Game theory based algorithms for community detection

Radhika Arava

2015

Radhika Arava. (2015). Game theory based algorithms for community detection. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/65384>

<https://doi.org/10.32657/10356/65384>

Game Theory based Algorithms for Community Detection

RADHIKA ARAVA

School of Physical and Mathematical Sciences

Nanyang Technological University

A thesis submitted to the Nanyang Technological University

in partial fulfillment of the requirement for the degree of

Doctor of Philosophy

Abstract

The problem of community detection is important as it helps in understanding the spread of information in a social network. All real complex networks have an inbuilt structure which captures and characterizes the network dynamics between its nodes. Linkages are more likely to form between similar nodes, leading to the formation of some community structure which characterizes the network dynamic. The more friends they have in common, the more the influence that each person can exercise on the other. People use their attributes to assess the similarity of the other people with them, similarly, the attributes are also influenced with the people they interact and the network structure. Hence, we assume that communities capture homophily as people of the same community share a lot of similar features. The contributions of my thesis are as follows:

- We propose a disjoint community detection algorithm, *NashDisjoint* that detects disjoint communities in any given network.
- We evaluate the algorithm *NashDisjoint* on the standard LFR benchmarks, and we find that our algorithm works at least as good as that of the state of the art algorithms for the mixing factors less than 0.55 in all the cases.
- On Real Social Networks, we observe that the modularity values for the community structure detected by our algorithm *NashDisjoint* is in comparison to that of one of the best modularity optimization algorithms so far.

- We propose an overlapping community detection algorithm *NashOverlap* to detect the overlapping communities in any given network.
- We evaluate the algorithm *NashOverlap* on the standard LFR benchmarks and we find that our algorithm works far better than the state of the art algorithms in around 152 different scenarios, generated by varying the number of nodes, mixing factor and overlapping membership.
- The algorithm *NashDisjoint* and *NashOverlap* are modeled as a sequence of weighted potential games.
- We run our algorithm *NashOverlap* on DBLP dataset to detect the top collaboration groups. We have identified a giant component as a top collaboration group and the second largest collaboration group is much smaller than the giant component. This leads to a reasoning which says that the scientific collaboration network is highly connected and there is a possibility of more interdisciplinary work in the future which is a good sign for the development of science. The diameter of the largest collaboration group for both the datasets is between 9 and 11 and has the average path length between 4 and 5, which strengthens the above argument. Also, the average number of intermediate researchers that connect any pair of researchers in the giant component is quite less.
- These results of our algorithm on DBLP collaboration network are compared with the results of the *COPRA* algorithm.

Acknowledgements

Firstly, I would like to thank my supervisor, Prof. Ning Chen, for being very patient with me throughout my four years at NTU. The constant guidance in all possible ways, valuable discussions and directions from him are deeply regarded. He was always supportive and has given me ample time to think about new ideas and work at my own pace. I greatly appreciate the help and support I received from my colleague Dr. Xiaohui Bei through his discussions, without which I could not have imagined my PhD work completion.

I am happy to have Yair, Keshav, Dmitry and Anisur as my colleagues, with whom I spent a lot of time in the lab. Their technical discussions related to other problems always motivated me. I have always learned something new after having a discussion with them.

I want to thank my friends, Mohan, Alex, Anjali and Geeta for their company in NTU. I learned many things outside of my research, which wouldn't have been possible without their discussions. I had great fun and relieved my stress while hanging out with them.

I would like to thank my brother and my mother for being extremely supportive and gave me a lot of emotional strength during my stress times, to whom I dedicate my work.

Contents

List of Figures	v
1 Introduction	1
1.1 Motivation	5
1.2 Research Contributions	8
1.3 Thesis outline	11
2 Literatures on Disjoint Community Detection	13
2.1 Overview	13
2.2 Recursive bisectioning methods	13
2.3 Methods based on Centrality	15
2.4 Data point clustering Methods	18
2.5 Methods based on Spectral properties	19
2.5.1 Minimizing cutsize	19
2.5.2 Maximizing modularity	21
2.5.3 Other Spectral methods	24
2.6 Methods based on Modularity	26
2.6.1 Modularity Optimization	26
2.6.1.1 Greedy Techniques	27

2.6.2	Other Modular optimizations	29
2.7	Limits of Modularity	30
2.8	Community Detection using statistical inference	31
2.9	Community detection using Random Walk	31
2.10	Methods based on Game Theory	34
2.11	Community Detection using Information theory	34
2.12	Summary	35
3	Literatures on Overlapping Community Detection	36
3.1	Overview	36
3.2	Percolation Methods	36
3.3	Community Detection using Information Theory	38
3.4	Centrality based methods	39
3.5	Methods based on Optimization	40
3.6	Label propagation methods	42
3.7	Game theory based methods	43
3.8	Statistical inference based methods	43
3.9	Other Methods	43
3.10	Summary	44
4	Preliminaries	45
4.1	Theory	45
4.1.1	Model	47
4.1.1.1	Disjoint Community Detection	51
4.1.2	Overlapping Community Detection	53

5	Disjoint Community Detection	56
5.1	Overview	56
5.2	Benchmarks	57
5.2.1	Girvan Newmann Benchmarks	57
5.2.2	LFR Benchmarks	58
5.3	Evaluation Measures	60
5.3.1	Normalized Mutual Information	60
5.4	Algorithm <i>NashDisjoint</i>	61
5.4.1	Time Complexity	62
5.5	Results	62
5.5.1	Tests on Girvan and Newman benchmark	63
5.5.2	Tests on LFR Benchmark	63
5.5.3	Performance Analysis	67
5.5.4	Tests on Real Social Networks	71
5.6	Summary	72
6	Overlapping Community Detection	73
6.1	Overview	73
6.2	Benchmarks	74
6.3	Measures	75
6.3.1	Normalized Mutual Information	75
6.4	Algorithm <i>NashOverlap</i>	77
6.4.1	Time Complexity	78
6.5	Performance Results	79
6.5.1	Experimental Setup	79

6.5.2	Tests on LFR overlapping benchmark Networks	80
6.5.3	Tests on LFR Disjoint Benchmark Networks	98
6.5.4	Performance Analysis	100
6.6	Summary	102
7	DBLP Collaboration Network: A Case Study	103
7.1	Overview	103
7.2	Determining Collaboration Groups	107
7.3	Summary	126
8	Conclusion	128

List of Figures

5.1	Community Structure detected by <i>NashDisjoint</i> on Girvan Newmann Benchmark at $\mu = 0.1$	63
5.2	Community Structure detected by <i>NashDisjoint</i> on Girvan Newmann Benchmark at $\mu = 0.3$	64
5.3	Community Structure detected by <i>NashDisjoint</i> on Girvan Newmann Benchmark at $\mu = 0.5$	64
5.4	Performance of <i>NashDisjoint</i> on 1000 node network and large communities	67
5.5	Performance of <i>NashDisjoint</i> on 1000 node network and small communities	68
5.6	Performance of <i>NashDisjoint</i> on 5000 node network and large communities	68
5.7	Performance of <i>NashDisjoint</i> on 5000 node network and small communities	69
6.1	These figures show the comparison of community structures resulting a) at the end of first game of the algorithm <i>NashOverlap</i> , b) <i>NashOverlap</i> at its best and c) <i>NashOverlap</i> at $\alpha = 0.50$ for network size 5000 and community size in the range [20-100].	83

6.2	These figures show the comparison of community structures resulting a) at the end of first game of the algorithm <i>NashOverlap</i> , b) <i>NashOverlap</i> at its best and c) <i>NashOverlap</i> at $\alpha = 0.50$ for network size 5000 and community size in the range [20-50].	84
6.3	These figures show the comparison of community structures resulting a) at the end of first game of the algorithm <i>NashOverlap</i> , b) <i>NashOverlap</i> at its best and c) <i>NashOverlap</i> at $\alpha = 0.50$ for network size 1000 and community size in the range [20-100].	85
6.4	These figures show the comparison of community structures resulting a) at the end of first game of the algorithm <i>NashOverlap</i> , b) <i>NashOverlap</i> at its best and c) <i>NashOverlap</i> at $\alpha = 0.50$ for network size 1000 and community size in the range [20-50].	86
6.5	Performance of <i>NashOverlap</i> for 1000 node networks, small communities and $\mu = 0.1$	87
6.6	Performance of <i>NashOverlap</i> for 1000 node networks, small communities and $\mu = 0.2$	87
6.7	Performance of <i>NashOverlap</i> for 1000 node networks, small communities and $\mu = 0.3$	88
6.8	Performance of <i>NashOverlap</i> for 1000 node networks, small communities and $\mu = 0.4$	89
6.9	Performance of <i>NashOverlap</i> for 1000 node networks, small communities and $\mu = 0.5$	89
6.10	Performance of <i>NashOverlap</i> for 5000 node networks, small communities and $\mu = 0.1$	90

6.11	Performance of <i>NashOverlap</i> for 5000 node networks, small communities and $\mu = 0.2$	90
6.12	Performance of <i>NashOverlap</i> for 5000 node networks, small communities and $\mu = 0.3$	91
6.13	Performance of <i>NashOverlap</i> for 5000 node networks, small communities and $\mu = 0.4$	91
6.14	Performance of <i>NashOverlap</i> for 5000 node networks, small communities and $\mu = 0.5$	92
6.15	Performance of <i>NashOverlap</i> for 1000 node networks, large communities and $\mu = 0.1$	92
6.16	Performance of <i>NashOverlap</i> for 1000 node networks, large communities and $\mu = 0.2$	93
6.17	Performance of <i>NashOverlap</i> for 1000 node networks, large communities and $\mu = 0.3$	93
6.18	Performance of <i>NashOverlap</i> for 1000 node networks, large communities and $\mu = 0.4$	94
6.19	Performance of <i>NashOverlap</i> for 1000 node networks, large communities and $\mu = 0.5$	94
6.20	Performance of <i>NashOverlap</i> for 5000 node networks, large communities and $\mu = 0.1$	95
6.21	Performance of <i>NashOverlap</i> for 5000 node networks, large communities and $\mu = 0.2$	95
6.22	Performance of <i>NashOverlap</i> for 5000 node networks, large communities and $\mu = 0.3$	96

6.23	Performance of <i>NashOverlap</i> for 5000 node networks, large communities and $\mu = 0.4$	96
6.24	Performance of <i>NashOverlap</i> for 5000 node networks, large communities and $\mu = 0.5$	97
6.25	Performance of <i>NashOverlap</i> vs Other Algorithms on LFR Benchmark Network of 1K nodes and large sized communities, with disjoint community structure	98
6.26	Performance of <i>NashOverlap</i> vs Other Algorithms on LFR Benchmark Network of 1K nodes and small sized communities, with disjoint community structure	99
6.27	Performance of <i>NashOverlap</i> vs Other Algorithms on LFR Benchmark Network of 5K nodes and large sized communities, with disjoint community structure	99
6.28	Performance of <i>NashOverlap</i> vs Other Algorithms on LFR Benchmark Network of 5K nodes and small sized communities, with disjoint community structure	100
7.1	Depiction of Community Structure of DBLP collaboration network (large dataset) as detected by <i>NashOverlap</i> with vertices as communities and weight of edge between vertices show the sum of the weights of the cutedges between the communities represented by the vertices.	109

7.2	WordCloud for the conferences of giant component detected by <i>NashOverlap</i> in DBLP collaboration network (large dataset). Observe that almost 90% of the conferences show up and even some of the conferences based on industry oriented fields also show up in the same font and size as that of the theory based conferences	110
7.3	Depiction of Community Structure of DBLP collaboration network (small dataset) as detected by <i>NashOverlap</i> with vertices as communities and weight of edge between vertices show the sum of the weights of the cutedges between the communities represented by the vertices (only major communities are shown)	111
7.4	WordCloud for the conferences of giant component detected by <i>NashOverlap</i> in DBLP collaboration network (small dataset). Observe that only theory based conferences appear much bolder and bigger, whereas the conferences related to industry related fields are comparatively smaller in size or do not exist	112
7.5	Collaboration group on Databases, ML, AI, Pattern Recognition, Big Data. This is the second largest collaboration group detected on the small dataset	113
7.6	WordCloud for the conferences of second largest component detected by <i>NashOverlap</i> in DBLP collaboration network (small dataset). Observe that all the conferences are based in Machine Learning, Databases and Artificial Intelligence	114

7.7	Two major collaboration groups of sizes 408 and 239 on Database theory and systems are detected on the small dataset. Overlap size is 92 and the number of overlap edges is 187. Only high degree nodes are shown for visualization convenience. Each of the groups have an overlap of 61 and 39 with the giant component respectively. Overlapping nodes are colored red	115
7.8	Two major collaboration groups of sizes 222, 219 on Multimedia, Computer Vision are detected on the small dataset. Both are based in Asia, and have very weaker links with the giant component. Most of the high degree researchers are based in Microsoft research Asia. Overlap size is 117 and overlap edges are 291. Overlapping nodes are colored red. Each of the groups has an overlap of 6 and 7 with the giant component respectively.	116
7.9	Asian collaboration group in Data mining, Machine Learning, Knowledge Representation of size 152 detected for the small dataset. It has overlap of 16 with the giant component. Only high degree nodes are shown for visualization convenience	117
7.10	Collaboration group in Computational Geometry of size 133 detected for the small dataset. It has an overlap of 39 with the giant component. Only high degree nodes are shown for visualization convenience	118
7.11	Cumulative distribution of Overlapping membership of community structure detected by <i>NashOverlap</i> for small dataset at $\alpha = 0.5$. Overlap membership of 2,3,4 is found to more common.	120

7.12	Cumulative distribution of Community Size of community structure detected by <i>NashOverlap</i> for small dataset at $\alpha = 0.5$. Observe that most of the significant communities are between sizes 50 and 100. .	120
7.13	Community Structure of the DBLP Collaboration network (small dataset) detected by <i>NashOverlap</i> algorithm at parameter $\alpha = 0.5$ and $\beta = 1$ using <i>Fruchterman-Reingold</i> layout algorithm in gephi. Each community is given a separate color and the overlapping nodes are red in color. The community structure given by the <i>Fruchterman-Reingold</i> layout and the algorithm <i>NashOverlap</i> are in tune with each other. However, the communities are bigger in size and most of the possible communities are identified.	122
7.14	Community Structure of the DBLP Collaboration network (small dataset) detected by <i>COPRA</i> algorithm at parameter $v = 2$ using <i>Fruchterman-Reingold</i> layout algorithm in gephi. Each community is given a separate color and the overlapping nodes are red in color. The community structure given by the <i>Fruchterman-Reingold</i> layout and the algorithm <i>COPRA</i> are in tune with each other. However, the communities are not bigger in size and all the possible communities are not identified. Also, it did not detect much of the overlap between the pair of collaboration groups.	123

7.15 Depiction of Community Structure of DBLP collaboration network

(large dataset) as detected by *COPRA* with vertices as communities and weight of edge between vertices show the sum of the weights of the cutedges between the communities represented by the vertices.

Only communities of size more than 60 are observed 125

Chapter 1

Introduction

The insight of the topology of the network in terms of its community structure is important as it helps in understanding the spread of information, rumor, fashion, joke or any disease or any epidemic in a social network. For example, social networks like Facebook can deploy viral marketing strategies and place online ads on user profiles in such a way that fetches them the maximum revenue. Future popularity of a meme can be predicted by quantifying its early spreading pattern in terms of its community concentration. For example, the more communities a meme permeates, the more viral it is. Thus, information about community structure gives predictive knowledge about what information will spread widely [81]. Also, study of community structure in a power grid helps in estimating the robustness and stability of power transmission as in [43, 14].

Community structure of metabolic or biological networks can help in identifying the cure for complex diseases as in [39]. Using the data of the spread of epidemic over time in a certain region and the community structure of the network, one can identify the reasons behind the transmission of the disease and its

eventual spread in the entire region and find out the ways that it can be controlled. In large protein-protein interaction data sets, protein-protein interactions enable proteins to act concertedly to carry out their functions. Community structure on protein interaction networks show the potentially good candidates for the modules of proteins responsible for some function.

In real-world networks, each node is likely to be a part of more than one community. For example, in a social network like Facebook, MySpace, there can be overlap between a user's high school friends' circle and his university friends' circle. Similarly, if the user made friends with people of common interests, and we associate each community to an interest, then it is very likely that the same person can be in more than one community. So, it is common to expect the overlap between the communities. In a citation network on research papers, a paper which proposes innovative methods with applications in varied fields is likely to be cited by research papers from different fields. Similarly, in a collaboration network, a researcher may have simultaneously worked in two different collaboration groups due to change in his geographical locations or research fields or having worked for a long timespan. In a biological network on genes, a gene can be responsible for more than one disease. So, it is common to observe overlap in the community structures in many networks.

However, there also exist real world networks such that we are interested in segregating the nodes into disjoint communities, even when there is a possibility of overlap between the communities. For example, in task scheduling on multiprocessors, and finding clusters in vlsi design circuits, overlapping is not allowed since a we dont want to run the same task on two processors simultaneously or similarly

we do not allow the same transistor in two different clusters of the circuit.

Depending upon the problem in hand, we need an algorithm which detects the best community structure for the given network. We start by defining the notation needed for overlapping community structure in networks. Each node i of a network can be characterized by an overlapping membership number om_i , which is the number of communities that the node belongs to. In turn, any two communities C_1 and C_2 can share $C_1 \cap C_2$ nodes, which we define as the overlap size between these communities. Naturally, any community structure constitutes a network with vertices as communities and edges exist between a pair of communities if there are any cutedges between the communities. The size of the community is its number of nodes. The weight of the edge between the vertices is the sum of the weights of the cutedges between the communities denoted by the vertices. A good community structure should have be very sparse or should be very lightweight in the weights of its edges.

For a given community assignment to all the vertices of a network, mixing factor μ is the maximum fraction of degree of each vertex, outside its community. If the mixing factor is very low, then the community structure is very clear, with very minimal inter-community edges. If the mixing factor is very high, the community structure cannot be identified easily, as there will be several inter-community edges. Given a mixing factor, the network's community structure can be either disjoint or overlapping. In the disjoint community structure, each node is a part of only one community. If the network has overlapping community structure, then there exists some nodes which are a part of more than one community. Newmann et al [62] studied several scientific collaboration networks of MEDLINE, NCSTRL

and SPIRES manually and discovered that in each of those datasets, a giant component exists which constitutes a major part of the network, with the second in size collaboration group in much lesser size and hence noted that the scientific community need not be frightened of any fragmentation in the near future, as it is very well connected in all the fields.

In the literature, a large number of community detection algorithms have been proposed, mostly in the area of disjoint community detection. Methods used for community detection, both in the cases of disjoint community detection and overlapping community detection are divisive methods based on betweenness and similar measures [32, 61], methods based on searching for small cliques [66], information-theoretic techniques [73, 74, 26], random walks [22], modularity optimization methods [15], [63], [10], statistical inference through belief propagation [33], label propagation [36, 85, 84], game theoretic techniques [13, 58], [8, 55]. Of them, a very few algorithms are focussed at the problem of detecting overlapping community structures in a network. All these algorithms are extensively described in the literature survey sections.

Having seen the significance of community detection in both overlapping and non-overlapping cases, there is a need for a single model which explains the reasons for why and how the communities form in the first place in a social network, which lead to the algorithms that can detect the communities in the network accurately. However, there is no standard definition as to what set of nodes describes a community accurately. So, there is a standard benchmark which generates the network with a community structure, given the mixing factor, and the distribution of degree, community size and the overlapping parameters which include overlap-

ping membership and number of overlapping nodes. It is a common practice in the scientific community, that all the algorithms are tested for their quality against this benchmark.

1.1 Motivation

An algorithm for community detection takes the network as input and outputs a community structure which is a set of groups of nodes, which we call communities/clusters. Each community is usually assumed to be a densely connected group of nodes with more links among themselves than between the group nodes and the rest of the network. A node can be a part of more than one community. Two community structures are evaluated for their similarity using the standard measure, *Normalized Mutual Information* [18].

The idea behind our model is as follows: all real complex networks have an inbuilt structure which captures and characterizes the network dynamics between its nodes. In a given geographical region, each person is more likely to interact with the other person, which can create a friendship link between them. People have a tendency to associate with people who are more similar to themselves in several socially significant ways. Any two people with several common interests, skills and cultural backgrounds are more likely to form a link. Also, people usually tend to become friends with the friends of their friends. Thus linkages are more likely to form between such similar nodes, leading to the formation of some community structure which characterizes the network dynamic. Also, the more friends they have in common, the more the influence that each person can exercise on the other. People use their attributes to assess the similarity of the other people with them,

similarly, the attributes are also influenced with the people they interact and the network structure. We assume that communities capture homophily as people of the same community share a lot of similar features.

Most of the existing community detection algorithms fall into the following categories: a) optimizing the network structural properties like modularity, betweenness, conductance b) inference based approach c) label propagation d) information theory e) game theory based algorithms. The first approach does not let us model the reason behind the formation of the communities probabilistically as it focusses on optimizing a graph structural property. The algorithms due to second approach so far are not very robust and scalable and that of third approach runs quadratic in time or make too simple assumptions. The information theory approach worked well for the disjoint community detection, but it is not efficient in case of the overlapping community detection. There are a very few game theory based algorithms and the best game theory based overlapping community detection algorithm so far, has running time which is quadratic in the number of edges and is not very efficient. We need an algorithm that outputs the network's inbuilt community structure, even when the network has a fuzzy community structure.

Also, the algorithm should be simple to implement, scalable and should be natural in its construction, such that the model can reason the formation of the detected communities probabilistically. Also, in the literature, there are no simple and practically efficient game theory based algorithms which work for both disjoint community detection and overlapping community detection.

In a social network, it is natural that the the friendship between a pair of nodes becomes more stronger with the increase in the number of their common friends.

Each of them is more influenced by the other, and since they are under the influence of each other, they follow the same trend, or have similar behavior. Most certainly, each node behaves the same way as his closer friends who influence him the most does. We assume that given the topology of the network, and when enough number of the trends say r are allowed to be adopted by the network, then the network adopts at most k number of trends, where k is the number of communities that the network can naturally group into. This assumption makes sense because a pair of nodes is likely to make new connections to some other nodes, provided there exists some similarity in the pair and also between pair and the new connections that it makes. Also, the similarity increases with the number of connections that the pair makes. Hence in a closely connected groups of nodes, the clustering coefficient is higher and hence the number of triangles is higher, which means that to assume that a pair of nodes can influence each other, there should be several common neighbors. This is the idea behind our model and algorithms for community detection which follow subsequently.

Our model is natural since its underlying principle is the formation of the network using network dynamics. It reasons the formation of the communities probabilistically in the following way. In a group of people, the more the attributes that each person characterizes, the more difficult it is to decide upon a good community structure that can be characterized by the entire group. This is because, the strength of the link is proportional to the number of attributes that the two persons have in common. Also, the algorithms are very simple to implement and practically efficient.

1.2 Research Contributions

We have examined the problem of community detection in networks using game theory. The insight of the topology of the network in terms of its community structure is important as it helps in understanding the spread of information, rumor, fashion, joke or any disease or any epidemic in a social network. All real complex networks have an inbuilt structure which captures and characterizes the network dynamics between its nodes. In a given geographical region, each person is more likely to interact with the other person, which can create a friendship link between them. People have a tendency to associate with people who are more similar to themselves in several socially significant ways. Any two people with several common interests, skills and cultural backgrounds are more likely to form a link. Also, people usually tend to become friends with the friends of their friends. Thus linkages are more likely to form between such similar nodes, leading to the formation of some community structure which characterizes the network dynamic. Also, the more friends they have in common, the more the influence that each person can exercise on the other. People use their attributes to assess the similarity of the other people with them, similarly, the attributes are also influenced with the people they interact and the network structure. Hence, we assume that communities capture homophily as people of the same community share a lot of similar features. The contributions of my thesis are as follows:

- We propose a disjoint community detection algorithm, *NashDisjoint* that detects disjoint communities in any given network.
- We evaluate the algorithm *NashDisjoint* against the best state of the art

algorithms so far, and we find that our algorithm works at least as good as that of the state of the art algorithms on the standard LFR Benchmarks for the mixing factors less than 0.55 in all the cases.

- On Real Social Networks, we observe that the modularity values for the community structure detected by our algorithm *NashDisjoint* is in comparison to that of one of the best modularity optimization algorithms so far.
- We propose an overlapping community detection algorithm *NashOverlap* to detect the overlapping communities in any given network.
- We evaluate the algorithm *NashOverlap* against the best state of the art algorithms so far and we find that our algorithm works far better than the state of the art algorithms on the standard LFR benchmarks in around 152 different scenarios, generated by varying the number of nodes, mixing factor and overlapping membership.
- The algorithm *NashDisjoint* is modeled as a sequence of weighted potential games. As a part of the first game, the algorithm computes the closeness values for every edge. Worst case time for the first game to converge is $O(k \cdot k_1 \cdot m)$ where k_1 is the number of iterations it takes for each of the k subgames to converge. As a part of the second game, the algorithm picks the closeness values returned by the first game and computes the disjoint community structure. Its time complexity is $O(k_2 \cdot m)$ where k_2 is the number of iterations it takes for the second game to converge.
- Of all the community structures possible in the given network, our algorithm *NashDisjoint* detects the community structure which has a minimal value

for the sum of the probability strengths of the cut edges of that community structure.

- Similarly, the overlapping community detection algorithm *NashOverlap* can be formulated as a sequence of weighted potential games, where the value of the potential function for a given cover is constantly proportional to the sum of the probability strengths of the cut edges of that cover. For a given cover, duplicate the vertex for each community that it is a part of, and the cut edges are the edges between the vertices between two different communities. The time complexity is $O(k \cdot k_1 \cdot m)$ for the first game and $O(k_2 \cdot m)$ for the second game where k_1 and k_2 are respectively the number of iterations it takes for both first game and the second game to converge and k is the number of subgames.
- We run our algorithm *NashOverlap* on DBLP dataset to detect the top collaboration groups. We have identified a giant component as a top collaboration group and the second largest collaboration group is much smaller than the giant component. This is due to the percolating regime characteristic of such networks, there exist very small connected clusters, each with very few edges in them, and of of which most of them group to form a giant component with the increase in the number of edges in the network, leaving behind small clusters, which only group later into the giant component, with the further increase in connections. This leads to a reasoning which says that the scientific collaboration network is highly connected and there is a possibility of more interdisciplinary work in the future which is a good sign for the development of science. The diameter of the largest collaboration group for both

the datasets is between 9 and 11 and has the average path length between 4 and 5, which strengthens the above argument. Also, the average number of intermediate researchers that connect any pair of researchers in the giant component is quite less.

- These results of our algorithm on DBLP collaboration network are compared with the results of the *COPRA* algorithm. We find that our algorithm has detected more collaboration groups of significant size each of which carries an identity, compared to that of *COPRA*. Also, the communities detected by our algorithm are much bigger in size with a least number of outlier nodes. The community structure is likely to have more mixing factor than that of detected by *COPRA*. It means, that the collaboration groups are big enough in size with several researchers in each, so that almost no researcher who concern this collaboration group is ignored.

1.3 Thesis outline

The overview of the thesis is as follows. Chapter 2 gives the extensive literature review on the disjoint community detection algorithms till date. There are thousands of community detection algorithms which focus on detecting the disjoint communities in the networks. We have classified some relevant algorithms into groups based on their methods and explained the idea behind most of the algorithms.

Chapter 3 gives the literature review on the overlapping community detection algorithms till date. There are a very few community detection algorithms, which focus on detecting overlapping communities.

Chapter 4 explains the needed terminology for the thesis. In this chapter, the

model is explained. Also, the outline for algorithms *NashDisjoint* and *NashOverlap* are explained. This chapter reasons the choice behind the model and the parameters chosen in the algorithms and provides the proofs of convergence for both the algorithms.

Chapter 5 explains the important benchmarks and measures related to disjoint community detection that are in the literature. In this chapter, we explained our algorithm, the experimental setup and the performance comparison of our algorithm with that of the other best algorithms in the state of the art. Also, we picked some real social networks and observed the modularity comparison of the community structures detected by our algorithm and one of the best modularity optimization algorithms.

Chapter 6 explains the important benchmarks and measures related to overlapping community detection that are in the literature. In this section, we explained our algorithm, the experimental setup and the performance comparison of our algorithm with the other algorithms extensively for 152 cases.

Chapter 7 explores the dataset of computer science co-authorship network, DBLP which has researchers who published from 1966 to 2014, in 69 top ranked conferences. We detected various collaboration groups of the network using our algorithm, and explained all the important collaboration groups detected by our algorithm and some interesting properties of the detected collaboration groups.

Chapter 2

Literatures on Disjoint Community Detection

2.1 Overview

The literature on disjoint community detection algorithms is quite vast and can be categorised into several classes based on the methods used in the algorithms. Each of the following sections describe a method and some of the literature related to that method. Along with the literature, the intuition behind the methods, the advantages and disadvantages are also listed for each method.

2.2 Recursive bisectioning methods

These algorithms partition the network by minimizing the cut size of the partition. They require the input of number of communities and the sizes of the communities beforehand. These methods of community detection have applications in parallel computing, vlsi design, multi-processing. For example, in the case of parallel com-

puting, given some processors and a set of tasks, the problem is to assign tasks to each processor to minimize the communication between the processors. Similarly, in vlsi design, given a set of circuit boards, we need to assign the nodes to the circuits such that the number of connections between the boards is minimum. In the above problems, the number of connections between the circuit boards or the communication between the processors is to be minimized, with the restriction on the number of boards and the number of processors.

Kernighen-Lin method is one of the first methods of community detection, where the number of communities and the size of each community is known beforehand. This method maximizes the objective function f , which is the number of edges within the communities minus the number of edges between the communities. This method [44] tries to optimize the function f each time it splits a big community into a pair of communities. The algorithm begins with a random bipartition and equal subsets of nodes are swapped between the lobes of partition such that it maximizes the value of the objective function f . When, the function f cannot be maximized further, each lobe of the partition is subjected to a similar partitioning in the above fashion. However, finding out the pairs of subsets that are swapped to maximize the function, is a time consuming task. Assuming the constant number of swaps in each iteration, the time taken by this algorithm is $O(n^2 \log n)$ where n is the number of nodes in the network.

Ford and Fulkerson [29] proposed an algorithm to compute the minimum cut set between any two vertices s and t of the graph. This cut gives a bipartition of the graph, each of the lobes are further broken down into a bipartition, giving rise to a community structure. This method of sending the flow and computing

the minimum size edge set that topologically separates s from t , is used in many clustering algorithms [28], [68].

On the whole, the above algorithms that are based on graph partitioning are not very good for all the problems related to community detection, as they require the number of communities and size of the communities as input. In most of the cases other than parallel computing and circuit building, the problem requires us to understand the community structure without the prior knowledge of the inputs. Also, in these methods, we do not merge any two pair of components, once they are separated out, and this comes out as a severe backlog in such algorithms.

2.3 Methods based on Centrality

Girvan-Newmann Algorithm is a divisive algorithm introduced in the paper [32] and, improved further in [60]. It is the most popular and the first of its kind that introduced community detection to the physicists. Edge betweenness centrality of an edge is the number of shortest paths between all vertex pairs in the network that pass through that edge. The outline of the algorithm is as follows:

- 1 compute the edge betweenness centrality of all edges
- 2 remove the edge with highest edge betweenness centrality, in case of ties, break them arbitrarily
- 3 recalculate the edge betweenness centralities for the new graph
- 4 repeat from the step 2

The intuition behind the algorithm is that the edge that lies between two different communities has more edge betweenness centrality compared to the edges that are

not. Compared to the algorithms based on other notions of betweenness centralities (for example, split-betweenness centrality), this algorithm is much better on the quality of partitions and the time complexity [65, 61]. In the first paper [32], there is a hierarchy of partitions and there is no way to decide which level is the best one. However, in the paper [60], the authors gave a notion of modularity, which decides which level of partition is the best one.

The betweenness centrality of all edges of the graph can be calculated in a time that scales as $O(mn)$, or $O(n^2)$ on a sparse graph, with techniques based on breadth-first search. Numerical studies show that the recalculation step 3 of Girvan-Newman algorithm is important to detect meaningful communities. This introduces an additional factor m in the running time of the algorithm. Consequently, the edge betweenness version runs in time $O(m^2n)$. Nevertheless, the algorithm is quite slow, and applicable to sparse graphs with up to $n \sim 10000$ vertices, with current computational resources. The method can be simply extended to the case of weighted graphs, by generalizing the edge betweenness centrality function to the weighted edge in the following way. The betweenness of a weighted edge equals the betweenness of the edge in the corresponding unweighted graph, divided by the weight of the edge [15, 59].

Tyler et al. [79] [82] proposed a modification of the Girvan-Newman algorithm, to improve the speed of the calculation. The gain in speed was required by the analysis of graphs of gene co-occurrences [82], which are too large to be analyzed by the algorithm of Girvan and Newman. Tyler et al. proposed to calculate the contribution to edge betweenness only from a limited number of centers, chosen at random, deriving a sort of Monte Carlo estimate.

Another fast version of the Girvan-Newman algorithm has been proposed by Rattigan et al. [71]. Here, a quick approximation of the edge betweenness values is carried out by using a network structure index, which consists of a set of vertex annotations combined with a distance measure [70]. Basically one divides the graph into regions and computes the distances of every vertex from each region. In this way Rattigan et al. showed that it is possible to lower the complexity of the algorithm to $O(m)$, by keeping a fair accuracy in the estimate of the edge betweenness values. This version of the Girvan-Newman algorithm gives good results on the benchmark graphs proposed by Brandes et al. [31], as well as on a collaboration network of actors and on a citation network.

The algorithm of Girvan and Newman is unable to find overlapping communities, as each vertex is always assigned to a single cluster.

The edge clustering coefficient is proportional to the number of cycles that contains that edge in the graph. Also, the more cycles it is a part of, the probability of that edge being a part of the community is higher than the possibility of it being an intercommunity edge. So, there are some algorithms in the literature that use edge clustering coefficient to compute the communities. Radicchi et al. proposed this new measure, the edge clustering coefficient, such that low values of the measure are likely to correspond to intercommunity edges [68]. The running time of the algorithm to completion is $O(m^4/n^2)$ and it is much shorter than the running time of the Girvan-Newman method.

2.4 Data point clustering Methods

These methods find the community structure of the datapoints that are embedded in a metric space. Here the number of communities are known before hand. In this literature concerning the datapoints, communities are best known by the name of clusters. In each of the methods, for every pair of points, a measure of similarity or dissimilarity is computed. Now, the objective is to group the points into k clusters such that each cluster has almost similar points, and the points in different clusters are very dissimilar to each other. Points are grouped into clusters based on the objective cost function each computed as a function of the points in the cluster. The literature concerning this clustering encompasses different cost functions each corresponds to a different algorithm. For example, if the diameter of the cluster is the cost function, then it is Minimum k -clustering. If, the average of the distances between every pair of points in the cluster is the cost function, then it is called by the name of k -clustering sum algorithm. Similarly in k -center algorithm, for each cluster i , a reference point called as centroid is picked and the maximum distance of the centroid to each vertex of the cluster is considered as the cost function.

The most popular partitioning technique in the literature with regards to the data points in the metric space is k -means clustering [56]. In this method, the cost function is the total intra-cluster distance. Since, the number of clusters needed to be detected is k , we pick k centroids, i.e., k points are picked such that points in each pair of them are enough far from each other [54]. Each vertex joins the cluster of that point that is nearest to it. Once k clusters are formed, now, the centers of the clusters are determined which are the new centroids and the process is repeated, until the clusters do not change any more. This solution is not optimal, as

it depends on the initially chosen set of centroids. However, this heuristic converges rapidly and is scalable for large datasets.

Several extension to k-means clustering are proposed in the following papers [71] [41]. Since for the methods of partitioning clustering, the number of clusters should be given beforehand. Also, not all the real world graphs can be represented as datapoints on a metric space.

2.5 Methods based on Spectral properties

There exist several community detection algorithms based on spectral properties of the Laplacian matrices. For any set of n points, and a given dissimilarity/similarity function between a pair of points and if the similarity function is symmetric and non-negative, we can always construct a Laplacian for the matrix and its eigen value and vector properties can be used to partition the graph [19]. However, the Laplacian matrix used in the algorithm depends on the objective function that the algorithm uses to discover the community structure.

In the following subsection, we gave spectral methods which are in the literature to find the communities in the network structure using the spectral properties of the matrices.

2.5.1 Minimizing cutsize

In this method, the objective function is to maximize the difference between number of edges within the communities and number of edges between the communities. Construct a bipartition of the graph in the form of vector of size n , where n is the number of the vertices. The component value s_i corresponding to the vertex v_i is

+1 if it belongs to one group and -1 if it belongs to a different group. Then the cut size R is given by

$$R = \frac{1}{4} s^T L s \quad (2.5.1)$$

where L is the Laplacian of the network. In order to minimize the cut size, the value of R needs to be minimized. If we write s in terms of the eigen vectors, μ_i , where $i = 1, 2, \dots, n$ of the Laplacian and if s is properly normalized, then in terms of the eigen values, the above equation can be rewritten as

$$R = \sum_i a_i^2 \lambda_i \quad (2.5.2)$$

where λ_i is the laplacian eigenvalue corresponding to the eigen vector μ_i . For the laplacian matrix, one of its eigen values is zero and is the minimum eigen value, so in the sum term, there are at most $n - 1$ positive terms. To minimize the cut size, we need to minimize the right side of the equation. So, we pick the second lowest eigen value or the first positive eigen value, say λ , and assign a parallel vector to its corresponding vector μ to s . This vector μ has a special name called as Fiedler vector. But, direct assignment of μ to s cannot happen, since s takes only values $+1$ or -1 . So, we approximate s and compute the vector that is almost parallel to that eigen vector. If in that eigen vector, if $\mu_i > 0$, then $s_i = +1$ and if $\mu_i < 0$, then $s_i = -1$. This method may not output equal partitions, or partitions with predefined sizes. In that case, order the values of s in the order of their values. The vertices corresponding to the first n_1 values in the order where n_1 is the size of the first partition, are put in the first partition and the rest in the second partition.

However, this solution yields two different partitions, and we pick the one with the minimum cut size. This algorithm is given by Barnes et al [6]. Now, this partitioning is repeated over each partition and we get a set of communities as

output.

The spectral bisection method is quite fast. It takes $O(n^3)$ to compute the entire set of eigen vectors. However, if the eigen gap $\lambda_k - \lambda_{k-1}$ is larger, then the time to compute k eigen vectors is quite smaller, if computed by Lancos method.

2.5.2 Maximizing modularity

This method constructs the Laplacian which reflects the modularity function. Given the network and a division of the network into groups, the modularity is, roughly proportional to the difference between the number of edges falling within groups and the expected number in an equivalent network with edges placed at random. The equivalent network implies the network with the same degree sequence as that of the given network, but the edges are placed at random. So, in the equivalent random network, the expected number of edges between the vertices i and j is $\frac{k_i \cdot k_j}{2m}$. The modularity can be positive or negative. If the modularity values are very highly positive, it is representative of the existence of a good community structure.

The idea of using modularity to evaluate the quality of the division of the network is very well researched. Given the enumeration of some divisions of a given network, the division which has the highest modularity value is considered the best division. Guimera and Amara et al [39], Danon et al [18] ran experiments on the computer generated test networks and enumerated all its possible divisions and computed the modularity values for each of them. It is found that in most of the cases, the division with the highest modularity value is the correct division.

Suppose our network contains n vertices. For a particular division of the net-

work into two groups, let $s_i = 1$ if vertex i belongs to group 1 and $s_i = -1$ if it belongs to group 2. Suppose the number of edges between vertices i and j be A_{ij} (allowing for multiple edges, else, $A_{ij} = 0$ or 1 depending on the presence of an edge between i and j). Given a degree sequence for the network, if we place the edges at random, then the expected number of edges between i and j is $\frac{k_i \cdot k_j}{2m}$ where k_i and k_j are the degrees of the vertices i and j respectively and $m = \frac{1}{2} \sum_i k_i$ is the total number of edges in the network. Thus modularity Q is given by

$$\frac{1}{4m} \sum_{ij} (A_{ij} - \frac{k_i k_j}{2m}) (\frac{s_i s_j + 1}{2}) \quad (2.5.3)$$

i.e., it is the normalized sum of $A_{ij} - \frac{k_i k_j}{2m}$ over all pairs of vertices i and j that fall in the same group or cluster.

However, the quantity, $\frac{s_i s_j + 1}{2}$ is 1 if i and j are in the same group and 0 otherwise. So, we can express the modularity as follows :

$$\frac{1}{4m} \sum_{ij} (A_{ij} - \frac{k_i k_j}{2m}) s_i s_j \quad (2.5.4)$$

from the observation that $2m = \sum_i k_i = \sum_{ij} A_{ij}$. This can be expressed conveniently in the following way

$$Q = \frac{1}{4m} s^T \mathbf{B} s \quad (2.5.5)$$

where s is the column vector whose elements are the s_i and we have defined a real symmetric matrix with elements

$$B_{ij} = A_{ij} - \frac{k_i k_j}{2m} \quad (2.5.6)$$

This matrix is called modularity matrix. Note that the elements of each of its rows and columns sum to zero, so that it always has an eigenvector $(1, 1, 1, \dots)$ with eigenvalue zero. This matrix is similar to that of the Laplacian matrix which is used

in the spectral partitioning, one of the best known methods of graph partitioning in the previous subsection.

Let us assume that the eigenvalues are labeled in decreasing order, $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n$. We wish to maximize the modularity by choosing an appropriate division of the network, or equivalently by choosing the value of the index vector s . This means choosing s so as to concentrate as much weight as possible in the terms of the sum involving the largest (most positive) eigenvalues. If there were no other constraints on our choice of s (apart from normalization), this would be an easy task: we would simply chose s proportional to the eigenvector v_1 . This places all of the weight in the term involving the largest eigenvalue λ_1 , the other terms being automatically zero, since the eigenvectors are orthogonal.

Unfortunately, there is another constraint on the problem imposed by the restriction of the elements of s to the values $+1$ or -1 , which means s cannot normally be chosen proportional to v_1 . However, s is chosen in such a way that it contains only the values $+1$ or -1 and is parallel to vector v . But this gives a division of network into only two parts.

In order to divide the network into more than two parts, one way is to repeat the division of each part into two parts, and then divide those two parts further and so forth. However, after making a bipartition, we cannot simply delete the edges falling between the two parts and then apply the algorithm directly on each subgraph. This is because the degrees appearing in the modularity definition will change when edges are deleted, and any subsequent maximization of modularity maximizes the wrong quantity for the subgraphs.

Instead, the correct approach is to define for each subgraph g a new $n_g \times n_g$

modularity matrix $B(g)$, where n_g is the number of vertices in the subgraph, the change in the modularity function, which takes care of the entire graph as follows:

$$\Delta Q = \frac{1}{4m} s^T \mathbf{B}(\mathbf{g}) s, \quad (2.5.7)$$

Thus, computation of the modularity after each bisection should be done based on the full adjacency matrix of the original graph. However, one drawback of this method is again the same as that of iterative bisectioning methods. This method is more suitable for bipartitioning rather than n-partitioning.

Sun et al. [77] have added a step after each bipartition of a cluster, in that single vertices can be moved from one cluster to another and even form the seeds of new clusters. However, this is different from the Kernighan and Lin procedure [44], because, the number of communities that are output here is not known beforehand.

The spectral optimization of modularity is quite fast. The leading eigenvector of the modularity matrix can be computed with the power method, by repeatedly multiplying B by an arbitrary vector (not orthogonal to v_1). The number of required iterations to reach convergence is $O(n)$. Each multiplication takes a time $O(m+n)$. So, time it takes to compute a graph bipartition is $O(n(m+n))$. If the number of bipartitions needed is d , then total time is $O(n(m+n)d)$. The modularity matrix and the corresponding spectral optimization can be trivially extended to weighted graphs.

2.5.3 Other Spectral methods

Alves [5] used eigen values and eigen vectors of the Laplacian matrix to compute the effective conductances for every pair of vertices in a graph, assuming the network as a electric network with edges of unit resistance. The algorithm by Alves is

rather slow, as one needs to compute the whole spectrum of the Laplacian, which requires a time $O(n^3)$. Capocci et al. [11] derived a similarity matrix, where the similarity between vertices i and j is the Pearson correlation coefficient between their corresponding eigenvector components, averaged over a small set of eigenvectors. They used eigenvector components of the right stochastic matrix R , which is derived from the adjacency matrix by dividing each row by the sum of its elements. The right stochastic matrix has similar properties as the Laplacian. If the graph has g connected components, the largest g eigenvalues are equal to 1, with eigenvectors characterized by having equal-valued components for vertices belonging to the same component.

Yang and Liu [86] adopted a recursive bisectioning procedure and used eigenvector centrality to partition the network. They defined a new centrality measure of the vertices, called as clustering centrality, which is same as Bonacichs eigenvector centrality. This eigen vector centrality is the eigenvector of the adjacency matrix corresponding to the largest eigenvalue. The clustering centrality of a vertex is the probability that a random walker starting at that vertex hits a given target. If the probability is higher, then both the vertices belong to the same cluster, else they do not. If the graph has well-separated communities, the values of the clustering centrality would be similar for vertices in the same cluster. So, a bipartition is computed and the k clusters are computed iteratively. If we have to compute k clusters and if t iterations are taken, then the time taken is $O(kt(n \log n + m))$.

Donetti and Munoz [20],[21] has devised a spectral method of clustering for data points in the euclidean space in the following way. They considered the method of hierarchical way of clustering where only pairs of clusters which have at least

one interconnecting edge in the original graph are merged. Among all partitions of the resulting dendrogram, the one with largest modularity is chosen. Donetti and Munoz used both the Euclidean distance and the angle distance, to define the similarity between the vertices. The angle distance between two points is the angle between the vectors going from the origin of the m -dimensional space to either point. Time complexity is $O(n^3)$.

2.6 Methods based on Modularity

The notion of modularity is important in the context of community detection as it serves as a measure to quantify the strength of the communities, in the given network. It is introduced by Girvan and Newmann in their paper [63].

Modularity notion is introduced as a stopping criterion in the Newman-Girvan algorithm. From then, it stays as an important notion that defines the strength of the communities and has become the basis for several community detection algorithms. In this section, we give several clustering algorithms that were in the literature which use modularity directly/indirectly.

2.6.1 Modularity Optimization

In order to find the partition of the graph, that gives the highest modularity, we need to consider all the possible partitions of the graph and compute its modularity, and this problem turns out to be an NP-Complete problem. Currently, many heuristics and approximations to this modularity optimization problem are in the literature, that can find the best partitions in a reasonable time.

2.6.1.1 Greedy Techniques

The first algorithm that optimizes modularity in order to detect communities is given by Girvan and Newmann through their paper in 2004 [60]. It follows a greedy approach. It is a hierarchical clustering method that uses the bottom-up approach to find a hierarchical community structure in the network. The algorithm begins with each vertex as a community and in each iteration, the algorithm computes a pair of communities that gives the maximum positive difference in the modularity of the final partition, when they are merged. Merge that pair of communities, thus reducing the number of communities by 1 and repeat the process until there is no more increase in the modularity. To compute the difference in the modularity values in each iteration, every pair of mergeable communities in the previous partition are considered. The number of pairs of mergable communities is $O(m)$ and to update the cluster matrix, it takes $O(n)$ and the number of possible iterations is $O(n)$. So, the time taken by this greedy version is $O((m + n)n)$.

Clauset et al [15], used various data structures to update the modularity differences in each iteration and update the cluster matrix in considerably less time. Time Complexity is $O(md \log n)$, where d is the depth of the dendrogram. The greedy optimization of Clauset et al [15], is one of those modularity optimization algorithms that gives the partition with the best possible modularity on even large graphs. However, this greedy optimization, favors the quick formation of large communities at the expense of small ones.

Danon et al [17], normalized modularity variations that allow for the cluster merging at each iteration in such a way that favors small clusters and prevents the explosion of a single large community. Wakita and Tsurimi [80] modified the

Clauset et al algorithm [15] by changing the objective function that needs to be maximized at each iteration such that the algorithm gets better modularity values for large social networks and stop the formation of large communities.

Schuetz and Cafilisch [76] improved Clauset et al method by allowing the merging of more than two communities at each iteration and thus prevented the formation of large communities. These methods have almost the same complexity of Clauset et al and still gave better modularity values when compared to that of Clauset et al. Du et al [24], Xiang et al [83] gave greedy based algorithms with better modularity values, however beginning with intermediate partitions obtained in a different way, rather than beginning with n vertices as n communities.

Blondel et al. [10] gave a greedy heuristic algorithm to detect communities for the general case of weighted graphs, which happens to work well over all the previous greedy versions. Initially, each vertex is in a separate community. Vertices are considered sequentially. A vertex u is put in another vertex v 's community, if v happens to be a neighbor of u and the modularity increase of the partition is higher when u is put in v 's community among all its neighbors. Once it finishes all the vertices, we get the first partition. Each community in the partition is considered as a super vertex and two super vertices are connected if there is at least an edge between the vertices of the corresponding communities. The weight of the edge connecting the supervertices is the sum of the weights of the edges between the communities represented by the supervertices. This is repeated, and thus we get a hierarchy of communities, until the modularity cannot increase anymore. The modularity of each partition is computed with respect to the original graph, and the new graph is not considered when modularity is computed. It takes around

$O(m)$ time. So, the algorithm is extremely fast and graphs with up to 10^9 edges can be analyzed in a reasonable time on current computational resources.

The modularity values obtained by this algorithm are better than that of Clauset et al and all the current modularity optimization algorithms based on greedy techniques. However, the results of this algorithm depend on the sequential sweep of the vertices. Despite so many improvements in the greedy techniques, the quality of partitions is not quite good compared to that of other algorithms.

2.6.2 Other Modular optimizations

In simulated annealing, the algorithm explores almost all the possible states to optimize an objective function f and get its global optimum. Guimera et al [40] used simulated annealing for the first time to detect communities by optimizing the modularity function. The algorithm [38] comprises of local moves where a vertex is allowed to move from one cluster to another and global moves. This method can give almost true modularity maximum or very close values to modularity maximum. However this method is pretty slow and can only run for graphs of size at most 10^4 vertices.

Extremal Optimization is based on the optimization of local variables, expressing the contribution of each unit of the system to the global function at study. This technique was used for modularity optimization by Duch and Arenas [25]. In this technique, the modularity is written as a sum over all the vertices and the value of the local modularity of a vertex is the value of the corresponding term in this sum. A fitness measure for each vertex is obtained by dividing the local modularity of the vertex by its degree. To begin with, a random bi-partition of equal sized lobes is

picked and in each iteration, the vertex with the lowest fitness is shifted to the other cluster. The move changes this partition, so the local fitnesses of many vertices need to be recalculated. The process continues until the global modularity cannot be improved any more by the procedure. The total complexity of the method is then $O(n^2 \log n)$. This method gives a good tradeoff between accuracy and speed, however the problems due to iterative bisectioning always exists on large networks with many communities.

In spectral optimization, first a laplacian matrix that represents the modularity function is constructed and the problem is reduced to finding the eigen vectors of the matrix such that the modularity function is maximized, which is extensively discussed in the previous section.

Chen et al. [12] have used integer linear programming to transform the initial graph into an optimal target graph consisting of disjoint cliques, which effectively yields a partition. Berry et al. [9] have formulated the problem of graph clustering as a facility location problem, consisting in the minimization of a cost function based on a local variation of modularity. Agarwal and Kempe formulated the modularity function in terms of a linear and a quadratic program and used the framework of mathematical programming [3].

2.7 Limits of Modularity

The modularity function assumes that every node in the network is likely to be connected to every other node in the network, as it compares the expected number of edges of the random null model to the actual edges of the network. However, as the size of the network increases, it is very unlikely that we assume every node to

be connected to every other node, but restrict a part of the network to a node. For larger networks, the expected number of edges between the two clusters in the null model is quite small, due to a very large denominator. So, the presence of an edge between any two clusters leads to a large increase in the modularity value and thus leads to merging of the two clusters. So, in modularity optimization algorithms, it becomes hard to detect small clusters in large networks. Fortunato et al detected this limit of modularity as resolution limit in his paper [30].

2.8 Community Detection using statistical inference

Methods of community detection based on Bayesian inference, considers the prior distribution on the three parameters: the number of clusters, the cluster assignment of the vertices and model parameters and collects some observations about the network. The algorithm decides on some prior distribution and computes the model that maximizes the posterior distribution of the model parameters given the model and the information collected about the system. This model of parameters and cluster assignment of the vertices reflect the detected community structure of the network. Sun et al [78], Newmann et al [64], Ren et al [72] have used methods based on statistical inference to detect the communities.

2.9 Community detection using Random Walk

Random walks can also be useful to find communities. A random walk in a network spends more time within the communities and happen to use more of the intracom-

munity edges rather than the intercommunity edges. So, if a graph has a strong community structure, a random walk spends a long time inside a community due to the high density of internal edges and the huge number of paths that could be followed when compared to the graph with a loose community structure.

In this section, we describe the most popular clustering algorithms based on random walks. All of them can be trivially extended to the case of weighted graphs. The algorithms based on random walks gave better results than that of modularity optimization algorithms.

The algorithm given by Zhou et al [89] computes the closeness distance d_{ij} between every pair of vertices which is the average number of edges that a random walker has to cross to reach j starting from i . The community detection is based on the intuition that any pair of vertices i and j with smaller values of d_{ij} will mostly likely belong to the same community. This algorithm requires the computation of the distance matrix which takes time $O(n^3)$. In a subsequent paper, Zhou et al [83] proposed a different computation of the distance matrix, allowing only selected vertices as a part of the given vertex's community. These algorithms are divisive in style, starting from the actual graph as a single community and separating the vertices into communities, thus minimizing the chance of having the dissimilar vertices in the same community. The time complexity of this algorithm is $O(n^3)$.

Van Dongen [23] et al proposed the best algorithm so far in random walks as a part of his thesis, famously called as Markov Cluster Algorithm. This method computes the transfer matrix T of the graph G . The element T_{ij} of the transfer matrix gives the probability that a random walker, sitting at vertex j , moves to i . The sum of the elements of each column of T is one. Each iteration of the algorithm

consists of two steps. In the first step, called expansion, the transfer matrix of the graph is raised to an integer power p (usually $p = 2$). The entry M_{ij} of the resulting matrix gives the probability that a random walker, starting from vertex j , reaches i in p steps (diffusion flow). The second step, which has no physical counterpart, consists in raising each single entry of the matrix M to some power α , where α is now real-valued. This operation, called inflation, enhances the weights between pairs of vertices with large values of the diffusion flow, which are likely to be in the same community. Next, the elements of each column must be divided by their sum, such that the sum of the elements of the column equals one and a new transfer matrix is recovered. After some iterations, the process delivers a stable matrix, with some remarkable properties. Its elements are either zero or one, so it is a sort of adjacency matrix. The graph described by the matrix is disconnected, and its connected components are the communities of the original graph. The method is really simple to implement, which is the main reason of its success: as of now, the MCL is one of the most used clustering algorithms in bioinformatics [22].

Due to the matrix multiplication of the expansion step, the algorithm should scale as $O(n^3)$. However, while computing the matrix multiplication, MCL keeps only a maximum number k of non-zero elements per column, where k is usually much smaller than n . So, the actual worst-case running time of the algorithm is $O(nk^2)$ on a sparse graph. A problem of the method is the fact that the final partition is sensitive to the parameter α used in the inflation step.

2.10 Methods based on Game Theory

Lung et al [55] used a game theoretical approach where each network node is a player and each player choose its community. Each player maximizes its payoff computed based on a community score. The Nash equilibrium of this game is a situation in which no node can improve its payoff by unilaterally changing its community. Xiaohui Bei et al in their paper [8] proposed that trial and error behaviour of individuals in a social network forms a reason behind the formation of its communities and gave a heuristic algorithm based on game theory that can detect the community structure of the network. Suri et al [58] proposed a decentralized game theoretical based community detection algorithm, in which they have defined a utility function for each node and in each iteration, each node maximizes its utility value. The utility function is defined in such a way that there always exists a nash equilibrium in such a game. However this algorithm is not scalable to larger graphs.

2.11 Community Detection using Information theory

Rosvall and Bergstrom in their paper [73], proposed a community detection algorithm by encoding the communities using lesser information than that encoded in the network's adjacency matrix, using random walks. The method works by naming the important structures of the graph with unique names and the vertex names may be recycled among different structures, thus leading to a more compact description than by simply coding all vertices with different names. Huffman coding is used to name vertices and the structures which are here communities. Thus, the

problem of graph clustering defined in the paper is to find the partition that yields the minimum description length of an infinite random walk. The minimization of the description length is carried out by combining greedy search with simulated annealing. Rosvall and Bergstrom in their next paper [74], used the greedy modular optimization technique of Blondel et al.[10] with some refinements. The method can be applied to weighted graphs, both undirected and directed.

2.12 Summary

An extensive overview on the literature behind the disjoint community detection algorithms is presented. As the literature is vast, segregation of the algorithms into classes based on the methods used in the algorithms is presented.

Chapter 3

Literatures on Overlapping Community Detection

3.1 Overview

In this chapter, an extensive literature survey on the algorithms related to the overlapping community detection is given. The literature on the overlapping community detection is segregated into classes based on the methods used in the algorithms. In each section, the method is explained along with its advantages and disadvantages.

3.2 Percolation Methods

Palla et al proposed the first percolation method for community detection in their Nature paper [66]. It assumes that the network has overlapping sets of completely connected subgraphs. Given the parameter k , the clique percolation method (we call it as *CPM* hereafter) is as follows :

1. identifies all the cliques of size k in a network.

2. constructs a new graph by replacing each k -clique by a new node and an edge is put between the two new nodes if the k -cliques denoted by the two new nodes share $k - 1$ vertices.
3. the new graph is given as input to the step 1 and steps 2 and 3 are repeated until no more edges can be put between the nodes.

The vertices of each connected component in the resultant graph represents a community, in the final output. In this method, the community can be visualized as a maximal union of k -cliques, in which every pair of k -cliques are connected by some k -cliques, which is adjacent to both of them by sharing $k - 1$ vertices, with each one of them. For directed networks, a directed k -clique is a complete subgraph (without the directions) with k nodes such that the given a rank to each of the nodes, there exists a directed link pointing from the higher rank node to the lower rank node.

In a real social network, there exists a giant component of size proportional to the network size. If we need to detect the giant component, then we need to decrease the value of the parameter k in this method. This is because, as the value of k increases, the size of the detected communities become smaller and hence the probability of detecting a giant component decreases. However, if we want to detect only the densely connected communities, we can set k to larger values. The community detection by means of *CPM* can be considered as local, because once a community with the above definition is formed, it is not disturbed in the subsequent iterations. So, this method does not suffer from the problem of resolution limit, which arises due to the optimization of a function globally.

Farkas et al. [27] modified the clique percolation algorithm for the weighted networks. The weight of k -clique is given by the geometric mean of the weights of all its edges. The value of the threshold is chosen slightly above the critical value at which a giant k -clique community emerges, in order to get the richest possible variety of clusters. Kumpula et al [45] modified *CPM* algorithm by enumerating all the $(k - 2)$ -size cliques among all the common neighbors of an edge in the order of their weights and processes all the cliques, for a community structure.

The clique percolation method has the same problem as that of Radicchi et al. [68]. If the network has too many cliques, then the algorithm may return the entire network as a single cluster. If the network has very few cliques, then it cannot give meaningful covers. In each real network, there are a considerable fraction of vertices which are left out of communities, which needs postprocessing to be included in the community structure. Also the criterion to choose the threshold for weighted graphs and the definition of directed k -cliques are rather arbitrary.

3.3 Community Detection using Information Theory

Rosvall et al [75] introduced hierarchical map equation into their algorithm and have used their algorithm to detect the multilevel organization of social and biological networks. In a subsequent paper, Rosvall et al [26] introduced a map equation for overlapping modules, which measures the compression of the flow to find the overlapping modules.

3.4 Centrality based methods

Pinney and Westhead [67] have proposed a modification of the Girvan and Newman algorithm in which vertices can be split between communities. They compute the vertex-betweenness of all the vertices of the graph, the edge betweenness of all the edges. They remarked that the two end vertices of an inter-cluster edge have similar betweenness values. An edge with the largest betweenness is removed only if the ratio of the betweenness values of its end vertices is between α and $\frac{1}{\alpha}$, with $\alpha = 0.8$. Otherwise, the vertex with highest betweenness (with all its adjacent edges) is temporarily removed. When an edge/vertex is removed from a component, dividing the component into subcomponents, all the deleted vertices are copied into each subcomponent along with the edges.

Gregory [35] proposed an algorithm, named *CONGA* (Cluster Overlap Newman-Girvan Algorithm), in which vertices are split among clusters if their betweenness centrality exceeds the maximum value of the betweenness of the edges. A vertex is split among the clusters by assigning some of its edges to one of its duplicates, and the rest to the other. One way to split the vertex among the clusters is to use the concept of split-betweenness proposed by Gregory. Split betweenness is the number of shortest paths that would run between two parts of a vertex if the latter were split. The algorithm CONGA proposes that the vertex needs to be split between the clusters in such a way that split-betweenness is maximum. The algorithm has the time complexity of $O(m^3)$, like the algorithm of Girvan and Newman.

3.5 Methods based on Optimization

These algorithms rely on a function which characterises the quality of a densely connected group of nodes. Baumes [7] proposed the algorithm where the nodes are ranked based on some criterion. Then the process iteratively removes highly ranked nodes until small, disjoint clusters are formed. These clusters serve as seed communities for the second step of the process that expands the clusters by adding or removing nodes until a local density function cannot be improved. The algorithm runs in time $O(n^2)$. Lancichinetti et al [48] proposed an algorithm which allows the growth of a community from a single node. The algorithm assumes that the communities are local structures and grows a natural community using the fitness function which is as defined as follows :

$$f_c = \left(\frac{k_{in}^c}{k_{in}^c + k_{out}^c} \right)^\alpha \quad (3.5.1)$$

where k_{in}^c and k_{out}^c are the total internal and external degree of the community c , and α is the resolution parameter that controls the size of the communities. Given a community c , the fitness of node v with respect to the community c is as follows:

$$f_c^v = f_c^{+v} - f_c^{-v} \quad (3.5.2)$$

Starting from a random node, the algorithm determines the community of the node v using the fitness function in the following way :

- each neighboring node of the community c which is not included in c is considered
- The fitness function for each neighboring node is calculated and the neighbor with the largest fitness is added to the community c

- the fitness of each vertex of c is recomputed
- if any node has a negative fitness, then it is removed from the community and the algorithm repeats from the previous step, otherwise, the algorithm repeats from the first step.

Once a community is found out, the algorithm picks another node which is not yet assigned to any community previously discovered and grows the community around it, in a similar fashion until there is no left out node. the time complexity and the performance of LFM depends a lot on the resolution parameter α . Higher α gives smaller communities and as it becomes smaller it gives rise to larger communities. It has been found that for $\alpha < 0.5$, the algorithm outputs only one community. The computational complexity for a fixed α -value is roughly $O(n_c s^2)$, where n_c is the number of communities and s is the average size of communities. The worst-case complexity is $O(n^2)$, when the sizes of communities are in comparable to the size of the network.

The method “Ordered statistics local optimization method” (*OSLOM*) [50] optimizes the statistical significance of the clusters, which it considers as a fitness measure, by defining them to a global null model, which is a random graph generated by the configuration model. The notion of statistical significance of a cluster, proposed first by Bianconi is roughly the probability of the existence of the cluster in a null model which is similar to that defined by Newmann.

To grow the current community for a given vertex, the cumulative probability of having the number of internal connections equal or larger than the number of connections from a neighbor into this community in the null model is computed for each neighbor. If the cumulative distribution of the smallest r value is smaller than

a given tolerance, it is considered to be significant, and the corresponding node is added to the community. Otherwise, the second smallest r is checked and so on. *OSLOM* usually results in a significant number of outliers or singleton communities. The worst-case complexity in general is $O(n^2)$, while the exact complexity depends on the community structure of the underlying network being studied.

3.6 Label propagation methods

Gregory in his paper [37] proposed the algorithm COPRA, in which in each iteration, each node updates its belonging coefficients to its communities by averaging the co-efficients from all its neighbors. The algorithm needs the parameter v to be tuned from 1 to 10 and it controls the maximum number of communities that a node can associate. The time complexity of the algorithm is $O(vm \log (vm/n))$ per iteration.

The label propagation algorithm [69] in which nodes with same label form a community, has been extended to overlapping community detection [85], by allowing a node to have multiple labels. SLPA [85] proposed by Xie et al, is a general speaker-listener based information propagation process. It spreads labels between nodes according to pairwise interaction rules. Unlike [69], [37], where a node forgets knowledge gained in the previous iterations, SLPA provides each node with a memory to store received labels. The time complexity is $O(tm)$, linear in the number of edges m , where t is a predefined maximum number of iterations (e.g., $t \geq 20$). SLPA can also be adapted for weighted and directed networks by generalizing the interaction rules, known as *SLPAw*.

3.7 Game theory based methods

Chen et al proposed a game-theoretic framework to detect overlapping communities in their paper [13], in which a community is associated with a Nash local equilibrium. A gain function and a loss function are associated with each agent. The game assumes that each agent is selfish and selects to join, leave and switch communities based on its own utility. An agent is allowed to join more than one community, if it results in its increased utility. The time it takes to reach a local equilibrium is bounded by $O(m^2)$, where m is the number of edges.

3.8 Statistical inference based methods

The paper by Prem et al [33] is based on statistical inference based methods and is quite fast on large networks and still work well on benchmarks over some other algorithms. At any point of time, the algorithm needs to load only a part of the network into the memory. The algorithm involves the following process: iteratively sample the graph and get an estimate of the communities and the community structure of the graph and improve on the estimate iteratively.

3.9 Other Methods

Leskovec et al [88], [87] proposed an overlapping community detection method, called BIGCLAM (Cluster Affiliation Model for Big Networks), which is scalable to large networks. They have assumed that the overlapping parts of the communities are also clustered and their algorithm could detect the densely overlapping parts of the communities and hierarchical nature of the community structure and

tested their algorithm on six social and collaboration networks whose groundtruth information is already known.

Leskovec et al [52], [51] characterized the community structure of real social networks. They introduced the term *network community profile plot*, which characterizes the best possible community according to the conductance measure, given a community structure of the network. For large networks, using the network community profile plot, they remark that the communities of size less than 100, have a very good conductance and communities beyond the size 100, blend into an expander-like core of the network and have bad conductance values. Leskovec et al [53] proposed a machine learning technique to identify the communities or groups in ego-networks, most commonly occurring in social networks. They developed a model to detect circles which are overlapping and have a nested circle structure, using the network structure and user profile information.

Ahn et al [4] proposed the overlapping community detection algorithm assuming that the overlap between the communities is very sparse. The algorithm focusses on grouping the links rather than the nodes and called them link communities. They introduced the term called *Partition density* which is similar to the modularity, but do not suffer from resolution limit, to distinguish the community structures.

3.10 Summary

From this chapter, it can be observed that in comparison to the literature on disjoint community detection, there is not much literature on overlapping community detection. Though the problem of overlapping community detection is well motivated, not much research has been done so far.

Chapter 4

Preliminaries

4.1 Theory

In a social network $G = (V, E)$, V is the set of vertices and E is the set of undirected edges that represent relations between vertices in the network. Let $w(i, j)$ be the weight of the edge $(i, j) \in E$. If the graph is unweighted, we assume that each edge has unit weight. A community in a social network is a non-empty connected subset of vertices with denser connections within themselves than with the rest of the network. A network has a community structure if the vertices in the network can be grouped into communities such that the connections are dense within the community and are sparse between the communities. The community structure can be non-overlapping or overlapping. Formally, let C_i be a subset of vertices denoting a community.

Definition 1. A partition of the network G is defined as $\Gamma = \{C_1, C_2, \dots, C_m\}$, where $\bigcup_i C_i = V$ and $C_i \cap C_j = \emptyset$ for any $i \neq j$.

Definition 2. A cover of the network G is defined as $\Gamma = \{C_1, C_2, \dots, C_m\}$, where

$\bigcup_i C_i = V$ and $C_i \cap C_j$ need not be $\emptyset$ for any $i \neq j$.

There is no standard notion as to how good the grouping of the subset of vertices should be, for it to be called as a community. A network can be naturally grouped into non-overlapping communities or overlapping communities. However, some scenarios require us to detect only disjoint community structure for its purposes, despite the existence of a few overlapping vertices. So, we proposed community detection algorithms one for each purpose: a) Disjoint Community Detection b) Overlapping Community Detection.

We pose each algorithm as a sequence of two weighted potential games. The first game is common for both the algorithms and at the end of the game, the closeness of vertices on each of its adjacent vertices is computed. The second game is slightly different for each of the algorithms, to accommodate its purpose. We first define the concept of weighted potential game and describe each algorithm as a sequence of weighted potential games.

A weighted potential game is first introduced by Monderer and Shapley in their landmark paper [57].

A game with a finite number of strategic players say n is a weighted potential game if it admits a weighted potential function, which is as defined follows.

Let $t = (t_i)_{i \in V}$ be a vector of positive real numbers called weights. Let Y_i be the set of strategies of vertex i and the utility function of player i is $u_i : Y \mapsto \mathbb{R}$ where $Y = Y_1 \times Y_2 \times \dots \times Y_n$. A function $\Phi : Y \mapsto \mathbb{R}$ is a weighted potential function if for every $i \in V$, and for every $y_{-i} \in Y_{-i}$, and for every $y_i, y'_i \in Y_i$,

$$\Phi(y'_i, y_{-i}) - \Phi(y_i, y_{-i}) = t_i(u_i(y'_i, y_{-i}) - u_i(y_i, y_{-i})) \quad (4.1.1)$$

Definition 3. A strategy profile is called a (pure) Nash Equilibrium if no individual

i can improve his utility by unilaterally changing his own strategy (i.e., adopting another strategy).

Every weighted potential game has a pure nash equilibrium [57].

First, the model and the assumptions underlying the algorithms are described. Each algorithm is described as a sequence of weighted potential games and show the existence of the pure nash equilibrium in all the cases by proving conditions of convergence wherever they are required.

4.1.1 Model

For a given social network G , consider the following game: Each vertex $i \in V$ represents a player. There are two types of strategies a and b , and each player i determines which strategy to adopt. To begin with, each vertex i picks a strategy $s_i \in \{a, b\}$ at random. Approximately, half the vertices pick strategy a and the other half pick the strategy b . In any given round, an arbitrary vertex i changes its strategy if it can increase its utility, greater than that of its current utility. At any moment, the cut is defined as the subset of vertices which adopt a single strategy while its complement adopts an another strategy and the cut edge is the edge between two vertices playing different strategies. This model is inspired from the paper by Bei et al. [8].

Let $s_i \in \{a, b\}$ denote the strategy that player i chooses; then $(s_i)_{i \in V}$ defines a strategy profile of the game. For any given strategy profile, the normalized *utility* of player i is defined as follows:

$$u_i = \frac{\sum_{\substack{j:(i,j) \in E \\ s_i = s_j}} t(i, j)}{\sum_{j:(i,j) \in E} t(i, j)} \quad (4.1.2)$$

where

$$t(i, j) = w(i, j) + \sum_{\substack{k: (i, k) \in E \\ (k, j) \in E}} \{w(i, k) + w(j, k)\}$$

Observe that $t(i, j)$ is the sum of the weights due to all common friends k of i and j plus the edge weight $w(i, j)$.¹ The numerator in the above formula is the total tie-strength of the neighbors of i that choose the same strategy as him, and the denominator is the total tie-strength of all edges incident to i , which is a fixed number. The utility of an individual thus completely depends on his and his friends' strategies.

Let the potential function Φ be the function from the cut to the sum of the tie-strengths of the edges which are not cut edges. Let t_i be the sum of the tie-strengths of the adjacent edges of player i . Let $u_i(s_i, s_{-i})$, be the utility of the player i when his strategy is s_i , while the strategy vector of the other players is s_{-i} . According to the definition, the utility function $u_i(s_i, s_{-i})$ is the fraction of the sum of the weights of the adjacent edges with condition $s_i = s_j$.

$$u_i(s_i, s_{-i}) = \frac{\sum_{\substack{j: (i, j) \in E \\ s_i = s_j}} t(i, j)}{t_i} \quad (4.1.3)$$

We now prove that in each round, when an arbitrary vertex k changes its strategy, along with the increase in the utility of the vertex k , the potential function also increases in its value and the difference in the change of the potential function is constant times proportional to the increase in the utility of the vertex k . The

---

¹The function $t(i, j)$ gives the tie strength between i and j . The concept of tie strength was first introduced by Granovetter in his landmark paper [34]; a basic hypothesis is that the stronger the tie strength between two individuals, the larger the proportion of individuals in the rest to whom they both are tied.

constant is t_i , which is given by the sum of the tie-strengths of adjacent edges of the vertex k .

Proof. When an arbitrary vertex k changes its strategy from s_k to s'_k to increase its utility, the set of cut edges changes from E_c to E'_c . The potential function increases in value and the increase in the value of the potential function Φ is as follows

$$\sum_{(i,j):(i,j) \in E \setminus E'_c} t(i,j) - \sum_{(i,j):(i,j) \in E \setminus E_c} t(i,j) \quad (4.1.4)$$

Since, in this round, only vertex k changes its strategy, the cut edges due to k become internal edges and internal edges before the change in the strategy become cut edges, and the rest of the cut edges do not change their state. So, the above expression is same as

$$\sum_{l:(k,l) \in E \setminus E'_c} t(k,l) - \sum_{l:(k,l) \in E \setminus E_c} t(k,l) \quad (4.1.5)$$

For each edge (k,l) in $E \setminus E'_c$, since s_k is same as s_l , the above expression is written as

$$t_k * [u_k(s'_k, s_{-k}) - u_k(s_k, s_{-k})] \quad (4.1.6)$$

However, since the vertex k changes its strategy only when his utility is greater than that of its current strategy, the expression 4.1.6 is always positive. Also, the difference in the potential function is same as the difference in the utility function of the player who changed its strategy. So, we formulated the dynamic as a weighted potential game and hence the dynamic always converges. $\square$

Thus, a pure Nash equilibrium always exists, and can be achieved by the *best response dynamics*, where players sequentially choose strategies that yield the highest utilities. This simulation or the sub-game is repeated 1000 times on the network,

with different random initializations. The number of times the sub-game is repeated depends on the size of the network and increases with its size to improve the accuracy of the algorithm. For every pair of vertices i and j that are connected by an edge, we compute the frequency that i and j end up with using the same strategy among all 1000 simulations and normalize it and call the normalized value as $p(i, j)$, defined as the edge closeness. The simulation is run enough number of times to get an accurate value for the edge closeness $p(i, j)$, for all the edges. In each game, we can use more than two strategies to compute the cut and still the game converges. However, we choose 2 strategies to simulate each sub-game in order to decrease the convergence time to reach Nash equilibrium.

The output of the first game is given as input to the second game which now differs for both the algorithms *NashDisjoint* and *NashOverlap*.

Closeness of a vertex v to a community C is defined as follows. A community C is adjacent to a vertex v , if it is in the community C or any of its adjacent vertices are in the community C .

Definition 4. Given a vertex i and a community $C \in \Gamma$, define the closeness $p(i, C)$ to be the sum of probability of closeness of all edges between i and C . That is,

$$p(i, C) = \sum_{\substack{(i,j) \in E \\ j \in C}} p(i, j).$$

Our algorithm for disjoint community detection computes the community structure of the network such that each vertex chooses to be a part of that community to which its closeness is higher than the rest of the communities that it can be a part of. Our algorithm for overlapping community detection computes the community structure of the network such that each vertex chooses to be a part of those

communities to which its closeness is at least α times its maximum closeness to any of its adjacent communities. Thus in both the cases, we chose to pick a community structure which happens to be at equilibrium for all the vertices, such that no vertex would like to change its community(ies) at equilibrium.

4.1.1.1 Disjoint Community Detection

The first game outputs the closeness for each edge, which measures the number of sub-games in which each vertex shared the same strategy as its adjacent vertex. Compute the connected components Γ in the network, considering only the network edges (i, j) with edge closeness $p(i, j) > \beta$. We pick $\beta = 0.95$ for all our cases. As we decrease the value of beta, the number of connected components detected after the first game decrease and we are likely to detect the community structure of greater mixing factor, ie., fuzzy community structure. So, we are at disadvantage when we pick a lower value of β when the network has a good community structure.

Let the initial community structure computed using the results of the first game be Γ . Consider the second game in which in each round, each player chooses to join the community to which his closeness is higher and dissociates himself from his previous community. In each round, an arbitrary player k is picked and he computes his closeness $p(k, C)$ to all communities. Note that the closeness to all communities other than his adjacent communities is zero (since $p(k, l) = 0$, $\forall (k, l) \notin E$). The player, since he is strategic and rational, chooses to associate with the community to which his closeness $p(k, C)$ is higher and if it is different from his previous community, he dissociates from his previous community. The utility of the player i towards his new community is his closeness to community C . The game is played until, no player finds it in his best interest to change his

community. We show that the game halts in finite number of rounds, by showing it as a weighted potential game.

Here, the potential function is a mapping of community assignment due to the partition to the sum of the edge-closeness values of the edges which are not cut-edges of that partition. A cut edge is an edge which has its vertices in two different communities. Formally, let Φ be a potential function of a partition as follows : $\Phi : \{C_1, C_2, \dots, C_n\} \mapsto \mathbb{R}$, where C_k is the community of player k .

We now show that when a player k leaves community C_k and moves into community C'_k , the value of the potential function increases and the increase in the potential function Φ is same as the change in the utility of the player k .

Proof. When player k changes its community from C_k to C'_k , let E_c be the cut-edges at the beginning of the round and E'_c be the set of cut edges at the end of the round. So, the change in the potential function is as follows:

$$\sum_{(i,j) \in E \setminus E'_c} p(i,j) - \sum_{(i,j) \in E \setminus E_c} p(i,j) \quad (4.1.7)$$

The internal edges of k at the beginning of the round become cut edges at the end of the round. Some of the cutedges at the beginning of the round become the internal edges of k at the end of the round. Edges which are not adjacent to k do not change their state. So, the above expression can be written as

$$\sum_{(k,l) \in E \setminus E'_c} p(k,l) - \sum_{(k,l) \in E \setminus E_c} p(k,l) \quad (4.1.8)$$

Since, the adjacent edges to k to all its adjacent communities other than C_k and C'_k do not change their state before and after the round, the above expression can be written as

$$\sum_{\substack{(k,l) \in E \setminus E'_c \\ l \in C'_k}} p(k,l) - \sum_{\substack{(k,l) \in E \setminus E_c \\ \forall l \in C_k}} p(k,l) \quad (4.1.9)$$

which is same as

$$p(k, C'_k) - p(k, C_k) \quad (4.1.10)$$

The player leaves the community C_k and moves into the community C'_k , because the sum of its closeness to C'_k is more than that of to C_k . So, the difference (4.1.10) is always positive. So, each time the player changes his community, he maximizes the potential function and the difference of the potential function in each round is same as the change in the utility function of the player which changes its community. So, this game converges to a nash equilibrium. $\square$

At equilibrium, we get a set of subsets of network, which are pairwise disjoint, such that each player do not wish to change his strategy any more. However, these subsets need not be connected. So, we compute the set of connected components of each of the subsets computed by this game and return it as the disjoint community structure detected by our algorithm.

4.1.2 Overlapping Community Detection

The output of overlapping community detection algorithm is a community structure in which a vertex can be a part of more than one community, which we call as a *cover*. Our algorithm *NashOverlap* computes the community structure of the network such that each vertex chooses to be a part of those communities to which its closeness is at least α (where $0 \leq \alpha \leq 1$) times its maximum closeness to any of its adjacent communities.

The first game outputs the closeness for each edge, which measures the number of sub-games in which each vertex shared the same strategy as its adjacent vertex. Compute the connected components Γ in the network, considering only the network

edges (i, j) with edge closeness $p(i, j) > \beta$. We choose β to be either 0.95 or 1 for the considered networks. This partition on the induced network along with the edge closeness values from the first game constitutes the input for the subsequent game.

Consider the following game: in each round, an arbitrary vertex k computes the closeness values to all the communities and associates itself only with the communities which are enough close to it i.e., $p(k, C) \geq \alpha \cdot \max$, where $\max$ is the maximum closeness of k to any community C . If it is previously associated with any community to which its closeness $p(k, C)$ is less than $\alpha \cdot \max$ in the current round, then it dissociates itself from that community. The game is played until, there exists no vertex k such that it needs to change its set of communities. We show that the game ends in finite number of rounds, by showing it as a weighted potential game.

For a given cover, we shall duplicate every vertex in each of its overlapping communities. Let C_k be the set of communities of player k in a given cover. So, in each of its communities $C \in C_k$, a copy of vertex k is put. Define, the utility of the player k as the sum of the probabilities of the edges of k to the communities $C \in C_k$. Define Φ , the potential function as the function from a cover to the sum of the edge-closeness values of the edges which are not cut-edges.

Formally, let Φ be a potential function of a cover as follows : $\Phi : \{C_1, C_2, \dots, C_n\} \mapsto \mathbb{R}$, and is given by

$$\frac{1}{2} * \sum_{\substack{i: i \in V \\ C: C \in \zeta}} p(i, C) \quad (4.1.11)$$

We now show that the difference in the potential function is same as the utility of the player k who changes his communities in that round.

Proof. When player k changes his communities from C_k to C'_k , let us say that the cover changes from ζ to ζ' . Then the difference in the potential function is as follows :

$$\frac{1}{2} * \left\{ \sum_{\substack{i:i \in V \\ C:C \in \zeta'}} p(i, C) - \sum_{\substack{i:i \in V \\ C:C \in \zeta}} p(i, C) \right\} \quad (4.1.12)$$

Note that $p(i, C) = 0$ for all C , which are not adjacent communities of player i .

This is same as

$$\sum_{\substack{C:C \in \zeta' \\ (i,j):(i,j) \in E \\ C:C \in C_i \cap C_j \\ (i,j):(i,j) \in C}} p(i, j) - \sum_{\substack{C:C \in \zeta \\ (i,j):(i,j) \in E \\ C:C \in C_i \cap C_j \\ (i,j):(i,j) \in C}} p(i, j) \quad (4.1.13)$$

where C_i denotes the set of communities of player i . When player k changes in its communities, then only the k 's adjacent edges (k, l) need to be taken into account.

$$\sum_{\substack{l:(k,l) \in E \\ C:C \in C'_k \cap C_l \\ (k,l):(k,l) \in C}} p(k, l) - \sum_{\substack{l:(k,l) \in E \\ C:C \in C_k \cap C_l \\ (k,l):(k,l) \in C}} p(k, l) \quad (4.1.14)$$

which is same as the

$$\sum_{C:C \in C'_k} p(k, C) - \sum_{C:C \in C_k} p(k, C) \quad (4.1.15)$$

So, the above equation gives the difference in the utility of the player k when it changed its set of communities from C_k to C'_k . So, this defines a weighted potential game and hence the game converges. $\square$

At equilibrium, we get a set of subsets of network, which are pairwise need not be disjoint, such that each player do not wish to change his strategy any more. However, these subsets need not be connected. So, we compute the set of connected components of each of the subsets computed by this game and return it as the overlapping community structure detected by our algorithm.

Chapter 5

Disjoint Community Detection

5.1 Overview

In this chapter, an algorithm for the disjoint community detection called *NashDisjoint* is presented. Initially, the benchmarks on which the performance of this algorithm is evaluated are described. There are several benchmarks in the literature, of which two important and standard benchmarks are picked for experiments. A standard measure, Normalized Mutual Information (NMI), used for evaluation is explained in the next section. This measure is used to evaluate the performance of all the algorithms. In the next section, the algorithm is explained followed by its performance evaluation. In the final section, the performance of this algorithm is evaluated on the real social networks, using the measure modularity, which is described in the previous chapter [2](#).

5.2 Benchmarks

One way to test the performance of any community detection algorithm is to run it on a set of specific networks whose solution is known and then we compare the known solution with the solution delivered by the algorithm. In the case of disjoint community detection, a problem with a well-defined solution is a graph with a clear community structure and no overlapping nodes. Many community detection algorithms are based on similar intuitive notions of what a community is, but each of them have a different implementation. So, the scientific community needs to agree on a set of reliable benchmark graphs to evaluate the performance of these algorithms. In this regard, in the literature, researchers have proposed the computer-generated benchmark networks of different variety and can be generated by varying the parameters, along with the network's built-in community structure. These computer generated benchmark networks with their ground truth community structures can be used to evaluate and compare the performance of the current existing community detection algorithms. There are also some benchmark real social networks whose community structure is well defined because of prior information which is available about their system.

5.2.1 Girvan Newmann Benchmarks

These are the computer-generated benchmarks and are very simple in construction. They do not represent the real world networks very much for the following reasons. In the network generated using this benchmark, a) all the communities are of same size and b) all the nodes have the same average degree. This is based on planted partition model [16], where a partition is planted in the network based on the given

parameters.

In the construction of Girvan Newmann (GN) Benchmark [32], the number of vertices and the number of partitions l that the graph is to be divided are assumed, so that the number of vertices in the group i.e., g are predetermined. Any pair of vertices in a given community are linked with a probability p_{in} and any pair of vertices in two different communities are linked with a probability p_{out} . Each community is an Erdos-Renyi random graph with the probability of existence of edge being $p = p_{in}$. If $p_{in} > p_{out}$, then the network will have a clearer community structure.

As a special case of this model, Girvan and Newmann set $n = 128, l = 4, g = 32$ and the average degree k is set to 16. These particular graphs were used to test the performance of community detection algorithms along with other benchmarks.

5.2.2 LFR Benchmarks

This benchmark [49] assumes that the degree distributions and the community size distributions follow a power law distribution with exponents τ_1 and τ_2 respectively. Given, $0 \leq \mu \leq 1$ is the mixing parameter, each vertex has an internal degree of $1 - \mu$ within its community and external degree μ outside its community. Each vertex has a fraction $1 - \mu$ of its edges with the other vertices of its community and a fraction μ with the vertices of the other communities. These benchmarks are generalizations of GN Benchmarks, by introducing power law distributions of degree and community sizes. However, these benchmarks are better than GN Benchmarks and are closer to real world social networks in their network properties.

The graphs are built as follows:

- First, we compute a sequence of community sizes obeying the prescribed power law distribution of exponent τ_2 .
- Each vertex i of a community is given an internal degree $(1 - \mu)k_i$, where k_i is the degree of vertex i , which is taken by a power law distribution with exponent τ_1 and it is fixed once for all. In this way, each vertex i can connect to $(1 - \mu)k_i$ vertices within the community.
- Each vertex is connected to $(1 - \mu)k_i$ randomly picked vertices of the same community. In this way the sequence of internal degrees of each vertex in its community is maintained.
- Each vertex i has an external degree, equal to μk_i (so that the final degree of the vertex is k_i), that are randomly attached to vertices of different communities.

For both *GN* and *LFR* benchmarks, given the above parameters, an interesting question that can be asked is, what is the maximum value of the mixing factor, in order that the communities can be well defined. However, it has been found that the communities are well defined, even when the mixing factor is around 0.75. For large networks, for a given community structure, when the average size of the community is far less than the size of the network, the limit value of μ approaches 1.

5.3 Evaluation Measures

5.3.1 Normalized Mutual Information

Given two partitions say X and Y , we have two different labellings for each vertex in the two partitions. For a given vertex v , let x_v be the community label in the partition X and y_v be the community label in the partition Y . Assuming that each possible labelling to the vertices is a possible value to the random variable, we can consider X and Y as two random variables, which are given the labellings x and y , respectively. The probability distributions are as follows:

$$P(X = x, Y = y) = \frac{n_{xy}}{n} \quad (5.3.1)$$

$$P(X = x) = \frac{n_x^X}{n} \quad (5.3.2)$$

$$P(Y = y) = \frac{n_y^Y}{n} \quad (5.3.3)$$

where n_x , n_y and n_{xy} are the sizes of the clusters labelled by x , y , and of their overlap respectively. The mutual information $I(X, Y)$ of two random variables is defined as

$$I(X, Y) = \sum_x \sum_y P(x, y) \log \left(\frac{P(x, y)}{P(x)P(y)} \right) \quad (5.3.4)$$

This value is the difference between the shannon entropy of the random variable X and the conditional entropy of X given Y . However, the mutual information has the same value for all the partitions which are derived from the original partition.

So, Danon et al [18] proposed a new measure, normalized mutual information

$$I_{norm}(X, Y) = \frac{2I(X, Y)}{H(X) + H(Y)} \quad (5.3.5)$$

This value equals 1 when the partitions are identical and is zero when the partitions are completely independent to each other.

5.4 Algorithm *NashDisjoint*

The algorithm *NashDisjoint* for disjoint community detection is as follows.

Algorithm 1 *NashDisjoint*

- 1: $G(V, E)$: is the input network
  - 2: **for** $v \in V$ **do**
  - 3:   Pick one of the strategies s_v in $\{a, b\}$ uniformly at random
  - 4: **end for**
  - 5: **for** $iter \in 1000$ **do**
  - 6:   **for** $v \in V$ **do**
  - 7:   choose s_v such that $u_v(s_v, s_{-v})$ is maximum
  - 8:   **end for**
  - 9:   update the closeness probabilities $p(i, j)$ for all edges
  - 10: **end for**
  - 11: Compute the connected components of the graph induced on the network with
edges of closeness probability $p(i, j) > \beta$
-

```

12: repeat
13:   for  $v \in V$  do
14:      $T$  is the set of  $v$ 's adjacent communities
15:     for  $C \in T$  do
16:       Compute the closeness function  $p_v(C)$ 
17:     end for
18:     Let  $C' = \arg_{C \in T} \max p_v(C)$ 
19:     Add  $v$  to community  $C'$  and remove  $v$  from its old community.
20:   end for
21: until partition computed is the same in two consecutive iterations
22: List out the connected components in the final partition as the output

```

5.4.1 Time Complexity

As a part of the first game, we run 1000 subgames. The time complexity of each subgame is $O(n) + O(k_1 \cdot m)$, where k_1 is the number of rounds a subgame takes to converge to a nash equilibrium. The complexity of the second game is $O(k_2 \cdot m)$ where k_2 is the number of rounds the second game takes to reach a nash equilibrium. The total time complexity of the algorithm *NashDisjoint* is $O(1000 * k_1 \cdot m) + O(k_2 \cdot m)$.

5.5 Results

To test the effectiveness of our algorithm, we used the Girvan and Newman benchmark graphs and gave our results against the other algorithms in the first subsection

Figure 5.1: Community Structure detected by *NashDisjoint* on Girvan Newmann Benchmark at $\mu = 0.1$

and the results run on LFR benchmark graphs [49], which resemble real social networks as they have power-law distributions on vertex degrees and community sizes in the subsequent subsection.

5.5.1 Tests on Girvan and Newman benchmark

Algorithm is run on the GN Benchmark networks with number of vertices set to 128 and varying the mixing factor parameter from 0.1 to 0.5. The following figures 5.1, 5.2, 5.3 are the community structures in the network detected using our algorithm varying the mixing factor μ at 0.1, 0.3 and 0.5.

5.5.2 Tests on LFR Benchmark

In our simulations, we pick the following parameters in generating the LFR benchmark graphs: The exponent of the degree distribution is 2 and the exponent of the

Figure 5.2: Community Structure detected by *NashDisjoint* on Girvan Newmann Benchmark at $\mu = 0.3$

Figure 5.3: Community Structure detected by *NashDisjoint* on Girvan Newmann Benchmark at $\mu = 0.5$

community size distribution is 1. The graph structures are specified as follows.

- 1000 nodes with small communities: the community size ranges from 20 to 50.
- 1000 nodes with large communities: the communities of size ranges from 20 to 100.
- 5000 nodes with small communities: the community size ranges from 20 to 50.
- 5000 nodes with large communities: the communities of size ranges from 20 to 100.

The average degree and the maximum degree are 20 and 50, respectively, for all of the above four scenarios. Another important parameter is the mixing parameter μ , which denotes that each vertex shares a fraction of $1 - \mu$ of its edges with other vertices of its own community and a fraction of μ with the rest vertices of the graph. The mixing parameter captures robustness of communities; for instance, when the mixing parameter approaches 0.5, the community structures become fuzzy.

The experiments to explore the performance of community detection algorithms are conducted on these benchmark graphs for several reasons. First, these benchmarks come along with a ground truth on the community structures so that the performance of the algorithms can be evaluated directly against the ground truth community structures. Second, since these benchmark graphs resemble real social networks: both the community sizes and the degrees of nodes follow a power-law distribution, they provide a good challenge to the algorithms which claim to detect the communities in real networks. Finally, these graphs can be generated in linear

time with respect to the number of edges of the graph.

To evaluate the performance of an algorithm, we adopt a popular measure, called normalized mutual information (NMI), introduced by Danon et al. [18]. NMI is a measure of similarity of partitions from information theory. It compares two partitions and gives a measure of how much similar the two partitions are. The NMI value lies between 0 and 1; when the two partitions are identical, the NMI equals 1.

All the softwares for the algorithms taken for comparison are available for free on the web. In each setting, we picked 10 graphs for the evaluation and comparison of these algorithms with our algorithm.

- We run *Louvain* algorithm [10] on each graph for once. For each graph, the algorithm generates a hierarchy of partitions and we pick the partition that gives the maximum NMI with respect to that of the ground truth community structure. Thus, we note the NMI values of the best partition in the hierarchy returned for each of the 10 graphs and recorded their average.
- We ran the *CFinder* algorithm [2] one time for each of the values $k = 3, 4, 5, 6, 7, 8, 9, 10$ on each graph and compute the NMI value for partition corresponding to each of the k values. We pick the partition that gives the maximum NMI value with respect to the ground truth community structure. We repeat the same for all the 10 graphs and record average NMI.
- For both the algorithms, *Infomap* [74] and *Infomod* [73], it is mentioned in their manuals to allow the algorithm to make at least 10 attempts to pick the best partition. So, we set the parameter to 10 and the parameter allows the algorithm to repeat for 10 times and returns the best partition for that

Figure 5.4: Performance of *NashDisjoint* on 1000 node network and large communities

particular graph. We repeat the same for each of the 10 graphs and record the average NMI.

- For *Markov Cluster* algorithm [23], we set the exponent k value to 1.4, 1.6, 1.8, 2.0, 2.2, 2.4, 2.6 and 3.0 and run the algorithm once on each graph for each exponent value. We record the NMIs of the partitions returned for each of the k value and picked partition that gives the maximum NMI value with respect to the ground truth community structure. We repeat this over each of the 10 graphs and record the average NMI.
- For *NashDisjoint* algorithm, we repeat the algorithm for 10 graphs and compute the average NMI. Note that there is no parameter involved in detecting the community structure.

5.5.3 Performance Analysis

We choose LFR Benchmarks to analyze the performance of our algorithm against other algorithms, as it helps in identifying the best algorithm that can discover the

Figure 5.5: Performance of *NashDisjoint* on 1000 node network and small communities

Figure 5.6: Performance of *NashDisjoint* on 5000 node network and large communities

Figure 5.7: Performance of *NashDisjoint* on 5000 node network and small communities

communities in real world networks. The performance of the algorithm *NashDisjoint* is as follows:

- 1000 nodes with large communities (figure 5.4): *Infomap*, *Louvain*, *NashDisjoint* are the best performing algorithms in order. *CFinder* is the worst performing of all. *Infomod* and *Markov Cluster* are the slow running algorithms of all. *Infomap*, *Louvain* are the fastest running algorithms of all.
- 1000 nodes with small communities (figure 5.5) : *Infomap* and *NashDisjoint* and *Louvain* are the best performing algorithms in order, and *Infomod* is the worst performing of all. *Markov Cluster*, *CFinder*, *Infomod* takes a lot of time and *Infomap*, *Louvain* are the fastest running algorithms. As the community size decreases and the size of the network remains same, the performance of *NashDisjoint*, *Markov Cluster*, *Infomap*, *CFinder* is found to increase and the performance of *Infomod* is found to decrease.
- 5000 nodes with large communities (figure 5.6): *NashDisjoint* performs the same way as does *Infomap* till the mixing factor 0.65 and then *Infomap* out-

performs. *Infomod* and *CFinder* are the worst performing algorithms. *Infomap* and *Louvain* are the fastest running algorithms. *NashDisjoint* is faster compared to *Infomod*, *CFinder* and *Markov Cluster*. As the network size increases, keeping the community size intact, the performance of *NashDisjoint*, *CFinder*, *Infomap*, *Markov Cluster* became better and the performance of *Infomod*, *Louvain* fell down.

- 5000 nodes with small communities (figure 5.7): *NashDisjoint* performs the same way as does *Infomap* till the mixing factor 0.6 and *Infomap* performs well over *NashDisjoint* after mixing factor 0.6. *Markov Cluster* and *Louvain* works well over the other algorithms. *Infomod* is the worst performing algorithm. With increase in the size and the community sizes are set to small, the performance of *Infomod* falls down and the performance of *Infomap*, *NashDisjoint*, *Markov Cluster* increased and for *CFinder*, the performance is almost the same. With size of the network fixed, and the community size decreased from large to small, the performance of *NashDisjoint* fell down for mixing factors more than 0.45 and the performance of *Louvain*, *CFinder*, *Infomap*, *Markov Cluster* increases.

Overall, *Infomap*, *NashDisjoint* are the best performing algorithms on all the benchmarks. *Infomod* performed very bad as the size of the network becomes bigger and becomes more worse for the case of smaller communities. *Markov Cluster*, *Infomod* and *NashDisjoint* are the slow running algorithms. However, in comparison to the other non-scalable game theory based algorithms in the literature, our algorithm not only improves on the performance of the previous game theory based algorithms but also they are scalable. Also, our algorithm takes more time in the

phase one, which can be very much parallelised.

5.5.4 Tests on Real Social Networks

The modularity values of community structure detected in the real social networks : karate, football, dolphins, celegan-neural, jazz, email, netscience is in comparison to that of the values given by the *Louvain* algorithm, the best algorithm known for modularity optimization. The modularity values are given in the following table :

network	$ V $	$ E $	<i>Louvain</i>	<i>NashDisjoint</i>
karate	34	78	0.418	0.415
football	115	615	0.602	0.533
dolphins	62	159	0.5185	0.518
celegan-neural	297	2345	0.3871	0.2094
jazz	198	2742	0.445	0.438
email	1133	10903	0.549	0.5204
netscience	1589	2742	0.959	0.8632

For a quantitative comparison between our algorithm and the Louvain algorithm, one of the best modularity optimization algorithms on real world networks, we ran our experiments on some of the standard datasets like karate, football, dolphins, celegan-neural, jazz, email and netscience. As it can be seen from this table, the modularity values for each of the datasets for both the algorithms do not differ by more than 0.1, which is a very small margin. However, in each of the datasets, the number of communities detected by both the algorithms is always the same.

5.6 Summary

In this chapter, *NashDisjoint*, an algorithm for disjoint community detection is presented. It is a heuristic algorithm, based on game theory, which is scalable to larger graphs. The algorithm is simple to implement and it performs similar to the best community detection algorithm so far *Infomap* till $\mu = 0.65$ or $\mu = 0.7$ for 5k node graphs and $\mu = 0.55$ or $\mu = 0.6$ for 1000 node graphs. It is the first game theory based community detection algorithm which runs in time pseudo-linear in the number of edges. The modularity values of some real social networks for the community structures detected by *NashDisjoint* are computed. These values are very much similar to that of the modularity values returned by the one of the best modularity optimization algorithms.

Chapter 6

Overlapping Community Detection

6.1 Overview

An algorithm for overlapping community detection is presented in this chapter. In the next section, the benchmarks and the algorithms in the literature that are used to generate such benchmarks are explained. In the subsequent section, Normalized Mutual Information (NMI), a measure to compare the similarity of the covers and evaluate the performance of the algorithms is explained. The algorithm *NashOverlap* and the results of its comparison to the state of the art algorithms on LFR benchmarks is given. It is shown that the algorithm performs well in comparison to the state of the art algorithms.

6.2 Benchmarks

To evaluate the performance of any overlapping community detection algorithm, we need the computer-generated benchmark networks with their in-built community structure, so that we compare their ground truth community structure with the community structure delivered by the algorithm. The paper by Lancichinetti et al [47] gave an algorithm that generates the benchmark networks in which each node has membership in more than one community. These benchmark graphs are useful in testing the overlapping community detection algorithms for their performance and the quality of detection of the overlapping communities in real world networks. In the literature, apart from synthetic benchmarks, real networks are also used to test the performance of overlapping algorithms. In such real networks, communities are well defined with the information about the system in hand.

The benchmark [47] assumes that the degree distributions and the community size distributions follow a power law distribution with exponents τ_1 and τ_2 respectively. Given, $0 \leq \mu \leq 1$ is the mixing parameter, each vertex has an $1 - \mu$ fraction of its degree within its community and μ fraction of its degree outside its community. Each vertex has a fraction $1 - \mu$ of its edges with the other vertices of its community and a fraction μ with the vertices of the other communities. To construct the graphs with overlapping nodes, along with the other usual parameters, two other parameters on and om are set where on is the number of overlapping nodes and om is the maximum community membership of a node. However, in these networks, any node has either the community membership of 1 or om and the community membership does not lie between 1 and om .

The graphs are built as follows [47]:

- Each vertex v_i is assigned a number om_i , which denotes the number of communities that the vertex can be a member of. If $om_i == 1$ for each node v_i , the network generated is similar to that of the benchmark [49].
- A sequence of community sizes obeying the prescribed power law distribution is extracted. This is done by picking random numbers from a power law distribution with exponent τ_2 . In this case, the sum of community sizes is equal to the sum of node memberships.
- Each vertex v_i is assigned a degree k_i which is a random number picked from a power law distribution of exponent τ_1 and the vertex is given an internal degree $(1 - \mu)k_i$, where k_i is the degree of vertex v_i and it is fixed once for all.
- Each vertex v_i has an external degree, equal to μk_i (so that the final degree of the vertex is k_i), that are randomly attached to vertices of different communities.
- Rewiring between the vertices is done again and again until the constraints are satisfied.

6.3 Measures

6.3.1 Normalized Mutual Information

The paper by Lancichinetti et al [48] gave a definition to compute the normalized mutual information to find the similarity between the two covers. Suppose a vertex belongs to more than two communities, then for each vertex v and community C_k

in the given cover say X , a random variable say X_k is defined, which stores the probability of whether the vertex v is in the community C_k . This probability is computed as the fraction of number of elements in the community C_k over the total number of vertices. Similary for the cover Y , the random variable Y_l takes values 1 or 0 depending on whether the vertex v is present in community C_l , in the cover Y . To infer X_k from a given Y_l , we have this information :

$$H(X_k|Y_l) = H(X_k, Y_l) - H(Y_l) \quad (6.3.1)$$

To infer X_k from a given cover Y , we need to find C_l that gives the minimum difference for the above equation, because, if C_k and C_l are both the same, then $H(X_k, Y_l) = H(Y_l)$ and hence $H(X_k|Y_l)$ is zero. That is, we can get all the information about X_k from Y_l . Then we normalize $H(X_k|Y)$ by dividing it by $H(X_k)$. Thus we compute the normalized conditional entropy of cover $\mathbf{X}$ from Y , as follows :

$$H(\mathbf{X}|\mathbf{Y})_{norm} = \frac{1}{|\mathbf{X}|} \sum_k \frac{H(X_k|\mathbf{Y})}{H(X_k)} \quad (6.3.2)$$

Similarly, the normalized conditional entropy of cover $\mathbf{Y}$ from $\mathbf{X}$ is computed and the normalized mutual information is computed as follows:

$$NMI(X|Y) = 1 - \frac{1}{2}[H(\mathbf{X}|\mathbf{Y})_{norm} + H(\mathbf{Y}|\mathbf{X})_{norm}] \quad (6.3.3)$$

This NMI value lies between 0 and 1, 1 happens to be the value when there is a perfect matching.

6.4 Algorithm *NashOverlap*

Algorithm 2 *NashOverlap*

```
1:  $G(V, E)$  is the input network

2: for  $v \in V$  do

3:   Pick one of the strategies  $s_v$  in  $\{a, b\}$  uniformly at random

4: end for

5: for  $iter \in 1000$  do

6:   for  $v \in V$  do

7:     choose  $s_v$  such that  $u_v(s_v, s_{-v})$  is maximum

8:   end for

9:   update the closeness probabilities  $p(i, j)$  for all edges

10: end for

11: Compute the connected components of the graph induced on the network with
    edge closeness probability  $p(i, j) > \beta$ 

12:  $\alpha$  is a parameter to be tuned.
```

```

13: repeat
14:   for  $v \in V$  do
15:     Let  $T$  be set of  $v$ 's adjacent communities
16:     for  $C \in T$  do
17:       Compute the closeness function  $p_v(C)$ .
18:     end for
19:     Compute  $\max_C \{p_v(C)\}$ 
20:     Let  $R = \{C | p_v(C) > \alpha \cdot \max_C(p_v(C))\}$ 
21:     Assign  $R$  as  $v$ 's communities
22:   end for
23: until Cover computed is the same in two consecutive iterations
24: List out the connected components in the final cover as the output

```

6.4.1 Time Complexity

The number of rounds that the second game needs increases with decrease in α . If α is one, then the algorithm *NashOverlap* is same as *NashDisjoint*. As a part of the first game, we run 1000 subgames. The time complexity of each subgame is $O(n) + O(k_1 \cdot m)$, where k_1 is the number of rounds a subgame takes to converge to a nash equilibrium. The complexity of the second game is $O(k_2 \cdot m)$ where k_2 is the number of rounds the second game takes to reach a nash equilibrium. The total time complexity of the algorithm *NashOverlap* is $O(n) + O(1000 \cdot k_1 \cdot m) + O(k_2 \cdot m)$.

6.5 Performance Results

6.5.1 Experimental Setup

To study the performance of our algorithm on overlapping community detection, we conducted extensive experiments on synthetic networks of size 1000, and 5000 and compared its performance with some of the existing well performing algorithms.

Lancichinetti et al [47], provided the benchmarks to test the performance of any overlapping community detection algorithm on networks resembling the real world networks. This software is provided for free and allows us to construct the graphs with overlapping community structure. These benchmark networks are generated along with a ground-truth community structure against which the accuracy of the community structures generated by the other algorithms can be evaluated.

This benchmark software called as LFR benchmarks in short, allows us to construct the graphs for overlapping network structures by means of two parameters: number of overlapping nodes, O_n and overlapping membership, O_m . For example, when we set O_n to 10, then the software generates exactly 10 nodes in the network, such that each node is a part of more than one community. When we set the membership parameter O_m to 3, then any overlapping node can be a part of at most 3 communities only. In our simulations, we pick the following parameters in generating the LFR benchmark graphs for overlapping communities: The exponent of the degree distribution is 2 and the exponent of the community size distribution is 1. The graph structures are specified as follows.

- 1000 nodes with small communities: the community size ranges from 20 to 50,

- 1000 nodes with large communities: the communities of size ranges from 20 to 100,
- 5000 nodes with small communities: the community size ranges from 20 to 50.
- 5000 nodes with large communities: the communities of size ranges from 20 to 100.

For each of the above four scenarios, we varied the mixing factor parameter μ from 0.1 to 0.5 and we varied the overlapping membership parameter from 2 to 8, thus considering a total of 140 different scenarios. In each scenario, the number of overlapping nodes is fixed at 10% of the total number of nodes. The average degree and the maximum degree are fixed at 20 and 50, respectively, for all of the above scenarios.

6.5.2 Tests on LFR overlapping benchmark Networks

We compared our algorithm with four well-performing algorithms on synthetic networks. All the results are averaged over 10 different LFR benchmark graphs.

- We ran the *CFinder* algorithm proposed by [66] one time for each of the values $k = 3, 4, 5, 6, 7, 8, 9, 10$ on each graph and compute the NMI value for partition, against the ground truth community structure, for each of the k values. We pick the partition that gives the maximum NMI value, over all possible k values. We repeat the same for all the 10 graphs and record average NMI.

- We ran *SLPA* algorithm [42] on each value of $r = 0.05, 0.10, 0.15, 0.20, 0.25, 0.30, 0.35, 0.40, 0.45$ for 10 times, which is a parameter which decides the number of times the algorithm can be repeated to improve its accuracy. This number allows *SLPA* to make 10 attempts to get the best cover and hence the best NMI value for the given value of r . The best NMI recorded is the maximum NMI picked over all values of r . Similarly, for each of the 10 different graphs, we find the best NMI and record the average NMI.
- For the execution of the algorithm *COPRA* [36], *COPRA* manual has provided a membership parameter v which is the maximum number of communities of any node in the given graph. Since, the number of communities for the ground-truth is known before hand, we can as well set the parameter v to the overlapping community membership of the ground-truth community structure and run the algorithm for each graph. Instead, we considered all possible values of v , i.e., from 1 to 10 and ran *COPRA* algorithm on each value of v . Since they claimed in the manual, that the algorithm *copra* picks the best cover if we enable the parameter mo , we performed experiments with membership parameter v set to every value from 1 to 10 and setting the parameter mo . On each graph, with the above setting, we ran the *COPRA* algorithm for 10 times and picked the returned cover and noted its NMI. This is repeated for 10 different graphs and we recorded the average NMI over all the 10 graphs.
- In case of *OSLOM* algorithm [50, 46], the parameter r is set to 10, which is the number of runs for the first hierarchical level. Higher the value r , the accurate the results are. The parameter hr is set to 2 and all the other

parameters *infomap*, *copra*, *louvain* are set to 1. The algorithm explores the network 10 times using all the parameters each time and returns the final output which is formed by the best modules found with all these methods. We record the NMI returned for each graph and record the average NMI over all the 10 graphs.

- For the execution of *NashOverlap*, we run Nash Dynamics for 1000 iterations and collect the tie-strengths of all the edges. These edge tie-strengths are given as input to Phase two of our algorithm and ran it for all the parameters from 0.20 to 0.60, in steps of 0.02. We recorded the NMI of the cover that gives the best NMI against the ground truth community structure. This is repeated for all the 10 graphs. The average NMI over all the 10 graphs is then recorded.

Figure 6.1: These figures show the comparison of community structures resulting a) at the end of first game of the algorithm *NashOverlap*, b) *NashOverlap* at its best and c) *NashOverlap* at $\alpha = 0.50$ for network size 5000 and community size in the range [20-100].

Figure 6.2: These figures show the comparison of community structures resulting a) at the end of first game of the algorithm *NashOverlap*, b) *NashOverlap* at its best and c) *NashOverlap* at $\alpha = 0.50$ for network size 5000 and community size in the range [20-50].

Figure 6.3: These figures show the comparison of community structures resulting a) at the end of first game of the algorithm *NashOverlap*, b) *NashOverlap* at its best and c) *NashOverlap* at $\alpha = 0.50$ for network size 1000 and community size in the range $[20-100]$.

Figure 6.4: These figures show the comparison of community structures resulting a) at the end of first game of the algorithm *NashOverlap*, b) *NashOverlap* at its best and c) *NashOverlap* at $\alpha = 0.50$ for network size 1000 and community size in the range [20-50].

Figure 6.5: Performance of *NashOverlap* for 1000 node networks, small communities and $\mu = 0.1$

Figure 6.6: Performance of *NashOverlap* for 1000 node networks, small communities and $\mu = 0.2$

Figure 6.7: Performance of *NashOverlap* for 1000 node networks, small communities and $\mu = 0.3$

Figure 6.8: Performance of *NashOverlap* for 1000 node networks, small communities and $\mu = 0.4$

Figure 6.9: Performance of *NashOverlap* for 1000 node networks, small communities and $\mu = 0.5$

Figure 6.10: Performance of *NashOverlap* for 5000 node networks, small communities and $\mu = 0.1$

Figure 6.11: Performance of *NashOverlap* for 5000 node networks, small communities and $\mu = 0.2$

Figure 6.12: Performance of *NashOverlap* for 5000 node networks, small communities and $\mu = 0.3$

Figure 6.13: Performance of *NashOverlap* for 5000 node networks, small communities and $\mu = 0.4$

Figure 6.14: Performance of *NashOverlap* for 5000 node networks, small communities and $\mu = 0.5$

Figure 6.15: Performance of *NashOverlap* for 1000 node networks, large communities and $\mu = 0.1$

Figure 6.16: Performance of *NashOverlap* for 1000 node networks, large communities and $\mu = 0.2$

Figure 6.17: Performance of *NashOverlap* for 1000 node networks, large communities and $\mu = 0.3$

Figure 6.18: Performance of *NashOverlap* for 1000 node networks, large communities and $\mu = 0.4$

Figure 6.19: Performance of *NashOverlap* for 1000 node networks, large communities and $\mu = 0.5$

Figure 6.20: Performance of *NashOverlap* for 5000 node networks, large communities and $\mu = 0.1$

Figure 6.21: Performance of *NashOverlap* for 5000 node networks, large communities and $\mu = 0.2$

Figure 6.22: Performance of *NashOverlap* for 5000 node networks, large communities and $\mu = 0.3$

Figure 6.23: Performance of *NashOverlap* for 5000 node networks, large communities and $\mu = 0.4$

Figure 6.24: Performance of *NashOverlap* for 5000 node networks, large communities and $\mu = 0.5$

6.5.3 Tests on LFR Disjoint Benchmark Networks

NashOverlap performs almost the same as *NashDisjoint* algorithm for networks having community structure with no overlapping nodes and its performance increases in similar to that of *NashDisjoint* as the community structure of the networks become clearer. On benchmark networks with community structure free of overlapping nodes (normal benchmark) [49], our algorithm performs better than other algorithms, which are considered in the previous chapter of disjoint community detection. The performance of this algorithm is same as that of its performance at $\alpha = 0.5$, and are as shown in the figures 6.25, 6.26, 6.27, 6.28. The reason is for a community structure which has no overlapping nodes, any vertex is more likely to be very close to only one of its adjacent communities, thus minimizing the possibility of overlap.

Figure 6.25: Performance of *NashOverlap* vs Other Algorithms on LFR Benchmark Network of 1K nodes and large sized communities, with disjoint community structure

Figure 6.26: Performance of *NashOverlap* vs Other Algorithms on LFR Benchmark Network of 1K nodes and small sized communities, with disjoint community structure

Figure 6.27: Performance of *NashOverlap* vs Other Algorithms on LFR Benchmark Network of 5K nodes and large sized communities, with disjoint community structure

Figure 6.28: Performance of *NashOverlap* vs Other Algorithms on LFR Benchmark Network of 5K nodes and small sized communities, with disjoint community structure

6.5.4 Performance Analysis

This section explains the performance of our algorithm with the other algorithms on the above LFR benchmarks. In all the above 152 scenarios that we have chosen, when we have performed the experiments as stated, *NashOverlap* works very well over these algorithms. A few observations from the plots are as follows:

- The best performance of *NashOverlap* is not very different from its performance at $\alpha = 0.5$ as can be seen from the Figures 6.3, 6.1, 6.4, 6.2
- The community structure at the end of the first game, the best community structure given by the algorithm *NashOverlap* (for all the possible α) and the community structure given by the algorithm *NashOverlap* at $\alpha = 0.50$ are illustrated in the Figures 6.4, 6.2, 6.3, 6.1.
- 1000 node large communities (Figures 6.15, 6.16, 6.17, 6.18, 6.19): *NashOver-*

lap performs well over all the other algorithms. *COPRA*, *OSLOM*, *SLPA*, *CFinder* happen to be the best performing algorithms in order. As the mixing factor increases, the performance of all the other algorithms appear to be almost the same except *Cfinder*, which happens to be the worst performing algorithm and *NashOverlap* is the best performing algorithm.

- 1000 node small communities (Figures 6.5, 6.6, 6.7, 6.8, 6.9): *NashOverlap* happens to be best performing algorithm for all the mixing factors from 0.1 to 0.5. *OSLOM*, *CFinder*, *COPRA*, *SLPA* are the best performing algorithm in order after *NashOverlap*. With increase in the mixing factor, the algorithms *OSLOM*, *COPRA*, *SLPA* performance appears almost same, while the performance of *CFinder* becomes worse.
- 5000 node small communities (Figures 6.10, 6.11, 6.12, 6.13, 6.14): *NashOverlap* happens to be the best performing community detection algorithm of all with *SLPA* performing the worst of all the algorithms.
- 5000 node large communities (Figures 6.20, 6.21, 6.22, 6.23, 6.24): *NashOverlap* happens to be the best performing community detection algorithm of all with *CFinder* performing the worst of all the algorithms, which even worsens with increase in mixing factor.
- For both the network sizes 1000 and 5000, the performance in the case of small communities is much better than that in the large communities.
- The performance of all the algorithms deteriorates with increase in the mixing factor. For *CFinder*, the performance of the algorithm deteriorates with increase in the network size for both the variations in the community sizes.

The performance of *NashOverlap* increases with increase in the network size.

- The performance of all the algorithms falls down with increase in overlapping membership.

6.6 Summary

NashOverlap, the algorithm for overlapping community detection is presented in this chapter. As a result of extensive experiments on LFR benchmark networks, this algorithm is observed to perform well on LFR benchmarks, when compared to the state of the art algorithms. LFR Benchmarks are computer generated benchmark networks which are very similar to the real social networks. The algorithm performs quite well compared to the state of the art algorithms using NMI measure. The same algorithm can be used as disjoint community detection algorithm, when the network has a clear community structure. It can be easily extended to weighted and directed networks as well. The algorithm is very simple to implement and is scalable to larger graphs.

Chapter 7

DBLP Collaboration Network: A Case Study

7.1 Overview

DBLP (*Digital Bibliography and Library Project*) is the Computer Science Bibliography. In this section, we identify and study the collaboration groups (communities) in DBLP collaboration network using our algorithm *NashOverlap*. We also study the distributions of both the sizes of collaboration groups and the overlapping membership in the detected community structure. We compared the results with that of the best algorithm, COPRA which is found to work reasonably good on synthetic datasets and on large datasets and since it is able to identify good enough collaboration groups in DBLP, compared to the other community detection algorithms considered in this paper.

We considered the dblp dataset [1] of only the papers published in tier one rank conferences in almost all the fields in computer science from the year 1966

to 2014. We considered 69 conferences on the whole from the following fields : algorithmic game theory, general algorithms and data structures, quantum computing, cryptography, computational complexity, computational geometry, databases, computer architecture, artificial intelligence, principles of programming languages, automata theory, computer networks, multimedia, computer vision, pattern recognition, Internet Fault tolerance, Data mining, machine learning. We constructed the DBLP network as a weighted graph with vertices as authors. An edge is put between a pair of authors if they have at least one paper that they have coauthored together. We constructed two datasets based on this DBLP data as follows:

- The first dataset consists of the authors who have published papers from 1966 to 2014 in only top ranked computer science conferences. We have considered 69 conferences altogether. The authors from the dataset form the vertices of the network. We put an edge between a pair of authors if they have coauthored a paper together. The weight of the edge is the number of papers that they have coauthored together. There is no edge between the authors, only if they have never coauthored any paper. The dataset has 122684 authors and 384320 edges. Excluding self-loops, the average degree of this network is 6 and the maximum degree is 284.
- The second dataset considers only the papers of top ranked conferences whose authors have published at least 10 papers from 1966 in only top ranked computer science conferences. So, the set of authors considered in this dataset is a subset of the set of authors of the above dataset. The authors in this dataset are the authors who have published papers only with the authors who also have published at least 10 papers from 1966 in only tier one computer science

conferences. The weight of the edge is the number of the papers that both the authors have coauthored. There is no edge between the authors, if they have never coauthored any paper. Note that even if a pair of authors have not coauthored 10 papers together, then there may be an edge between them. The dataset has 7349 number of authors and 24161 number of edges. Excluding self-loops, the average degree of this network is 5.6 and the maximum degree is 86.

On the first dataset, it took 9000 seconds to run 1000 subgames of the first game and compute the closeness probabilities for all the edges. For the second game, the algorithm took 50 seconds to compute the final community structure. In the first game, observe that each subgame is independent of each other and takes around 2.5 million rounds to reach a nash equilibrium. The network is very fuzzy and the the average degree differs from the maximum degree by a large amount and the distribution of community sizes and the distribution of node degrees are found to follow a very steep power law distribution.

On the second dataset, the first game took 558 seconds to run all the 1000 subgames of nash dynamics (actually, 100 subgames is enough for the network size of 7000 if a clearer and well defined community structure is present). The second game, takes around 20 seconds to detect the communities. Each iteration in phase one is independent of each other, and it takes on average 123,000 rounds to converge to a nash equilibrium. At the end of the first game, the number of nash components detected is 4366. That is, the number of disjoint communities found at the end of the first game are 4366 and 99.99% of the sizes of all the disjoint communities which are output at the end of the first game, are all less than 15.

In the second game, we choose to analyze the network with $\alpha = 0.5$ for both the datasets where the identified collaboration groups are much clearer.

We visualize the collaboration groups using the Gephi tool. Gephi is a tool to visualize and manipulate the networks. We used *Fruchterman-Reingold*, a forced layout algorithm to visualize the network. In the figures that visualize each community, the size of vertices is proportional to their degree, and the thickness of the edges is proportional to their tie-strengths $t(i, j)$ computed in the algorithm. All the overlapping nodes are shown red in color. When we say, we load at least one community into gephi, we are actually loading the induced subgraph of the network on the vertices of all those communities into gephi. Fruchterman Reingold is a force directed layout. It simulates a physical system, in which nodes repulse each other, while edges attract the nodes they connect. These forces create a movement that converges to a balanced state. The final configuration of each of the layout algorithms is expected to explore and understand the network structure. We thus use the tool gephi, in each case to justify the quality of our community detection algorithm.

We justify the strength of our community detection algorithm in the following ways. We compare our community structure with that of the community structure identified by *COPRA*, for both the datasets. For each dataset, we manually, look into the top collaboration groups (ordered in terms of size) and find that our algorithm has identified more meaningful community structure than *COPRA*. We find that our algorithm has detected more collaboration groups of significant size each of which carries an identity, compared to that of *COPRA*. Also, the communities detected by our algorithm are much bigger in size with a least number of outlier

nodes. The community structure is likely to have more mixing factor than that of detected by *COPRA*. It means, that the collaboration groups are big enough in size with several researchers in each, so that almost no researcher who concerns this collaboration group is ignored. We visualise the community structures detected for both the datasets for both the algorithms and the significant collaboration groups detected in the community structures detected by both the algorithms. For visualization convenience, we have only shown the high degree researchers in each of the identified collaboration groups, by loading the gephi with the induced network of the collaboration group with the higher degree nodes.

7.2 Determining Collaboration Groups

Our algorithm detected a large collaboration group of size 38% of the network size for the large dataset and 22% of the network size for the small dataset. The large collaboration group detected for each dataset has the researchers, mostly based in United States, from almost all the fields, and the second largest collaboration group is far smaller in size than the first. This finding is in congruence with the statement from the paper [62] which says that in all the collaboration networks studied, they identified a giant component which is a considerable fraction of the network size and the second group happened to be far smaller than the first. The existence of a giant component and the subsequent smaller components is due to the percolating regime characteristic of collaboration networks. When the network has a fewer edges, there exist a large number of small connected clusters. With the increase in the number of edges, most of these small clusters group to form a giant component, leaving behind smaller clusters which join the giant component

in the future with the increase in the number of connections. This leads to a reasoning that the scientific collaboration network is highly connected and there is a possibility of more interdisciplinary work in the future which is a good sign for the development of science. The diameter of the largest collaboration group for both the datasets is between 9 and 11 and has the average path length between 4 and 5, which strengthens the above argument. Note that, this is because the average number of intermediate researchers that connect any pair of researchers in the giant component is quite less.

Figure 7.3 and Figure 7.1 show the community structures of small and large dataset respectively, identified by the algorithm *NashOverlap* at $\alpha = 0.5$ and $\beta = 1$. In the figures, the vertices represent communities, and the weight of the edge between two vertices is the sum of the weights of the cutedges between the two communities. The size of the vertex is proportional to the size of the community. The weight of the edge shows how close the researchers of both the communities are collaborating with each other. So, the larger the weight of the edge between two communities, the larger the overlap between two communities or the larger the collaboration between the two communities. In the figures, the numbers on the vertices show the size of the community represented by the vertex.

Figure 7.1: Depiction of Community Structure of DBLP collaboration network (large dataset) as detected by *NashOverlap* with vertices as communities and weight of edge between vertices show the sum of the weights of the cutedges between the communities represented by the vertices.

and some of the small collaboration groups which have weaker links with the giant component of the same fields are detected. For the large dataset, a major group at least one for each of the fields cryptography, algorithms, computer networks, complexity are detected.

Figure 7.3: Depiction of Community Structure of DBLP collaboration network (small dataset) as detected by *NashOverlap* with vertices as communities and weight of edge between vertices show the sum of the weights of the cutedges between the communities represented by the vertices (only major communities are shown)

Figure 7.4: WordCloud for the conferences of giant component detected by *NashOverlap* in DBLP collaboration network (small dataset). Observe that only theory based conferences appear much bolder and bigger, whereas the conferences related to industry related fields are comparatively smaller in size or do not exist

The distribution of the fields of the researchers in the giant component and the distribution of the fields identified uniquely by each of the subsequent larger collaboration groups detected for both the datasets differ. This owes to the fact that the fields like computer vision, multimedia, machine learning, software engineering, embedded systems are more industry oriented and any two collaborating researchers of those fields are less likely to stay in academics for a long time and publish. Hence, the possibility of each such researcher having at least 10 publications in top tier conferences is less likely. So, we termed it as the reason for finding several large collaboration groups of those fields in the community structure detected for the large dataset, where as in the case of small dataset, those groups are almost non-existent, not even of lesser size. Also, this is the reason for the giant component in the large dataset to contain the researchers from all the fields equally in proportion, where as the researchers from industry oriented fields are very less

in number in the giant component in the small dataset. This is observed as in the figures 7.2 and 7.4. The average tie-strength of the links between the researchers in most of the collaboration groups of the industry oriented fields is also found to be less compared to the theory fields of computer science. We identified the fields corresponding to each collaboration group by looking at the frequency of the conferences in which those researchers have published, the keywords in their papers and the research interests, affiliations of the researcher.

Figure 7.5: Collaboration group on Databases, ML, AI, Pattern Recognition, Big Data. This is the second largest collaboration group detected on the small dataset

We studied only the collaboration groups of size more than 20 for the case of small dataset. We identified that the collaboration groups in the following fields:

computational geometry, formal methods and automata theory, programming languages, Artificial Intelligence, Machine Learning, Database theory group have a large overlap with the giant component. Figure 7.5 depicts the high degree researchers of second largest collaboration group detected for the small dataset. The researchers in this group are found to be working in various areas: Big Data, Database theory, query optimization, database consistency and efficiency, data streaming, Machine Learning, Artificial Intelligence and Pattern Recognition. The underneathing wordcloud 7.6 corresponds to all the conferences in which the researchers of that group publish and as you can see, all of them are conferences related to Databases, Big Data, machine learning and Artificial Intelligence. This collaboration group has a large overlap with the giant component, which depicts the currently happening percolating regime of these fields in a huge way. It can be seen in a much clearer way in the case of collaboration groups identified for the large dataset. The largest collaboration group has researchers from almost all the fields and shows all the conferences in equally sizes in the conferences wordcloud as is seen in the figure 7.2.

Figure 7.6: WordCloud for the conferences of second largest component detected by *NashOverlap* in DBLP collaboration network (small dataset). Observe that all the conferences are based in Machine Learning, Databases and Artificial Intelligence

Figure 7.7: Two major collaboration groups of sizes 408 and 239 on Database theory and systems are detected on the small dataset. Overlap size is 92 and the number of overlap edges is 187. Only high degree nodes are shown for visualization convenience. Each of the groups have an overlap of 61 and 39 with the giant component respectively. Overlapping nodes are colored red

Figure 7.8: Two major collaboration groups of sizes 222, 219 on Multimedia, Computer Vision are detected on the small dataset. Both are based in Asia, and have very weaker links with the giant component. Most of the high degree researchers are based in Microsoft research Asia. Overlap size is 117 and overlap edges are 291. Overlapping nodes are colored red. Each of the groups has an overlap of 6 and 7 with the giant component respectively.

Figure 7.10: Collaboration group in Computational Geometry of size 133 detected for the small dataset. It has an overlap of 39 with the giant component. Only high degree nodes are shown for visualization convenience

Some of the collaboration groups identified in the case of small dataset have very weak links with the giant component and they are mostly the groups based in east asia. These groups are in the fields of data engineering, multimedia, computer vision and have most of its people from Microsoft research Asia and is dominated by Asians. Two collaboration groups on Media Computing, Data and Web Mining, Pattern Recognition, Vision, Human Computer Interaction are detected as can be seen at figure 7.8. 99% of both the groups are dominated mostly by asian community, and the top degree people in this community are all affiliated to MS Research Asia. Both the groups are highly clustered and have some overlapping

nodes colored red and are shown in the figure 7.8. Both the collaboration groups differ in their research interests and their methods.

Some large collaboration groups of other fields like databases, BigData have researchers from Microsoft and IBM in at least their top 10% set of researchers as can be seen in the figure 7.7. Our algorithm has identified a collaboration group for every possible area of the computer science. Our algorithm gives all the possible collaboration groups, and there is no restriction on the size of overlap detected between pairs of groups. However, the clustering returned by the algorithm depends on the α chosen. The algorithm detects more collaboration groups of large size, however with a lot of overlap when the α is lesser. To detect more fuzzy collaboration groups, we can tune the parameter used in the first game to 0.95 or 0.9.

In most of the cases, each collaboration group happens to represent a field of computer science, as can be seen in figure 7.10 and figure 7.9. However, the collaboration group need not contain all the researchers in the world working in that field. This is intuitive because all the researchers in the world who are working on a common area of interest need not have strong collaborations with each other. The collaboration groups are formed because of the common institution or the common race or the common nationality or the common geographical area or the common culture or the common university or the common research labs or the common time span in which the researchers in that group are active or any combination of the above.

Figure 7.11: Cumulative distribution of Overlapping membership of community structure detected by *NashOverlap* for small dataset at $\alpha = 0.5$. Overlap membership of 2,3,4 is found to more common.

Figure 7.12: Cumulative distribution of Community Size of community structure detected by *NashOverlap* for small dataset at $\alpha = 0.5$. Observe that most of the significant communities are between sizes 50 and 100.

Figure 7.11 and Figure 7.12 shows the cumulative distribution of overlap membership and the cumulative distribution of the community size of the community structure detected for the small dataset.

The figures 7.14,7.13 shows the community structure detected by the algorithms *COPRA*, *NashOverlap* respectively. In the layout, the communities detected by *COPRA* algorithm appear denser as that of *NashOverlap*. There is not much overlap that is detected by *COPRA* algorithm and the communities apart from the giant community are not of significant size.

Figure 7.13: Community Structure of the DBLP Collaboration network (small dataset) detected by *NashOverlap* algorithm at parameter $\alpha = 0.5$ and $\beta = 1$ using *Fruchterman-Reingold* layout algorithm in gephi. Each community is given a separate color and the overlapping nodes are red in color. The community structure given by the *Fruchterman-Reingold* layout and the algorithm *NashOverlap* are in tune with each other. However, the communities are bigger in size and most of the possible communities are identified.

Figure 7.14: Community Structure of the DBLP Collaboration network (small dataset) detected by *COPRA* algorithm at parameter $v = 2$ using *Fruchterman-Reingold* layout algorithm in gephi. Each community is given a separate color and the overlapping nodes are red in color. The community structure given by the *Fruchterman-Reingold* layout and the algorithm *COPRA* are in tune with each other. However, the communities are not bigger in size and all the possible communities are not identified. Also, it did not detect much of the overlap between the pair of collaboration groups.

We find that our algorithm has detected more collaboration groups of significant size each of which carries an identity, compared to that of *COPRA* as shown in the following tables 7.1. Also, the communities detected by our algorithm are much bigger in size with a least number of outlier nodes. The community structure is likely to have more mixing factor than that of detected by *COPRA*. It means, that the

collaboration groups are big enough in size with several researchers in each, so that almost no researcher who concern this collaboration group is ignored. These results are much more pronounced and clearly seen as in the large dataset of the DBLP collaboration network. For the small dataset, in the community structure detected by *NashOverlap*, some hierarchy is also observed in the case of fields :Databases (of sizes :48, 40) , multimedia (of sizes : 67, 58), of computer-aided-verification (of size 42) and of computational geometry (of size 97). Similarly, for the large dataset, the results of the algorithms *NashOverlap* and *COPRA* are as shown in the figures [7.1](#), [7.15](#). The number of observed collaboration groups is greater than that of *COPRA* and the hierarchy is observed in the case of *NashOverlap*. Thus, we get more information using our algorithm, in the case of DBLP collaboration network.

<i>Component Significance</i>	<i>sizes of all the components detected by NashOverlap</i>	<i>sizes of all the components detected by COPRA</i>
Giant component	1683	1652
Databases	976, 408, 239, 152, 98, <i>48, 40</i>	811
Multimedia	222, 219, 67, <i>58</i>	279, 123
Computation Geometry	133, <i>97</i>	103
Computer Networks	123, 94	79
Computer Vision	67, 45	77
Machine Learning	976, 44	398
Computer aided Verification	57, <i>42</i>	53
KR and AI	976, <i>40</i>	58

Table 7.1: Considering only the component sizes (≥ 40), the above table compares the components detected by the algorithms *NashOverlap* and *COPRA*. Keeping aside the giant component, *NashOverlap* detects several components of databases and some hierarchical components in that area. The sizes of the detected components in the lower hierarchical level are italicised. The sizes of the components detected for each of the areas by *NashOverlap* are larger than that of *COPRA*

7.3 Summary

In this chapter, we explored the DBLP Collaboration Network, using our algorithm and identified the largest collaboration groups, of each field. The largest collabo-

ration group detected at both the threshold values for both the networks, has the average path length between 4 and 5 and the diameter between 9 and 11. The size of the largest collaboration group or the giant component detected by our algorithm is around one-third the size of the network for both the networks considered. The second largest collaboration group has size which is much smaller than the largest collaboration group, and is one-fourth the size of the largest collaboration group detected. The overlapping membership and community size membership distributions of the detected community structure follows power law as can be seen from the figures [7.11](#) and [7.12](#). The possibility of a node put in more than one community increases with decrease in the threshold.

Chapter 8

Conclusion

In the thesis, we have examined the problem of community detection in interaction networks using game theory. Our model is under the assumption that the similarity of any pair of nodes increases with the number of connections that the pair makes. Also, the more the similar the pair of nodes are, the more the number of common friends that the pair makes eventually.

We proposed two simple and scalable algorithms that inputs the interaction network and outputs its community structure, and the overlapping nodes present in the network. We proposed the algorithms for each of the cases of detecting the disjoint communities and the overlapping communities in a given network.

In this thesis, we proposed a disjoint community detection algorithm, *NashDisjoint* which works as good as the state of the art algorithms on both GN Benchmarks and LFR Benchmarks for the mixing factors less than 0.55. We have observed that the modularity values for the community structure detected by our algorithm on real social networks is in comparison to that of the Louvain algorithm, one of the best modularity optimization algorithms so far. We proposed

an overlapping community detection algorithm *NashOverlap* which works better than the state of the art algorithms on LFR benchmarks by a large margin, for the mixing factors from 0.1 to 0.5 and overlapping membership varying from 2 to 8 and when the percentage of overlapping nodes is 10%. This algorithm also works well on the networks with disjoint community structure and can detect the existence of disjoint communities.

Our algorithms can be very much parallelized, as in the first game, all the subgames are independent of each other. We performed experiments on DBLP co-authorship network and identified the top collaboration groups of the network, ordered in terms of their size. We observed that the largest collaboration group has people from almost all the computer science fields and the largest degree researchers are from the theoretical computer science, Machine Learning and Data mining. The largest collaboration group has size around 40% of the actual network size. Using these results, we want to show that our algorithm can detect meaningful information from collaboration networks.

One important thing to note in the context of community detection algorithms is that there is no community detection algorithm so far that can be considered best for all kinds of networks. This field lacks a theoretical framework which gives the definition of a community and the definition of the best measure that the entire research community agrees upon. Until there is such a framework, there is no other way to compare the performance of any two community detection algorithms other than to evaluate their performance on the networks whose groundtruth is already known. Also, there are no scalable game theoretic algorithms that can detect communities in dynamic networks.

Bibliography

- [1] *dblp computer science bibliography*. 2014. [Online; accessed 20-Sep-2014].
- [2] Balzs Adamcsek, Gergely Palla, Ills J. Farkas, Imre Dernyi, and Tams Vicsek. CFinder: locating cliques and overlapping modules in biological networks. *Bioinformatics (Oxford, England)*, 22(8):1021–1023, April 2006.
- [3] Gaurav Agarwal and David Kempe. Modularity-maximizing graph communities via mathematical programming. *The European Physical Journal B-Condensed Matter and Complex Systems*, 66(3):409–418, 2008.
- [4] Yong-Yeol Ahn, James P Bagrow, and Sune Lehmann. Link communities reveal multiscale complexity in networks. *Nature*, 466(7307):761–764, 2010.
- [5] Nelson A Alves. Unveiling community structures in weighted networks. *Physical Review E*, 76(3):036101, 2007.
- [6] Earl R. Barnes. An algorithm for partitioning the nodes of a graph. *SIAM Journal on Algebraic Discrete Methods*, 3(4):541–550, December 1982.
- [7] Jeffrey Baumes, Mark K Goldberg, Mukkai S Krishnamoorthy, Malik Magdon-Ismail, and Nathan Preston. Finding communities by clustering a graph into overlapping subgraphs. *IADIS AC*, 5:97–104, 2005.

- [8] Xiaohui Bei, Ning Chen, Liyu Dou, Xiangru Huang, and Ruixin Qiang. Trial and error in influential social networks. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1016–1024. ACM, 2013.
- [9] Jonathan W Berry, Bruce Hendrickson, Randall A LaViolette, Vitus J Leung, and Cynthia A Phillips. Community detection via facility location. *arXiv preprint arXiv:0710.3800*, 2007.
- [10] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, October 2008.
- [11] Andrea Capocci, Vito DP Servedio, Guido Caldarelli, and Francesca Colaiori. Detecting communities in large networks. *Physica A: Statistical Mechanics and its Applications*, 352(2):669–676, 2005.
- [12] J. Chen and B. Yuan. Detecting functional modules in the yeast protein-protein interaction network. *Bioinformatics*, 22(18):2283–2290, September 2006.
- [13] Wei Chen, Zhenming Liu, Xiaorui Sun, and Yajun Wang. A game-theoretic framework to identify overlapping communities in social networks. *Data Mining and Knowledge Discovery*, 21(2):224–240, 2010.
- [14] Zengqiang Chen and Zheng Xie. Community detection based on local topological information in power grid. In *Neural Networks (IJCNN), 2014 International Joint Conference on*, pages 907–912. IEEE, 2014.

- [15] Aaron Clauset, M. Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical Review E*, 70(6), December 2004.
- [16] Anne Condon and Richard M. Karp. Algorithms for graph partitioning on the planted partition model. *Random Structures and Algorithms*, 18(2):116–140, March 2001.
- [17] Leon Danon, Albert Daz-Guilera, and Alex Arenas. The effect of size heterogeneity on community identification in complex networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2006(11):P11010–P11010, November 2006.
- [18] Leon Danon, Albert Daz-Guilera, Jordi Duch, and Alex Arenas. Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(09):P09008–P09008, September 2005.
- [19] W. E. Donath and A. J. Hoffman. Lower bounds for the partitioning of graphs. *IBM Journal of Research and Development*, 17(5):420–425, September 1973.
- [20] Luca Donetti and Miguel A Muoz. Detecting network communities: a new systematic and efficient algorithm. *Journal of Statistical Mechanics: Theory and Experiment*, 2004(10):P10012, October 2004.
- [21] Luca Donetti and Miguel A. Muoz. Improved spectral algorithm for the detection of network communities. In *In Modeling cooperative behavior in the social sciences*, pages 104–107, 2005.
- [22] Stijn Van Dongen. PhD thesis, Dutch National Research Institute for Mathematics and Computer Science, University of Utrecht, Netherlands, 2000.

- [23] Stijn Van Dongen. Performance criteria for graph clustering and markov cluster experiments. Technical report, National Research Institute for Mathematics and Computer Science in the University of Utrecht, Netherlands, 2000.
- [24] Haifeng Du, Marcus W. Feldman, Shuzhuo Li, and Xiaoyi Jin. An algorithm for detecting community structure of social networks based on prior knowledge and modularity. *Complexity*, 12(3):53–60, January 2007.
- [25] Jordi Duch and Alex Arenas. Community detection in complex networks using extremal optimization. *Physical Review E*, 72(2), August 2005.
- [26] Alcides Viamontes Esquivel and Martin Rosvall. Compression of flow can reveal overlapping-module organization in networks. *Physical Review X*, 1(2):021025, 2011.
- [27] Illés Farkas, Dániel Ábel, Gergely Palla, and Tamás Vicsek. Weighted network modules. *New Journal of Physics*, 9(6):180, 2007.
- [28] Gary William Flake, Steve Lawrence, and C. Lee Giles. Efficient identification of web communities. pages 150–160. ACM Press, 2000.
- [29] L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8(0):399–404, January 1956.
- [30] Santo Fortunato and Marc Barthélemy. Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, 104(1):36–41, 2007.
- [31] Marco Gaertler and Ulrik Brandes Dorothea Wagner. Experiments on graph clustering algorithms. In *Algorithms - ESA 2003*, number 2832, pages 568–579. Springer Berlin Heidelberg, September 2003.

- [32] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, June 2002.
- [33] Prem K Gopalan and David M Blei. Efficient discovery of overlapping communities in massive networks. *Proceedings of the National Academy of Sciences*, 110(36):14534–14539, 2013.
- [34] Mark S. Granovetter. The strength of weak ties. *American Journal of Sociology*, 78(6):pp. 1360–1380, 1973.
- [35] Steve Gregory. An algorithm to find overlapping community structure in networks. In JoostN. Kok, Jacek Koronacki, Ramon Lopez de Mantaras, Stan Matwin, Dunja Mladeni, and Andrzej Skowron, editors, *Knowledge Discovery in Databases: PKDD 2007*, volume 4702 of *Lecture Notes in Computer Science*, pages 91–102. Springer Berlin Heidelberg, 2007.
- [36] Steve Gregory. Finding overlapping communities in networks by label propagation. *New Journal of Physics*, 12(10):103018, October 2010.
- [37] Steve Gregory. Fuzzy overlapping communities in networks. *CoRR*, abs/1010.1523, 2010.
- [38] R. Guimera, S. Mossa, A. Turtleschi, and L. A. N. Amaral. The world-wide air transportation network: Anomalous centrality, community structure, and cities’ global roles. *Proceedings of the National Academy of Sciences*, 102(22):7794–7799, May 2005.

- [39] Roger Guimer and Lus A. Nunes Amaral. Functional cartography of complex metabolic networks. *Nature*, 433(7028):895–900, February 2005.
- [40] Roger Guimer, Marta Sales-Pardo, and Lus Amaral. Modularity from fluctuations in random graphs and complex networks. *Physical Review E*, 70(2), August 2004.
- [41] Adel Hlaoui and Shengrui Wang. A direct approach to graph clustering. In *Neural Networks and Computational Intelligence'04*, pages 158–163, 2004.
- [42] Jierui Xie. *SLPA Manual*. 2011. [Online; accessed 20-Sep-2014].
- [43] Brian Karrer, Elizaveta Levina, and M. E. J. Newman. Robustness of community structure in networks. *Physical Review E*, 77(4):046119, 2008.
- [44] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, 49(2):291–307, February 1970.
- [45] Jussi M Kumpula, Mikko Kivelä, Kimmo Kaski, and Jari Saramäki. Sequential algorithm for fast clique percolation. *Physical Review E*, 78(2):026109, 2008.
- [46] Lancichinetti. *OSLOM Manual*. 2013. [Online; accessed 20-Sep-2014].
- [47] Andrea Lancichinetti and Santo Fortunato. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Physical Review E*, 80(1), July 2009.
- [48] Andrea Lancichinetti, Santo Fortunato, and Jnos Kertsz. Detecting the overlapping and hierarchical community structure in complex networks. *New Journal of Physics*, 11(3):033015, March 2009.

- [49] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. Benchmark graphs for testing community detection algorithms. *Physical Review E*, 78(4), October 2008.
- [50] Andrea Lancichinetti, Filippo Radicchi, Jos J. Ramasco, and Santo Fortunato. Finding statistically significant communities in networks. *PLoS ONE*, 6(4):e18961, April 2011.
- [51] Jure Leskovec, Kevin J Lang, Anirban Dasgupta, and Michael W Mahoney. Statistical properties of community structure in large social and information networks. In *Proceedings of the 17th international conference on World Wide Web*, pages 695–704. ACM, 2008.
- [52] Jure Leskovec, Kevin J Lang, Anirban Dasgupta, and Michael W Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6(1):29–123, 2009.
- [53] Jure Leskovec and Julian J Mcauley. Learning to discover social circles in ego networks. In *Advances in neural information processing systems*, pages 539–547, 2012.
- [54] S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, March 1982.
- [55] Rodica Ioana Lung, Camelia Chira, and Anca Andreica. Game theory and extremal optimization for community detection in complex dynamic networks. *PloS one*, 9(2):e86891, 2014.

- [56] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman, editors, *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.
- [57] Dov Monderer and Lloyd S Shapley. Potential games. *Games and economic behavior*, 14(1):124–143, 1996.
- [58] Ramasuri Narayanam and Yadati Narahari. A game theory inspired, decentralized, local information based algorithm for community detection in social graphs. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 1072–1075. IEEE, 2012.
- [59] M. Newman. Analysis of weighted networks. *Physical Review E*, 70(5), November 2004.
- [60] M. Newman. Fast algorithm for detecting community structure in networks. *Physical Review E*, 69(6), June 2004.
- [61] M. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(2), February 2004.
- [62] M. E. J. Newman. The structure of scientific collaboration networks. *Proceedings of the National Academy of Sciences*, 98(2):404–409, 2001.
- [63] M. E. J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, June 2006.

- [64] M. E. J. Newman and Elizabeth A Leicht. Mixture models and exploratory analysis in networks. *Proceedings of the National Academy of Sciences*, 104(23):9564–9569, 2007.
- [65] M.E. J. Newman. A measure of betweenness centrality based on random walks. *Social Networks*, 27(1):39–54, January 2005.
- [66] Gergely Palla, Imre Derenyi, Ills Farkas, and Tams Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, June 2005.
- [67] John W Pinney and David R Westhead. Betweenness-based decomposition methods for social and biological networks. *Interdisciplinary Statistics and Bioinformatics*, pages 87–90, 2006.
- [68] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi. Defining and identifying communities in networks. *Proceedings of the National Academy of Sciences*, 101(9):2658–2663, March 2004.
- [69] Usha Raghavan, Rka Albert, and Soundar Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*, 76(3), September 2007.
- [70] Matthew J. Rattigan, Marc Maier, and David Jensen. Using structure indices for efficient approximation of network properties. page 357. ACM Press, 2006.
- [71] Matthew J. Rattigan, Marc E. Maier, and David Jensen. Graph clustering with network structure indices. In *International Conference on Machine Learning*, pages 783–790, 2007.

- [72] Wei Ren, Guiying Yan, Xiaoping Liao, and Lan Xiao. Simple probabilistic algorithm for detecting community structure. *Physical Review E*, 79(3):036111, 2009.
- [73] M. Rosvall and C. T. Bergstrom. An information-theoretic framework for resolving community structure in complex networks. *Proceedings of the National Academy of Sciences*, 104(18):7327–7331, May 2007.
- [74] M. Rosvall and C. T. Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4):1118–1123, January 2008.
- [75] Martin Rosvall and Carl T Bergstrom. Multilevel compression of random walks on networks reveals hierarchical organization in large integrated systems. *PloS one*, 6(4):e18209, 2011.
- [76] Philipp Schuetz and Amedeo Caffisch. Efficient modularity optimization by multistep greedy algorithm and vertex mover refinement. *Physical Review E*, 77(4):046112, 2008.
- [77] Y. Sun, B. Danila, K. Josi, and K. E. Bassler. Improved community structure detection using a modified fine-tuning strategy. *EPL (Europhysics Letters)*, 86(2):28004, April 2009.
- [78] Yizhou Sun, Jie Tang, Jiawei Han, Manish Gupta, and Bo Zhao. Community evolution detection in dynamic heterogeneous information networks. In *Proceedings of the Eighth Workshop on Mining and Learning with Graphs*, pages 137–146. ACM, 2010.

- [79] Joshua R Tyler, Dennis M Wilkinson, and Bernardo A Huberman. E-mail as spectroscopy: Automated discovery of community structure within organizations. *The Information Society*, 21(2):143–153, 2005.
- [80] Ken Wakita and Toshiyuki Tsurumi. Finding community structure in mega-scale social networks:[extended abstract]. In *Proceedings of the 16th international conference on World Wide Web*, pages 1275–1276. ACM, 2007.
- [81] Lilian Weng, Filippo Menczer, and Yong-Yeol Ahn. Virality prediction and community structure in social networks. *Scientific reports*, 3, 2013.
- [82] Dennis M Wilkinson and Bernardo A Huberman. A method for finding communities of related genes. *proceedings of the national Academy of sciences*, 101(suppl 1):5241–5248, 2004.
- [83] Biao Xiang, En-Hong Chen, and Tao Zhou. Finding community structure based on subgraph similarity. In *Complex Networks*, page 7381. Springer Berlin Heidelberg, 2009.
- [84] Jierui Xie, Stephen Kelley, and Boleslaw K. Szymanski. Overlapping community detection in networks: The state-of-the-art and comparative study. *ACM Computing Surveys*, 45(4):1–35, August 2013.
- [85] Jierui Xie, Boleslaw K Szymanski, and Xiaoming Liu. Slpa: Uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process. In *Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on*, pages 344–349. IEEE, 2011.

- [86] Bo Yang and Jiming Liu. Discovering global network communities based on local centralities. *ACM Transactions on the Web (TWEB)*, 2(1):9, 2008.
- [87] Jaewon Yang and Jure Leskovec. Community-affiliation graph model for overlapping network community detection. In *Data Mining (ICDM), 2012 IEEE 12th International Conference on*, pages 1170–1175. IEEE, 2012.
- [88] Jaewon Yang and Jure 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.
- [89] Haijun Zhou. Distance, dissimilarity index, and network community structure. *Physical Review E*, 67(6), June 2003.