edges $|E|$, and number of communities $|\mathcal{C}|$.

Prime Number Graph. The prime number graph was described earlier in Section ??. We use a prime number graph whose vertex set is the integers from 2 to $1,000$. The number of ground truth communities is 168, which is the number of prime numbers less than 1,000. There is great heterogeneity in the community sizes, with some communities constituting half of the vertices, while others being isolated vertices.

Culture Show Graphs. The Culture show 2010 and 2011 graphs represent performances from a college culture show at MIT in 2010 and 2011. The vertices are performers and the edges indicate whether or not two performers were in the same performance. Each performance is a separate ground truth community in this graph.

Les Miserables Graph. The Les Miserables graph captures the social interactions of the characters in the novel Les Miserables. The vertices are characters from the novel and an edge is placed between two characters if they appear in the same chapter of the novel. Each chapter corresponds to a separate ground truth community in this graph.

Internet Movie Database (IMDB) Graph.

The IMDB graph consists of actors in movies [5]. Each vertex is an actor and an

Figure 5.1. Plot of average F1 score versus the average runtime per edge of each algorithm. The dashed line indicates the 0.5 power set score.

edge is placed between two actors if they performed in the same movie. Each ground truth community consists of actors who were all in the same movie. This graph is very large, with 382,219 vertices (actors) and 127,823 communities (movies). We will use this graph to demonstrate the our algorithms scale to larger graphs while also maintaining high accuracy.

■ 5.0.5 Experimental Results

We compare the performance of FLFA and ILFA to other algorithms on these graphs. Tables 5.2 and 5.3 show the resulting F1 based score (equation (5.1)) and number of communities found by each algorithm. Table 5.4 shows the runtimes of each algorithm.

Algorithm	Prime number graph	Culture show 2010	Culture show 2011	Les Miserables	IMDB
Modularity optimization	0.732	0.730	0.63	0.47	0.48
BigClam	0.54	0.88	0.79	0.63	0.49
CESNA	0.51	0.57	0.67	0.42	0.46
ILFA	**1.00**	**1.00**	**1.00**	**0.65**	**0.81**
FLFA	**1.00**	**1.00**	**1.00**	**0.65**	**0.81**

Table 5.2. F1 based score of communities produced by different algorithms on our dataset and average score of each algorithm. The best scoring algorithms for each graph are highlighted in bold.

Algorithm	Prime number graph	Culture show 2010	Culture show 2011	Les Miserables	IMDB
Ground-truth	168	13	10	80	127,823
Modularity optimization	105	7	5	5	2198
BigClam	56	45	38	8	100
CESNA	2	2	2	2	2
ILFA	**168**	**13**	**10**	**34**	**61,059**
FLFA	**168**	**13**	**10**	30	60,876

Table 5.3. Number of communities produced by each algorithm. The most accurate algorithms in terms of community number for each graph are highlighted in bold.

Accuracy

Table 5.2 shows that FLFA and ILFA perform well in terms of accuracy on these graphs, consistently obtaining the highest scores. In the prime number graph, FLFA and ILFA detect all communities exactly, obtaining a score of 1, outperforming the next highest performing algorithm by 23%. Similarly, in the culture show graphs, FLFA and ILFA again outperform the other algorithms. On both culture show graphs, FLFA and ILFA both achieve a perfect score of 1. The next best algorithm achieves a score of 0.88 on culture show 2010 and a score of 0.79 on culture show 2011.

In the Les Miserables graph, the ILFA and FLFA have the best score of 0.65. While this is not the perfect score we had on the prime number and culture show graphs, it is greater than the power set score. Finally, on the IMDB graph, FLFA and ILFA once

Algorithm	Prime number graph	Culture show 2010	Culture show 2011	Les Miserables	IMDB
Modularity optimization	6.53	0.07	0.12	0.02	681
BigClam	464	2.04	2.61	1.31	3,107
CESNA	81.80	0.47	0.47	0.13	881
ILFA	1.46	0.47	0.47	0.33	558
FLFA	**0.004**	**0.0004**	**0.0004**	**0.0003**	**0.81**

Table 5.4. Runtime (in seconds) of different algorithms on our dataset. The fastest algorithm for each graph is highlighted in bold.

again detect communities extremely well. As can be seen in the table, FLFA and ILFA achieve a score of 0.81. The other algorithms are not able to even cross the power set score.

In addition to having the best scores, the ILFA and FLFA also are the most accurate in terms of number of communities found, as seen in Table 5.3. In some instances, such as the IMDB graph, they are the only algorithms that come within the same order of magnitude of the number of ground-truth communities.

Runtime

Not only are FLFA and ILFA the most accurate algorithms on these datasets, but they are also the fastest. As shown in Table 5.4, FLFA and ILFA consistently perform orders of magnitude faster than alternate methods. In particular, FLFA is able to run much faster than the other algorithms. In some instances it is three orders of magnitude faster than the next best algorithm. What is even more striking is the fact that it achieves this incredible speed with without sacrificing much in accuracy. This is made more clear in Figure 5.1 where we plot the average runtime (normalized by the number of edges in the graph) of each algorithm versus the average F1 score. The ILFA has high accuracy and runtime comparable to fast modularity optimization and CESNA. However, the FLFA is the only algorithm which simultaneously has a very fast runtime

and high accuracy.

Robustness

Very often we will have missing data in an observation graph. We would like to know how robust our community detection algorithms to this type of noisy observation. To check robustness, we perform the following experiment. We randomly remove different fractions of edges from the IMDB graph. This random deletion of edges is meant to model missing data. We then apply the FLFA and look at how the edge deletion fraction affects the accuracy in terms of score and number of communities found. The results are shown in Figure 5.2. As more edges are deleted, the F1 score decreases, but not substantially. With 25% of the edges removed, the score decreases by only 12.5%. This shows that the FLFA's performance is not significantly degraded by missing data.

We saw earlier that missing data would result in spurious communities being found. From Figure 5.2 we see that this is indeed the case. With full observation, 61,876 communities were found by the FLFA. At 25% edge deletion, this number grows by 50%. These spurious communities generally have strong overlap with the communities found with no missing data, so even though they are numerous, their impact on the score is not as strong.

Chapter 6

Conclusion

WE presented the iterative and fast leader-follower algorithms (ILFA and FLFA) for community detection. The algorithms require no input parameters and learn the number of communities naturally from the graph. We proved that the ILFA and FLFA can exactly recover the community structure for a fairly broad class of graphs which are natural models for social networks. Experiments on graphs with ground truth communities showed that the ILFA and FLFA perform better than many state of the art algorithms. The FLFA was found to be very fast and while maintaining high accuracy. This suggests that it can be used to perform accurate, real-time community detection on extremely large graphs.

Bibliography

[1] E. M. Airoldi, D. M. Blei, S. E. Fienberg, and E. P. Xing. Mixed membership stochastic blockmodels. *The Journal of Machine Learning Research*, 9:1981–2014, 2008.

[2] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.

[3] R. L. Breiger. The duality of persons and groups. *Social forces*, 53(2):181–190, 1974.

[4] S. L. Feld. The focused organization of social ties. *American journal of sociology*, pages 1015–1035, 1981.

[5] C. for Complex Network Research. Network databases. http://www3.nd.edu/ networks/resources.htm, Jan. 2014.

[6] L. Hagen and A. B. Kahng. New spectral methods for ratio cut partitioning and clustering. *Computer-aided design of integrated circuits and systems, ieee transactions on*, 11(9):1074–1085, 1992.

[7] R. Karp. Reducibility among combinatorial problems. *Complexity of Computer Communications*, pages 85–103, 1972.

[8] A. Lancichinetti and S. Fortunato. Limits of modularity maximization in community detection. *Physical Review E*, 84(6):066122, 2011.

[9] M. Meila and J. Shi. A random walks view of spectral segmentation. *8th International Workshop on Artificial Intelligence and Statistics (AISTATS)*, 2001.

[10] J. W. Moon and L. Moser. On cliques in graphs. *Israel Journal of Mathematics*, 3:23–28, 1965.

[11] M. E. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006.

[12] G. Palla, I. Derényi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, 2005.

[13] J. Shi and J. Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905, 2000.

[14] G. Simmel. *Conflict and the web of group affiliations*. SimonandSchuster. com, 2010.

[15] D. A. Spielmat and S.-H. Teng. Spectral partitioning works: Planar graphs and finite element meshes. In *Foundations of Computer Science, 1996. Proceedings., 37th Annual Symposium on*, pages 96–105. IEEE, 1996.

[16] Y. Weiss. Segmentation using eigenvectors: a unifying view. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 975–982. IEEE, 1999.

[17] J. Yang and J. Leskovec. Community-affiliation graph model for overlapping net-

work community detection. In *Data Mining (ICDM), 2012 IEEE 12th International Conference on*, pages 1170–1175. IEEE, 2012.

[18] J. Yang and J. Leskovec. Overlapping community detection at scale: A nonnegative matrix factorization approach. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 587–596. ACM, 2013.

[19] J. Yang, J. McAuley, and J. Leskovec. Community detection in networks with node attributes. *IEEE International Conference On Data Mining (ICDM)*, 2013.