SCALABLE AND ROBUST ALGORITHMS FOR MINING
CLUSTERS IN GRAPHS

A Dissertation in
Industrial Engineering
by
Supreet Reddy Mandala

© 2013 Supreet Reddy Mandala

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

May 2013
The dissertation of Supreet Reddy Mandala was reviewed and approved* by the following:

Soundar Kumara  
Professor of Industrial Engineering  
Dissertation Advisor, Chair of Committee

Tao Yao  
Professor of Industrial Engineering

Ravi Ravindran  
Professor of Industrial Engineering

Kalyan Chatterjee  
Professor of Economics

Paul Griffin  
Professor of Industrial Engineering  
Department Head

*Signatures are on file in the Graduate School.
Several social and technological systems around us can be modeled as a network or a graph. The topology of such networks is known to play a major role in understanding and predicting the system behavior. A topological feature which has been observed across several domains is the formation of clusters. A cluster or module or community is a subset of vertices which have high density of connections between cluster members and few connections leaving the cluster. The knowledge of clusters is extremely useful in a variety of domains - product recommendations in social networks, characterizing functions on unknown proteins in protein interaction networks, product placement and promotions in retail networks. In this thesis, we focus four problems addressing different aspects of graph clustering. Firstly, we propose an algorithm which can detect disjoint clusters in graphs. We optimize an existing quality function, called modularity, which quantifies the quality of a given partition. Therefore, detecting clusters is equivalent to finding a partition which maximizes modularity. Due to the combinatorial nature of the modularity maximization it is hard to find optimal solutions. We propose an ant colony optimization based heuristic which can detect near optimal partitions in a reasonable amount of time. We validate the effectiveness of our algorithm on various social network data sets. Our second problem addresses the issue of detecting robust graph partitions. The motivation behind this problem is to detect partitions which not only maximize modularity but are also immune perturbations in network topology. In several real world setting, perturbations in network topology can be expected due to noisy experimental data. We develop a robust optimization based max-min optimization model in order to define a robust graph partition. Furthermore, we propose an algorithm which can detect a robust partition while maintaining scalability. The usefulness of robust partitions in illustrated on both real and artificial networks. The third problem deals with detecting multiple diverse alternative partitions with high modularity scores. It
has already been reported that several dissimilar partitions may exist with quite similar modularity scores. We propose an approach to detect alternative graph partitions by modifying the network topology forcing clustering algorithm to detect diverse partitions. We confirm the existence of diverse partitions in several real world networks. A case study is presented for a retail graph (book co-purchase network) at amazon.com illustrating the utility of our approach. As a part of the fourth problem, we propose a first principles definition of a cluster which does not have any limitations of modularity based approaches. Moreover, our approach is able to detect overlapping clusters. We show that our definition of a cluster has a direct correspondence with the nash equilibrium of a non-cooperative game. We define a clustering game and show the existence of a nash equilibrium. Both parallel and non-parallel algorithms are proposed which have a near linear time running time. We compare our approach to existing overlapping cluster detection techniques when tested on artificial networks. Also, potential applications are shown in natural language processing (word sense disambiguation) and WWW segmentation problems.
# Table of Contents

List of Figures viii
List of Tables xii
Acknowledgments xiii

Chapter 1
Introduction 1
1.1 Challenges in Graph Clustering ............................... 2
1.2 Research Objectives and Contributions ........................ 4
1.2.1 Detecting Graph Partitions using Ant Colony Optimization . 4
1.2.2 Detecting Robust Graph Partitions ............................ 5
1.2.3 Detecting Alternative Graph Clusters ........................ 5
1.2.4 A Game-theoretic approach to Graph Clustering ............... 6
1.3 Organization of Thesis ........................................... 6

Chapter 2
Detecting Graph Partitions using Ant Colony Optimization 7
2.1 Introduction ..................................................... 7
2.2 Existing Work .................................................. 10
2.3 Methodology ................................................... 14
2.3.1 Overview of ACO ........................................... 15
2.3.2 Solution Construction ...................................... 16
2.3.3 Local Search ............................................... 17
2.3.4 Pheromone Update ......................................... 17
2.3.5 Convergence ............................................... 19
2.4 Results ......................................................... 20
5.7 Experimental Results ........................................ 86
  5.7.1 Generating Artificial Networks .......................... 87
  5.7.2 Parameter Selection ..................................... 87
  5.7.3 Performance on Artificial Networks .................... 89
5.8 Applications .................................................. 90
  5.8.1 Word Sense Disambiguation ............................. 90
  5.8.2 Segmenting the World Wide Web ....................... 92
5.9 Conclusion ..................................................... 92

Chapter 6
  Future Work .................................................. 94

Bibliography ..................................................... 96
List of Figures

1.1 Illustration of two networks of equal size (16 nodes and 28 edges) with and without modular topology ........................................ 2
1.2 Overlapping Clusters .......................................................... 3
1.3 Hierarchical Clusters .......................................................... 3

2.1 A hypothetical network with communities/modules. ............... 8
2.2 An example illustrating our methodology. Each split requires us to solve a network bisection problem using our ACO metaheuristic. Note that a split is carried out only if it results in increase in modularity score. .................................................. 15
2.3 Pseudo code for our ACO heuristic. The parameters used are: $n_a$ – number of ants, $T$ – maximum number of iterations, $s^{kt}$ – solution constructed by ant $k$ during iteration $t$, $s_{bt}$ – best solution among all ants in iteration $t$, $s_{br}$ – best solution since last reset and $s_{bs}$ – best solution so far .................................................. 16
2.4 Pseudo code for the local search ............................................ 18
2.5 The performance of ACO heuristic on a synthetic network. (a) A sample graph with $f = 0.05$ (b) The solid line indicates the expected modularity whereas the error bars indicate the modularity achieved. (c) The VI of detected communities w.r.t true communities is shown. Only 10% of the nodes are misclassified even when $f$ is increased to 0.35 (d) Test for the stochastic nature of the algorithm. Both the axis represents the node numbers. The black (white) region corresponds to the nodes which have never (always) been grouped together .................................................. 22
2.6 The partition corresponding to maximum modularity obtained for Zachary’s karate club. The actual split had occurred due to a dispute between node 1 and node 34. ............................................ 23
2.7 The best partition obtained for jazz music bands. In both figures, the solid line represents main partition and the dashed lines represent sub-partitions. Note that the algorithm does not explicitly identify main partitions and sub-partitions, authors used subjective evaluation to identify such partitions. Network layouts were generated using Pajek. 24

2.8 The distribution of community assignments inside each of the eight houses 25

3.1 A toy network illustrates the importance of detecting robust clusters. There are two clusterings shown - by color and by shape. The standard modularity maximization detects two clusters indicated by the shapes (circles and squares). The proposed robust modularity maximization approach detects five clusters indicated by the colors. 30

3.2 Modularity plots depicting the concept of robust partition. Variation of modularity across graph perturbations (dotted lines). The solid line refers to the original graph which peaks at partition-5. The minimum modularity plot across all perturbations. The robust partition (partition-10) maximizes the minimum modularity. 32

3.3 Accuracy of robust partitions as compared to non-robust partitions at varying noise levels. Each data point is average over 10 independent replications. 40

3.4 The ability of robust partitions to detect clusters at higher levels of resolution as compared to non-robust partitions. The adjacency matrix is constructed by setting $n = 3$ and $p = 10$. 41

3.5 Football Network consisting of 179 vertices and 759 edges. All 11 conferences are annotated (and color coded) along with the 4 independent teams in the middle. 43

3.6 Non-Robust Partition of Football Network ($Q = 0.60$ and $VI = 0.357$) 44

3.7 Robust Partition of Football Network ($Q = 0.548$ and $VI = 0.052$). 44

4.1 Example of a near-optimal clustering obtained using our methodology in a toy network. 47

4.2 Figures illustrating the family of perturbed graphs when $v = \{0, 0.5, 1\}$. Width of the edge represents the strength of connection. A perturbed graph is constructed by (1) adding a dummy node and (2) adding new edges between the dummy and all non-dummy nodes. The weight on any edge at perturbation level, $v$, can be computed from Eq(4.6). 50
4.3 All alternative clusterings detected in the sample network. Each cluster is marked by a different color. .......................................................... 58
4.4 Analysis of perturbed clusters in artificially generated networks under various levels of noise ($q$). For each noise level, alternative clusterings ($c_v$) are detected when $v \in \{0, 0.01, \ldots, 1\}$. A pair of similar (dissimilar) clusterings will result in a dark(light) spot in the distance matrix. ................................................................. 64
4.5 Analysis of alternative clusters in social networks. The alternative clusterings ($c_v$) are detected when $v \in \{0, 0.01, \ldots, 1\}$. A pair of similar (dissimilar) clusterings will result in a dark(light) spot in the distance matrix. ................................................................. 65
4.6 Analysis of alternative clusters in biological networks. The alternative clusterings ($c_v$) are detected when $v \in \{0, 0.01, \ldots, 1\}$. A pair of similar (dissimilar) clusterings will result in a dark(light) spot in the distance matrix. ................................................................. 65
4.7 Analysis of alternative clusters in technological networks. The alternative clusterings ($c_v$) are detected when $v \in \{0, 0.01, \ldots, 1\}$. A pair of similar (dissimilar) clusterings will result in a dark(light) spot in the distance matrix. ................................................................. 66
4.8 Alternative clusters in political book network. The shape indicates the political inclination of the book (circle-conservative, square-liberal, triangle-neutral). ................................................................. 66
5.1 (a) Cycling of states in simultaneous best response in a two node network (b) The states and their transition probabilities as a markov chain. ................................................................. 80
5.2 The number of best responses required by both algorithms depend nearly linearly on the number of vertices. The algorithms are run with parameters $(\rho, \lambda, \epsilon) = (0.1, 2, 0.99)$. ................................................................. 84
5.3 The effect of parameters $(\rho, \lambda, \epsilon)$ on the computational complexity and the accuracy of SEBR and SMBR. The networks are generated with parameters $(N = 100, k_{avg} = 10, k_{max} = 20, s_{min} = 3, s_{max} = 20, on = 10, om = 2)$ under five different scenarios of varying noise levels ($\mu$). Each data point is obtained by averaging over 10 random graph realizations. ................................................................. 88
5.4 Comparing the accuracy of SEBR, SMBR and Clique Percolation Method (CPM). The parameters of SEBR/SMBR and CPM are set as $(\rho = 0.05, \lambda = 2, \epsilon = 0.99)$ and $(k = 4)$ respectively. Each data point is obtained by averaging over 10 random graph realizations. ................................................................. 90
5.5 Clusters around the words PLAY and COLD in the word association network. An edge is placed between two words if their total frequency of occurrence is at least 0.025. Parameters used \((\rho, \lambda, c) = (0.1, 2, 2)\) . . . . . . . . . . . . . . . . . . . . . . . . . . . . 91

5.6 This is segmentation of the web pages in Google’s domain. The parameter settings are \((\rho, \lambda, c) = (0.5, 2, 1)\) . . . . . . . . . . . . . . 92
List of Tables

2.1 Comparison of modularity values achieved and the number of communities found across various algorithms (EO - Extremal Optimization, NM - Newman’s Spectral Partitioning) and test networks. The entries are the modularity values achieved by each of the algorithms (and respective communities found). ........................................ 26

2.2 Time taken by our ACO heuristic on various problem sizes. The computer program is written in Java and the computational experiments were run on a standard desktop windows pc with pentium 4 processor and memory of 1 Gb. ........................................ 27

4.1 Detailed list of books contained in each of the additional clusters detected ........................................ 63
Acknowledgments

I am greatly indebted to many people who have made my stay at Penn State an enriching experience. I thank my advisor Dr. Soundar Kumara for his constant support and encouragement. I am also grateful to my committee members Dr. Tao Yao, Dr. Kalyan Chatterjee and Dr. Ravindran Arunachalam for their support and time. I have gained much from their thoughtful suggestions on my research work.

I would also like to thank all the researchers in LISQ lab. I feel lucky to have worked by the side of such wonderful people, and the lessons learnt will stick with me for life. I would like to thank Abhinav, Ajay, Ashwini, BG, Chetan, Mahesh, Mouni, Phani Kiran, Prahalad, Phani Teja, Sharat, Sridhar, Uday and Venkat for making my life at Penn State enjoyable.

On a personal note, I will always be indebted my parents Sudheer and Sunitha for their support. Their simplicity and humility has played a big part in shaping my life. I would also like to thank my grandparents, my sister Sushmita, my brother-in-law Swadeep and my little niece Dhiya who have showered me with love and affection. I also greatly enjoyed the enthusiasm and support shown by my parents-in-law Muralidhar and Lakshmi. Last but not the least I would like to thank my loving, supportive, dedicated, and always encouraging wife Manini.
Dedication

To my parents and wife
Chapter 1

Introduction

The understanding of several social, biological and technological systems, both naturally occurring and engineered, will play a major role in the progress of our society. Examples range from neural systems, global economic systems to the Internet. A major difficulty in understanding such systems is due to the complexity induced by the large number of components which interact with each other in a highly non-trivial fashion. We refer to such systems as complex systems. A decade ago, there were a series of discoveries which pointed to the fact that the topology of the interconnected components of the complex systems plays a major role in determining their behavior. The most natural way of abstracting the interconnected components of a Complex System is a Graph (or Network) where the components are modeled as nodes and their interactions as edges. These networks exhibited highly non-trivial topological features when compared to random graphs or regular graphs. One such topological feature is cluster formation or modular organization. A cluster (or community or module) can be informally defined as a set of nodes with large number of edges inside the cluster with fewer edges leaving the cluster. Figure 1.1 illustrates two toy networks with equal size but with different topologies - with (1.1(a)) and without (Figure 1.1(b)) underlying cluster structure. The interpretation of a cluster may vary depending on the domain being modeled. In the case of biological systems, clusters could correspond to functional units. For example, in a Protein Interaction Network, a module corresponds to a protein complex performing a particular biological function. One can predict functions performed by previously unknown proteins using the identified clusters. In a social network,
a cluster corresponds to a set of people who share similar views/interests. The knowledge of social clusters can play a vital role in product marketing decisions. In the case of retail domain, clusters observed in a product co-purchase graph can indicate inherent associations between products. These hidden associations are crucial in pricing, location and promotional decisions. Due to a wide variety of applications, detecting clusters in graphs will play a major role in understanding and predicting the behavior of complex systems. Therefore, our focus in this thesis is to develop automated algorithms which can detect clusters in graphs.

1.1 Challenges in Graph Clustering

The seemingly simple problem of cluster detection has a lot of potential pitfalls. We discuss some of the major challenges in this section.

1. Overlapping Clusters

Typically, clusters are disjoint groups of vertices. However, there are several practical situations where clusters may overlap (see Figure 1.2). In the case of social networks, people may belong to multiple circles - professional, family, so on. In the case of biological networks, a biological component (protein, gene or metabolite) can be involved in performing several bodily functions. Therefore, it becomes important to discover overlapping vertices. The complexity of cluster detection, however, increases dramatically if a vertex can belong to multiple clusters.
2. Hierarchical Clusters

Another commonly observed property is that the clusters are hierarchically arranged. In other words, smaller clusters are nested in larger clusters. A classic example are the neuronal networks - each cluster corresponds to a particular brain function which are in turn organized hierarchically so that different functions can be coordinated. Similarly in case of social networks, people are often organized according to the specificity of their interests. Ex: All sports fans can be further organized into football fans, baseball fans, soccer fans and so on.

3. Scalability

With the amount of digital content generated by users, it is important that
mining algorithms maintain scalability. In fact, it is not even sufficient for algorithms to maintain polynomial complexity in the size of the network. Algorithms have to scale linearly or nearly linearly in order to process excessively large amount of information.

4. Stability
Any data collection technique involves false positives and false negatives i.e. existing edges not recorded or edges which do not exist. In this regard, one has to make sure that cluster detection algorithm is not sensitive to the input graph - minor perturbations to network topology should not result in drastically different clusters.

1.2 Research Objectives and Contributions
This thesis addresses four sub problems related to graph clustering. The specific research objectives, methodology and contributions for each of the sub problems are discussed in the following subsections.

1.2.1 Detecting Graph Partitions using Ant Colony Optimization
Several e-marketing applications rely on the ability to understand the structure of social networks. Social networks can be represented as graphs with customers as nodes and their interactions as edges. Most real world social networks are known to contain extremely dense subgraphs (also called as communities) which often provide critical insights about the emergent properties of the social network. The communities, in most cases, correspond to the various segments in a social system. Such an observation led researchers to propose algorithms to detect communities in networks. A modularity measure representing the quality of a network division has been proposed which on maximization yields good partitions. The modularity maximization is a strongly NP-complete problem which renders mathematical programming based optimization intractable for large problem sizes. Many heuristics based on Simulated Annealing, Genetic Algorithms and Extremal Optimization have been used to maximize modularity but have lead to suboptimal solutions.
In this paper, we propose an Ant Colony Optimization (ACO) based approach to detect communities. To the best of our knowledge, this is the first application of ACO to community detection. We demonstrate that ACO based approach results in a significant improvement in modularity values as compared to existing heuristics in the literature. The reasons for this improvement when tested on real and synthetic data sets are discussed.

1.2.2 Detecting Robust Graph Partitions

The problem of partitioning graphs into clusters has gained a lot of attention in the past decade. Clusters can be described as a group of nodes with abnormally large number of edges between them as compared to edges falling outside the group. A quality function, ”modularity”, has been proposed to detect such clusters. Modularity, on maximization, is known to result in good partitions when tested on several real world networks. However, modularity maximization can sometimes produce false positives i.e partitions which maximize modularity but have no real underlying cluster. In this paper, we define a robust partition by extending the notion of modularity maximization. The main idea is to detect partitions which not only have high modularity values but are also immune to graph perturbations. We develop a robust optimization based max-min optimization model and propose a scalable algorithm to detect robust partitions. We show that by incorporating robustness to perturbations one can improve the accuracy of the clusters detected. We illustrate the effectiveness of our approach when tested on real and synthetic data sets.

1.2.3 Detecting Alternative Graph Clusters

The problem of graph clustering or community detection has enjoyed a lot of attention in complex networks literature. A quality function, modularity, quantifies the strength of clustering and on maximization yields sensible partitions. However, in most real world networks there are exponentially large number of near-optimal partitions with some being very different from each other. Therefore, picking an optimal clustering among the alternatives does not provide complete information about network topology. To tackle this problem, we propose a graph perturbation
scheme which can be used to identify an ensemble of near-optimal and diverse clusterings. We establish analytical properties of modularity function under the perturbation which ensures diversity. Our approach is algorithm-independent and therefore can leverage any of the existing modularity maximizing algorithms. We numerically show that our methodology can systematically identify very different partitions on several existing data sets. The knowledge of diverse partitions sheds more light into the topological organization and helps gain a more complete understanding of the underlying complex network.

1.2.4 A Game-theoretic approach to Graph Clustering

Last decade has witnessed an explosion in the modeling of complex systems. Predominantly, graphs are used to represent these systems. The problem of detecting overlapping clusters in graphs is of utmost importance. We present a novel first principles definition of overlapping clusters. A non-cooperative game is proposed such that the equilibrium conditions of the game correspond to the clusters in the graph. Several properties of the game are analyzed and exploited to show the existence of a pure Nash Equilibrium (NE) and compute it effectively. We present two algorithms to compute NE and prove their convergence. Empirically, the complexity of both algorithms are nearly linear in the number of edges. Also, one of the algorithms can be readily parallelized, making it scalable. Finally, our approach is compared to an existing overlapping cluster detection algorithm and validated on several artificial and real datasets.

1.3 Organization of Thesis

The four sub problems listed in the previous section are organized into four chapters (Chapters 2 - 5). Each chapter is self contained i.e the details (model, related literature and experimentation) are addressed separately. Finally, Chapter 6 discusses research extensions and future work.
Detected Graph Partitions using Ant Colony Optimization

2.1 Introduction

With the ubiquity of the internet, Information Technology (IT) has emerged as a key component for most organizations. Several business functions like marketing, operations and logistics are now closely intertwined with IT. Internet based marketing or e-marketing, in specific, has become very popular as the potential customer market has expanded from a local to global scale. Moreover, the digital traces left behind by customers can be extremely useful in designing market segmentation and product placement strategies. These digital traces can be obtained from social networking websites, online forums, e-mails, call records and so on which capture the interactions among the people. The interactions between people in a social systems can be conveniently represented as graphs with nodes denoting system components and edges capturing the interactions between components. The topology of these graphs help us gain significant insights regarding the functionality of the system. In the past decade, there has been a significant amount of work done in the field of networks in order to characterize the structure of networks. Although social network research existed for a long time, the recent surge of interest is due to the availability of data and the observation that the topology of networks play a major role in the determining behavior of the network
as a whole. The small world phenomenon [1], existence of power law degree distributions [2] and formation of motifs in biological networks [3] are some of the many observations which have demonstrated the importance of connectivity. One such significant result is that, most of the real world networks naturally form modules or communities. These modules are groups of nodes with abnormally large number of edges between them as opposed to spanning out of them (see Figure 2.1). The detection of such modules could be of utmost importance as they often correspond to a set of system components performing a similar function. This is especially true in the context of social networks where communities often correspond to people with common interests. Such a phenomenon is also referred to as homophily in sociology[4]. The knowledge of such communities can help significantly in various aspects of e-marketing like market segmentation, product placement, product recommendations and so on. A prime application of clustering is in collaborative filtering and has been used in various product recommendation systems such as Amazon recommendation system [5] and the Netflix movie recommendation system [6]. A common approach to recommend new products to a customer is to detect a cluster of customers who have similar preferences[7]. The preferences of the customers in the cluster are used to extrapolate the preferences of the customer to whom the recommendation is to be made. Also, applications of clustering in market segmentation has been explored in the context of an online auction site [8]. A network of bidders was constructed based on similar product interests. The network was later clustered and analyzed to detect groups with special interests. There are several other applications of clustering spanning areas such as biology, social sciences and finance [9].

There is an inherent ambiguity in the definition of a community - when should a subgraph be labelled as dense? Hence, researchers have proposed a variety of
definitions and corresponding algorithms to detect communities. Moreover, these communities can be overlapping as people may belong to different social groups. We, however, restrict our selves to detecting non-overlapping communities. Among the existing definitions, (arguably) the most widely used definition of community is based on modularity function [10]. The modularity function measures the quality or fitness of a community assignment with respect to a graph. A high modularity score indicates a strong community structure. Interestingly, it was found that known communities in many real world networks correspond closely to the community assignments which maximize the modularity function. Hence, community detection problem can be posed as modularity maximization which is a discrete optimization problem. The problem of maximizing modularity is shown to be strongly NP-Complete [11]. In other words, it is difficult to even approximate the optimal modularity partition in polynomial time. As a result, there are plethora of heuristics which try to maximize this measure using various techniques. However, it is shown by Fortunato and Barthelemy [12] that modularity maximization may not identify community structures whose size is below a certain resolution limit. Moreover, Guimera et al. [13] show that sparse random networks may also exhibit high modularity values. Such results indicate that modularity maximization, although a necessary condition, may not be a sufficient condition for existence of a community structure. In spite of such drawbacks, maximizing modularity seems to work surprisingly well on many real world networks [14].

Several computational algorithms have been suggested in the literature to solve the optimization problem. However, they all belong to a class of instance based search [15] i.e. future solutions evolve based only on the current solution. In contrast, we propose a methodology based on an ACO heuristic which belongs to a class of model based search techniques i.e. future solutions are generated based on a probabilistic model which is updated by the current solution. To the best of our knowledge, ACO has not yet been applied to community detection. We wish to at least partially overcome some drawbacks of instance based search techniques such as premature convergence [16] by using model based search. ACO is best described as a metaheuristic inspired by the food foraging behavior of ants. The modularity measure is maximized by recursively solving several unconstrained binary quadratic problems using ACO. We will see that the main advantage of us-
ing ACO based algorithm is that certain theoretical guarantees can be provided regarding convergence and solution quality. The remainder of the article is arranged as follows: in section 2.2 we introduce the existing work based on which the current model is constructed. In section 2.3, we present the details about the current approach followed by results and validation in section 2.4. The summary and conclusions are recorded in section 2.5.

2.2 Existing Work

This section deals with the existing literature in the field of community detection and ACO which will serve as the background for section 3. We will use the following notation to introduce the concept of modularity. Consider a binary undirected graph, $G = (V, E)$, where $V$ is the set of vertices s.t $|V| = n$ and $E$ is the set of edges s.t $|E| = m$. An adjacency matrix, $A_{n \times n}$, is a binary symmetric matrix whose entries $a_{ij}$ are non-zero if vertex $i$ and vertex $j$ are connected i.e. $a_{ij} = 1$ if $(i, j) \in E$. Let $k_i = \sum_{j=1}^{n} a_{ij}$, denote the degree of node $i$.

As we discussed before, the definition of a community remains ambiguous. Newman [10] defined a cluster by comparing it to a random graph, $G_{Null}$, with the same degree distribution as the original graph $G$. In order to understand the random graph, imagine a graph with isolated vertices, $V$. We then place an edge at random by choosing any two vertices according to some probability distribution. This is repeated $m$ times so that the graph now has $m$ edges. Suppose the vertices are chosen with probabilities proportional to their degrees i.e. probability of choosing vertex $i$, $p_i = \frac{k_i}{2m}$. Note that the same vertex can be chosen twice (allowing self edges) and also multiple edges can be placed between two vertices. As a result, the number of edges between two vertices $i$ and $j$ is given by a random variable $A_{ij} \sim Binomial(m, 2p_ip_j)$. It follows that the expected number of edges between two vertices $i, j$ in $G_{Null}$ is given by, $E[A_{ij}] = 2mp_ip_j = \frac{k_i k_j}{2m}$. This mechanism of constructing a random graph ensures that it has a total of $m$ edges and the expected degree of a vertex in $G_{Null}$ is equal to the actual degree of corresponding vertex in the original graph. The quality of a cluster is defined as the excess the number of observed edges inside a cluster relative to the expected number of edges as predicted by the random graph model. We can now define $b_{ij}$ as the excess
number of edges between $i$ and $j = (\text{actual number of edges between } i \text{ and } j, a_{ij}) - (\text{expected number of edges between } i \text{ and } j, \frac{k_i k_j}{2m})$. In other words, $b_{ij}$, defined in Eq.(2.1), is the contribution of vertex $i$ and $j$ to the modularity of $G$ if they belong the same community.

$$b_{ij} = a_{ij} - \frac{k_i k_j}{2m} \quad \text{(2.1)}$$

$Q_G(x)$ represents the quality of partitioning the network, $G$, according to partition vector $x$. The partition vector indicates the grouping of vertices i.e. $x(i) = x(j) \Rightarrow i$ and $j$ belong to the same community. It follows that the modularity measure, $Q_G$, for the graph $G$ is defined as shown in Eq.(2.2)

$$Q_G(x) = \frac{1}{2m} \sum_{i=1}^{n} \sum_{j=1}^{n} b_{ij} \delta(x_i, x_j) \quad \text{Where}$$

$$\delta(\ldots) - \text{Kronecker delta function}$$

$$x_i - \text{Partition index of vertex } i$$

As $Q_G(x)$ represents quality of partition $x$ w.r.t graph $G$, partitions with higher Q values are preferred. The modularity score of any partition ranges from $-0.5$ to $1$. Empirically, it has been found that any partition with a modularity score $\geq 0.3$ exhibits a strong modular structure[14]. Also, one can easily extend the definition of modularity to weighted graphs as the binary nature of the adjacency matrix, $[a_{ij}]$, is never utilized in the derivation of equation 2.2. Due to the combinatorial nature of the problem, maximizing modularity is strongly NP-Complete [11]. Therefore, even an approximation algorithm to estimate optimal communities is a far cry. Hence, several heuristics have been employed to obtain the best possible partition of the graph into communities by maximizing $Q$. Of course, there exists several non-modularity based definitions and algorithms for detecting communities (see [17, 9]) but we will focus on modularity based approaches only.

Most modularity maximizing algorithms for community detection fall into two categories: divisive and agglomerative. The divisive (agglomerative) methods try to divide (aggregate) the nodes of the network based on a dissimilarity (similarity) measure [17]. An agglomerative hierarchical clustering method [18] implements a greedy heuristic to maximize $Q$. The key idea is to cluster communities which result in maximum improvement in modularity. Clauset et al [19] improved the
running time of the greedy algorithm [18] by employing a smarter codification scheme. Many divisive spectral partitioning algorithms have also been proposed which directly optimize modularity measure. Newman’s spectral method [20] recursively splits the given network into two based on eigen vectors of the modularity matrix. The splitting vector is chosen as parallel as possible to the eigen vector corresponding to highest eigen value. A variant of the eigen vector algorithm has been employed [21] which proposes a k-way partition (as opposed to a 2-way partition) and reports improvement in the results. Although the core of most spectral methods are inspired by traditional graph partitioning techniques, several heuristics are often deployed to improve the performance. The modularity function is discrete nonlinear and multimodal, rendering traditional optimization procedures less effective for large scale problem instances. The computational techniques in the literature like Simulated Annealing (SA) [22], Extremal Optimization (EO) [23] and Genetic Algorithms (GA) [24, 25] have been used and indeed report good results [26]. Typically, the accuracy of such approaches is marred by its computational complexity. A common feature of GA and the other previously employed approaches is that they belong to a general class of search techniques called instance based search. One of problems of instance based searches is premature convergence to sub-optimal solutions [16]. For example, GA requires the current population to generate the next set of candidate solutions. This generation process is carried out by certain operators like selection, mutation and crossover. As the current pool of solutions directly generates the next pool of solutions, it is assumed that these operators ensure enough population diversity to produce constantly improving candidate solutions. However, this may not be true and finding a suitable operator for the given problem instance is difficult. Hence, there is always a possibility of a biased search leading to a suboptimal solution. Moreover, most of the existing heuristics to detect community structures fail to provide theoretical guarantees on convergence.

In the hope of alleviating such problems, we use a search technique belonging to a class of model based search. In such approaches, the current solution pool updates an underlying solution model which in turn generates the next set of candidate solutions. A typical example of such an approach is Ant Colony Optimization (ACO). All ACO heuristics fall under a general category of swarm intelligence
based techniques. We use the term swarm intelligence in the context of any algorithm which is inspired by the collective behavior of social insect colonies and other animal societies [27] to solve problems in a distributed fashion. The collective behavior of ants in order to optimally forage food attracted researchers to this field. Such an emergent behavior is a result of indirect communication achieved by use of chemicals produced by ants, called pheromone. The pheromone helps ants mark the path from the nest to food source creating a pheromone trail. Such a trail serves as a guide for other ants to reach the food source. In this way, cooperation is achieved among the ants resulting in the ant colony choosing the shortest path. Based on this observation, the first ACO algorithm called Ant System (AS) was originally published by Dorigo et al [28] in 1996. The AS consists of a set of agents and a set of solution components. Each of the agents probabilistically constructs a solution based on the artificial pheromone levels associated with each solution component. Then the pheromone values are updated based on the quality of the solutions obtained. This procedure is iterated until certain stopping condition, like maximum possible iterations or convergence of pheromone values, is reached. Several modifications and extensions to AS have been published in the past decade. The Ant Colony System [29], Max-Min Ant System [30], Ant System in Hyper Cube Framework [30], Rank Based AS [31] are some of the many variants. The variation lies mainly in the way the pheromone values are updated and additional local search procedures to improve the solutions. Many design parameters such as number of ants, choice of local search, amount of exploration and exploitation and when and how to update, are left to the designer. ACO has been successfully applied to many combinatorial problems. It is interesting to note that other metaheuristics such as SA, tabu search out perform ACO in some combinatorial problems like Travelling Salesman Problem, graph coloring problem whereas ACO is one of the best heuristics for Quadratic Assignment problem and Sequential Ordering Problem [32].

Clearly, the choice of a metaheuristic must depend on the type of problem instance. We will see that recursive binary partitioning of network transforms Equation (2.2) into an unconstrained binary quadratic optimization problem. Recent work suggests that Ant System (AS) implemented in Hyper Cube Framework (HCF) works well for such optimization problems [33]. The main advantage of HCF-AS [33]
is that it can ensure improvement in the average objective function after each pheromone update. Therefore, we use Ant System like heuristic in HCF for community detection hoping for similar results. ACO has been applied to traditional graph partitioning formulations whose objective is to find a prespecified number of balanced (nearly equal size) partitions with minimum edge cut [34]. It is important to note that community detection is very different from the traditional graph partitioning as we do not know the optimal number of partitions. Moreover, solutions of min cut formulations have been known to be less effective especially on social networks [20]. To the best of our knowledge, ACO has not yet been applied to community detection. The central idea behind the proposed methodology is that a set of artificial ants are chosen so that each of them partitions the graph into two. This partition is based on the artificial pheromone levels associated with each node. The best partition (highest change in modularity) among all the ant solutions is rewarded and the pheromone values are updated accordingly. The process is repeated until the pheromone values converge. A detailed description of the methodology follows in the next section.

2.3 Methodology

We employ a recursive 2-way partitioning approach to find communities i.e. iteratively partition the network into two. A toy example to illustrate the network splitting process is shown in Figure 2.2. Each graph bisection is carried out using our ACO metaheuristic. As we are partitioning the graph into two, the converged solution, $x^*$, of ACO is binary. As a result, we create two new sub graphs $G'$ and $G''$ with vertices $V'$ and $V''$ where $V' = \{ i \in V \mid x^*_i = 1 \}$ and $V'' = \{ i \in V \mid x^*_i = 0 \}$. The modularity matrix for the sub graph $G'$ is updated as follows

$$b'_{ij} = b_{ij} - \delta(i, j) \sum_{k \in V'} b_{ik} \quad \forall i \in V', j \in V'$$

This transformation retains the zero row and column sum property of modularity matrices for all the sub graphs. The importance of the zero row and column sum property is discussed in section 2.3.1. The ACO metaheuristic is again used to partition each of the sub graphs. This process is continued till further splitting
Figure 2.2. An example illustrating our methodology. Each split requires us to solve a network bisection problem using our ACO metaheuristic. Note that a split is carried out only if it results in increase in modularity score.

does not result in increase in modularity value. In other words, either $V' = \emptyset$ or $V'' = \emptyset$ is an indication that the current subgraph is in fact a community and needs no further division.

2.3.1 Overview of ACO

This section provides an overview of the ACO metaheuristic which will be used for network bisection. The partition vector, $x$, can be expressed as a binary vector where $x_i = 1$ if node $i$ belongs to group 1 and $x_i = 0$ if node $i$ belongs to group 2. In such a setting, we can show that $\delta(x_i, x_j) = 2x_ix_j - x_i - x_j + 1$, where $\delta$ is the Kronecker delta function. It follows from Equation (2.2) that maximizing modularity for a two way partitioning of graph $G$ can be formulated as an unconstrained binary quadratic problem (UBQP).

$$\text{Max } Q_G(x) = \frac{1}{m} \sum_{i=1}^{n} \sum_{j=1}^{n} b_{ij}x_ix_j \quad \text{where } x \in \{0,1\}^n \quad (2.4)$$

Note that we used the zero row and column sum property of modularity matrix to derive the above equation i.e. $\sum_{i=1}^{n} b_{ij} = \sum_{j=1}^{n} b_{ij} = 0$. In order to solve the above UBQP, we assign a set of pheromone values $\tau = (\tau_1, \ldots, \tau_n)$ to all the nodes. The
Initialize $t = 0; \tau_i = 0.5, n_a, T$

while $t < T$ do
  for $k = 1 \ldots n_a$ do
    $s^{kt} \leftarrow$ Construct Solution($\tau$) [Described in Section 2.3.2]
    $s^{kt} \leftarrow$ Local Search($s^{kt}$) [Described in Section 2.3.3]
  end for
  $\tau \leftarrow$ Pheromone Update($\tau, s_{bt}, s_{br}, s_{bs}$) [Described in Section 2.3.4]
  if Convergence then
    Reset $\tau$ and $s_{br}$ [Described in Section 2.3.5]
  end if
  $t \leftarrow t + 1$
end while

Figure 2.3. Pseudo code for our ACO heuristic. The parameters used are: $n_a$ – number of ants, $T$ – maximum number of iterations, $s^{kt}$ – solution constructed by ant $k$ during iteration $t$, $s_{bt}$ – best solution among all ants in iteration $t$, $s_{br}$ – best solution since last reset and $s_{bs}$ – best solution so far

pheromone on node $i$, $\tau_i$, is equal to the likelihood of an ant assigning node $i$ to group 1. During each iteration, each ant generates a solution by selecting each vertex to either belong to group 1 or group 2 based on the pheromone level on the vertex (see section 2.3.2). A local search is then used to improve the solution generated (see section 2.3.3). The pheromone vector, $\tau$, is updated such that more pheromone is deposited on solution components which result in higher solution quality (see section 2.3.4). This process continues till the pheromone vector, $\tau$, converges to a solution $x^*$ (see section 2.3.5). The high-level overview of the ACO heuristic is depicted in Figure 2.3. The following sections present the details of our ACO methodology which solves Equation (2.4).

2.3.2 Solution Construction

As our problem is unconstrained, this phase is relatively simple. In each iteration $t$, every ant $k$ constructs a potential solution, $s^{kt}$, using the pheromone deposit, $\tau$, as probability of assigning group 1 to a node. Therefore, the probability of constructing a solution $s^{kt}$ is given as

$$P(s^{kt}_i = 1) = \tau_i \text{ and } P(s^{kt}_i = 0) = 1 - \tau_i \quad \forall i \in \{1, \ldots, n\}$$ (2.5)
2.3.3 Local Search

The solution constructed by each ant \( k \) is further improved by 1-opt local search procedure [35]. This local search improves the solution by recursively performing a descent search within one hamming neighborhood of \( s \) where \( s' \) is a 1-hamming neighbor of \( s \) if they differ exactly in one component. The working of this search procedure can be described by defining gain, \( g_i(s) \), which is the increase in the objective function value by flipping \( i^{th} \) component of the solution \( s \) (see Eq.(2.6)).

\[
g_i(s) = b_{ii}(1 - 2s_i) + 2 \sum_{j=1, j\neq i}^{n} b_{ji}s_j (1 - 2s_i) \tag{2.6}
\]

We greedily flip the component which produces largest gain to obtain a new solution. We recompute the gains for the new solution and continue this process till all gains are non-positive. Note that recomputing gains for each node requires \( n \) summation operations which is computationally very expensive. Therefore, we make use of an observation which will enable faster computation of gain vectors. Let \( g(s^{new, i}) \), be the gain vector of new solution, \( s^{new, i} \), obtained by flipping the \( i^{th} \) component of solution \( s \). The new gain vector is related to the old gain vector as shown in Eq.(2.7).

\[
g_j(s^{new, i}) = \begin{cases} 
-g_i(s) & j = i \\
g_j(s) + 2b_{ji}(1 - 2s_j)(2s_i - 1) & j \neq i
\end{cases} \tag{2.7}
\]

The summary of local search implementation is shown in Figure 2.4. Such a search procedure is equivalent to the famous Kernighan-Lin algorithm [36] used extensively in graph partitioning.

2.3.4 Pheromone Update

The solution construction and their improvement estimate the best possible solution based on the current pheromone values. This phase proposes an update scheme which will increase the pheromone deposits on solution components which have lead to higher quality solutions. Intuitively, the best solution obtained from the start of the algorithm, \( s_{bs} \), should guide the evolution of \( \tau \). However, such an
Input: $s$
Calculate $g_j(s)$ using Eq.(2.6)

\[ \textbf{while} \ \max_j g_j(s) > 0 \ \textbf{do} \]

\[ i = \arg \max_{j=1,...,n} (g_j(s)) \]
\[ s_i = 1 - s_i \]

Update gains using Eq.(2.7)

\[ \textbf{end while} \]

\[ \textbf{return} \ s \]

**Figure 2.4.** Pseudo code for the local search

update mechanism can easily lead the search process to a local optima as there is little or no exploration of the search space. Therefore, a modified best solution is used to update the pheromone value. We consider two additional solutions - iteration best ($s_{tb}$) and best solution since last convergence ($s_{rb}$). The iteration best solution is defined as $s_{tb} = \arg \max_{s \in \{s_1,...,s_n\}} Q(s)$. The best solution since last convergence, $s_{rb}$, is defined because the algorithm may converge to a solution before the end of the maximum iterations. In such a case, we reset $s_{rb}$. It follows that $s_{rb}$ captures the best solution since the algorithm was last reset. Both solutions, $s_{rb}$ and $s_{bs}$, are updated each iteration if they are worse than the iteration best solution $s_{tb}$.

The modified best solution, $s^*$, is a weighted combination of all three solutions. The weights are chosen proportional to the solution quality as this will direct the search process into regions with higher objective function values. In other words,

\[ s^* = \frac{\sum_{s \in S} Q(s)}{\sum_{s \in S} Q(s)} s \]  \text{ where } S = \{s_{tb}, s_{rb}, s_{bs}\} \quad (2.8) \]

In theory, the modularity function, $Q(s)$, can assume negative values as the range of $Q$ is $[-0.5, 1]$. In order to ensure that the weights assigned to each solution ($s$) in Eq.(2.8) is non-negative, we add a positive constant (0.5) to the modularity function. The pheromone vector, $\tau$, is updated as

\[ \tau \leftarrow \tau + \rho (s^* - \tau) \]  \quad (2.9)
where $\rho$ is the evaporation factor.

The evaporation factor controls the pace of search process i.e. higher values of $\rho$ will result in faster convergence but will likely lead to a severely sub-optimal solution. Hence, smaller values of $\rho$ are preferred. The modified best solution, $s^*$, clearly lies in a hyper cube as it a linear combination of extreme points of the hyper cube. This ensures that the pheromone vector always stays within the hypercube and can be used as probabilities in the solution construction phase. One of the biggest advantages of the hyper cube framework is that it can provide theoretical guarantees on the solution quality. In fact, if set $S$ in Equation (2.8) is replaced by $S = \{s^1, \ldots, s^n\}$, Blum and Dorigo [33] show that a monotone non-decreasing average solution quality can be guaranteed after each iteration (Theorem 1).

**Theorem 1.** (Blum and Dorigo [33]) Given a sufficiently large number of ants ($n_a$), the solution construction based on Equation (2.5) and pheromone update based on Equation (2.9), then the average solution quality does not decrease over time

$$
\mathbb{E}_n[Q(s)|\tau(t + 1)] \geq \mathbb{E}_n[Q(s)|\tau(t)]
$$

### 2.3.5 Convergence

It follows from Theorem (1) that the pheromone values must converge after a finite number of iterations as the objective function is bounded. Therefore, we check for convergence of the search process at the end of each iteration. The convergence factor $\lambda = \frac{2}{n} \sum_{i=1}^{n} |\tau_i - 0.5|$ captures the proximity of the current solution to an extreme point of the hyper cube. As $\lambda \to 1$ the search process converges to a solution. Also, to guard against any premature convergence of the algorithm, the pheromone values ($\tau$) and solution $s_{rb}$ are reset if convergence occurs before $T$ iterations are complete.

The results of applying this heuristic to artificial and real data sets are discussed in the next sections.
2.4 Results

We now test the accuracy and validity of our heuristic on real and synthetic social networks. The ACO heuristic is implemented with parameters \((\rho, n_a, T) = (0.05, 10, 50)\).

2.4.1 Mining Artificial Networks

The advantage of constructing artificial networks is that it naturally provides a validation scheme. The network consists of \(n\) nodes which are categorized into \(C\) groups (true communities) of equal size. The nodes within a community are connected with probability \(p\) and nodes in different communities are connected with probability \(q = fp\) where \(0 \leq f \leq 1\). These probability values are chosen such that the expected degree of every node is constant \(\bar{k}\). The relationship between the connection probabilities and the average degree of a node is given in Eq.(2.10). A sample graph with \(f = 0.05\) is shown in Figure 2.5(a).

\[
p\left(\frac{n}{C} - 1\right) + q(n - \frac{n}{C}) = \bar{k} \tag{2.10}
\]

The performance of the algorithm is recorded as \(f\) is increased from 0 to 1 i.e. expected number of edges going out of the communities increases and makes the communities more unclear. In the extreme case of \(f = 1\), the community structure is completely lost. For this experiment, we fix \(n = 128\), \(C = 4\) and \(\bar{k} = 16\). We measure the performance of the algorithm on three counts: (a) Optimal objective value achieved (b) Accuracy of the clusters detected and (c) Consistency of the clusters detected.

For the artificial test bed considered, we can approximate the expected optimal modularity value of the network when nodes are grouped according to the true communities. It follows from the definition of modularity (Eq.(2.1)) that the expected modularity of the network is shown in Eq.(2.11). It has already been shown that Eq.(2.11) is a good approximation for small values of \(f\) but communities with higher than expected modularity values tend to emerge for large values of \(f\) [37].

\[
\langle Q \rangle = \frac{1}{1 + f(C - 1)} - \frac{1}{C} \tag{2.11}
\]
We can see from Figure 2.5(b) that the average objective achieved by the ACO always achieves optimal objective (if not more). The solid line indicates the expected modularity whereas the error bars signify the mean and standard deviation of the computed modularity.

The other performance measure is the accuracy of community assignment. An information theory based measure, Variation Information (VI) [38], measures the fraction of nodes which change their community membership w.r.t their true communities. Let $X = \{x_1 \ldots x_r\}$ and $Y = \{y_1 \ldots y_s\}$ be two sets of communities. The measure $VI(X, Y)$ (see Eq.(4.12)) captures the fraction of nodes that have changed their community memberships between the two community assignments. It is easy to verify that the variation between two exactly same communities is 0. On the other hand, the maximum variation is achieved when one community assignment includes all nodes in a single community while the other involves $n$ different communities each containing a single node.

$$VI(X, Y) = \frac{1}{n\log n} \sum_{x \in X, y \in Y} n_{xy} \log \frac{n_x n_y}{n_{xy}}$$

(2.12)

where $n_x$ is the number of nodes in community $x$ and $n_{xy}$ is the number of nodes that are common to communities $x$ and $y$. From Figure 2.5(c), we can see that ACO heuristic misclassifies less than 10% of the nodes even as the expected number of edges inside a community is equal to the expected number of edges leaving the community i.e. $f = \frac{n-C}{n(C-1)} = 0.32$. This clearly indicates that the heuristic is able to identify accurate communities even under noisy input.

Thirdly, as the nature of the algorithm is stochastic, we need to perform robustness analysis to ensure that the solutions obtained are consistent and not a mere chance occurrence. Therefore, the algorithm is executed 100 times on the previously constructed network when $f = 0.32$. The fraction of times any given pair of nodes is grouped into the same community is plotted in Figure 2.5(d). A lighter shade indicates similar grouping of the nodes and a darker shade indicating the opposite. The white blocks along the diagonal indicates a fairly robust nature of the clusters detected by the algorithm.
Figure 2.5. The performance of ACO heuristic on a synthetic network. (a) A sample graph with $f = 0.05$ (b) The solid line indicates the expected modularity whereas the error bars indicate the modularity achieved. (c) The VI of detected communities w.r.t true communities is shown. Only 10% of the nodes are misclassified even when $f$ is increased to 0.35 (d) Test for the stochastic nature of the algorithm. Both the axis represents the node numbers. The black (white) region corresponds to the nodes which have never (always) been grouped together

2.4.2 Mining Real Social Networks

We now test our heuristic on many real world social networks with known communities [39]. One of the most commonly used testbed is a friendship network (see Figure 2.6) of a karate club[40]. Zachary (1978) had collected acquaintance information between 34 members in a karate club. The club split soon after due to a conflict between the club manager (node 34) and the instructor (node 1). Curiously, one can predict the split based on the friendship network data using the community detection algorithm. The optimal partitions obtained for karate club network almost matches exactly with known community structure except for node 10. In fact, assigning node 10 to the other partition decreases the modularity value. The reason for this observation may be the lack of weights in our model. For example, node 10 is connected to node 34 and node 3 with equal weights
Figure 2.6. The partition corresponding to maximum modularity obtained for Zachary’s karate club. The actual split had occurred due to a dispute between node 1 and node 34.

whereas node 34 is the group leader who probably influences node 10 more than node 3. In addition, there are two sub-communities which are densely connected to the key players (the manager and the instructor). One can infer that probably these sub-communities were closely related to the club’s fission.

The jazz music bands network [41] during 1920’s is another well studied example. There were a total of 198 bands identified and two bands are connected if they have a musician in common. In Figure 2.7, there are two main communities with a significantly less number of links between them. This is a reflection of the social divide between white and African American jazz bands. Moreover, some African American bands further segment into two subgroups indicating geographical divide. It has already been identified that New York and Chicago are two major cities where bands recorded which clearly correspond to the sub-groups formed [41].

We now consider mining online collegiate social networks [42]. Specifically, data regarding student friendships at California Institute of Technology (Caltech) is analyzed. In addition to the friendship information, self-identified attributes of students indicating gender, year of study, house affiliations, so on are also available. Our focus in this study is to determine the relationship between on-line and off-line friendship networks. The largest connected component consists of 762 students who belong eight different houses. The ACO heuristic partitions the network into 4 communities with a modularity score of 0.371. A previous study on
Figure 2.7. The best partition obtained for jazz music bands. In both figures, the solid line represents main partition and the dashed lines represent sub-partitions. Note that the algorithm does not explicitly identify main partitions and sub-partitions, authors used subjective evaluation to identify such partitions. Network layouts were generated using Pajek.

The same data reported a modularity score of 0.351 with seven communities[42]. The distribution of community assignments inside each of the eight student house affiliations is shown in Figure 2.8. The students that did not specify their house affiliations fall into the category of unspecified and are evenly distributed among the four communities. Note that for most houses (except house 1), majority of students in the house belong to the same community. This observation indicates that house affiliations play a big role in friendship formations. In other words, the off-line social networks drive the on-line social friendships to a large extent. Moreover, there are interesting relationships between various houses. There seems to exist another community structure at the level of houses. One can see that students in houses (2,8), (3,7) and (4,6) are more closer to each other than the rest of the houses. Such hierarchial nature of complex networks have been found in several other social and biological networks [43]. This analysis has huge implications in marketing where the objective is to identify specific nodes which have the maximum influence over the spread of opinions. For example, one can detect highest degree nodes within a community (provincial hubs) and highest degree nodes which bridge two different communities (connector hubs) as the potential seeds for marketing a new product. However, such studies require further network analysis which is outside the scope of this paper.
We have chosen several other networks as test beds to compare the accuracy of our approach with respect to existing techniques. It is interesting to note that the modularity value achieved for Jazz network (=0.595) is much higher than any of the available results (see Table 2.1). In fact, we find similar results for the metabolite network [44] and the E-mail network [45]. Our finding is significant because an improvement of 0.15 – 0.2 in modularity score is nearly a 50% increase in solution quality. Another interesting observation is that the number of communities found by ACO is (in most cases) less than the number found in the literature (see Table 2.1). The results are a clear indication that our ACO heuristic is able to significantly improve the solution quality relative to the existing algorithms. The success of ACO can be attributed to two important factors. Primarily, it is the ACO metaheuristic which guides the search process into regions which produce higher modularity values (Theorem 1). Secondly, the local search (1-opt-local search) greatly improves the solutions constructed by the ants based on pheromone levels.

Finally, it is worthwhile to shed some light on the computational complexity of our algorithm. On a network with $n$ nodes, the complexity of the ACO is mainly determined by the local search procedure as every other operation in algorithm is linear. It can be shown that initialization of gains in 1-opt-local-search is in $O(n^2)$ and the running time is $O(n)$ per iteration. Therefore, computational complexity
Table 2.1. Comparison of modularity values achieved and the number of communities found across various algorithms (EO - Extremal Optimization, NM - Newman’s Spectral Partitioning)and test networks. The entries are the modularity values achieved by each of the algorithms (and respective communities found).

<table>
<thead>
<tr>
<th></th>
<th>Nodes</th>
<th>EO [23]</th>
<th>NM [20]</th>
<th>ACO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Karate Club</td>
<td>34</td>
<td>0.419(4)</td>
<td>0.419(2)</td>
<td>0.419(4)</td>
</tr>
<tr>
<td>Jazz Musicians</td>
<td>198</td>
<td>0.445(5)</td>
<td>0.445(4)</td>
<td>0.595(4)</td>
</tr>
<tr>
<td>Metabolic</td>
<td>453</td>
<td>0.434(12)</td>
<td>0.434(10)</td>
<td>0.669(7)</td>
</tr>
<tr>
<td>E-mail</td>
<td>1333</td>
<td>0.574(15)</td>
<td>0.574(13)</td>
<td>0.716(8)</td>
</tr>
</tbody>
</table>

of local search is $O(n^2)$. Hence, the time complexity for one split of the network is $O(n_a T n^2)$. The total number of splits required is one less than the number of communities ($C$). It follows that the overall time is in $O(n_a TC n^2)$. As it is generally difficult to analytically estimate the number of communities, the overall complexity of our ACO heuristic will lie between $O(n^2)$ and $O(n^3)$. In order to verify our theoretical predictions, we record the actual running time of our heuristic on various problem instances ranging up to 10,000 nodes as shown in Table 2.2. We included two additional data sets - co-authorship network of physicists working in High Energy Physics Theory (HEP) and General Relativity (GR)[46]. Such collaboration networks are known to have modular structure as scientists often work in collaboration with other selected scientists. Our theoretical prediction on computational complexity is roughly verified by our computational experiments. Also, one can see that the complexity is extremely sensitive to the design parameters ($n_a, T$). For example, the complexity of a 1000 ($n$) node network with 20 ants ($n_a$) and 50 iterations ($T$) is equivalent to $O(n^3 C)$. Hence, one has to pay particularly close attention to the choice of design parameters. During our experiments, we found that the number of ants required to produce a high quality (high modularity) solution is proportional to the size of the network. For instance, in a 34 node network, just two ants are sufficient to produce a high quality solution. Any additional ants will only increase the computational burden without providing additional solution quality. A more detailed analysis is required to empirically estimate the relationship between the problem size and design parameters which is out of scope of this paper.
Table 2.2. Time taken by our ACO heuristic on various problem sizes. The computer program is written in Java and the computational experiments were run on a standard desktop windows pc with pentium 4 processor and memory of 1 Gb.

<table>
<thead>
<tr>
<th>Network</th>
<th>Nodes</th>
<th>Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Karate Club</td>
<td>34</td>
<td>\approx 0</td>
</tr>
<tr>
<td>Jazz Musicians</td>
<td>198</td>
<td>1</td>
</tr>
<tr>
<td>Metabolic</td>
<td>453</td>
<td>8</td>
</tr>
<tr>
<td>E-mail</td>
<td>1333</td>
<td>100</td>
</tr>
<tr>
<td>Co-authorship Network (GR)</td>
<td>5242</td>
<td>6823</td>
</tr>
<tr>
<td>Co-authorship Network (HEPt)</td>
<td>9875</td>
<td>19613</td>
</tr>
</tbody>
</table>

2.5 Conclusions

In this paper, we have proposed an algorithm to detect community structures in networks using ant colonies by maximizing the modularity measure. The ACO approach leads to significantly better solutions than existing algorithms with improvements of upto 50%. One of the main reasons for the improvement is the pheromone update rule which ensures a monotone increase in average solution quality after each iteration. The 1-opt-local search procedure embedded in the heuristic also significantly contributes to the overall solution quality. A better choice of data structures during the implementation can possibly improve the time complexity. There are many interesting future research directions. In the current context, we assume a static network structure which may not be the case as some of the networks. This is especially true for social networks where relationships between between people are dependent on time. Hence, we need to develop algorithms which can detect communities which evolve temporally. Also, fundamental research needs to be carried out to uncover the underlying principles of community formation. This will help researchers and practitioners to perform both mining and prediction in social networks.
Chapter 3

Detecting Robust Graph Partitions

3.1 Introduction

There are several engineered or naturally occurring systems exhibiting emergent phenomenon which cannot be understood by the behavior of its constituent elements. Such systems are often referred to as complex systems and their complexity arises due to dynamic interactions between their elements. A convenient modelling framework for representing complex systems is a network - with vertices as elements of the system and edges corresponding to their interactions. For example, WWW (World Wide Web) can be represented as a network with nodes as web pages and edges as hyperlinks between them. The networks of such systems often exhibit non-trivial topological features like power-law degree distributions, small average path lengths [47]. The overall objective of such analyses is to understand and predict the behavior of complex systems. Many real world networks are found to naturally decompose into modules or clusters which have high density of edges between them as opposed to going out of them. Such modules or communities often act as functional units and hence provide insights into the behavior of the network as a whole [48]. Therefore, detecting clusters could be of utmost importance.

More formally, suppose a graph is represented as $G = (V,E)$, where $V$ is the set of vertices and $E \subset V \times V$ is the set of edges connecting the vertices. Let each edge be associated with a weight $w_{ij} \ \forall (i,j) \in E$. The total strength of a vertex $i$ is $s_i = \sum_{j: (i,j) \in E} w_{ij}$ and the total strength of the graph is $m = \sum_{(i,j) \in E} w_{ij}$. A disjoint partition of the vertex set $V$ is a collection of clusters denoted by
\( P = \{C_1 \ldots C_K\} \) where \( \cup_k C_k = V \) and \( C_k \cap C_l = \phi \). Typically, a quality function is defined which measures the quality of a partition. A *modularity* measure is an effective tool to quantify the quality of the partition [49]. The quality of a partition \( P \) is defined as

\[
Q(P) = \sum_{C \in P} \left[ \sum_{(i,j) \in E, i,j \in C} \frac{w_{ij}}{m} - \left( \frac{\sum_{i \in C} s_i}{2m} \right)^2 \right]
\] (3.1)

Modularity measures the fraction of total strength observed within the cluster when compared to the expected fraction of strength. It has been found that several real world networks naturally decompose into clusters which have high modularity values. As a result, the problem of finding a partition which maximizes the quality has generated a lot of interest in recent years.

**Definition 1.** A *modularity maximizing partition of graph* \( G = (V, E) \) corresponds to the solution of the following optimization problem

\[
\max_P Q(P)
\] (3.2)

Clearly, graph partitioning is a combinatorial problem and not surprisingly it is NP-Complete [50]. There is a plethora of existing heuristics which try to maximize modularity measure [51, 23] Although, modularity maximization is necessary for the existence of meaningful partitions, it may not be a sufficient condition. Several drawbacks of modularity measure have surfaced over the past few years. Firstly, sparse random graphs tend to exhibit large modularity values although they do not possess any true community structure [13]. Secondly, there is an inherent resolution limit which forces optimal modularity partitions to merge smaller clusters to form large clusters [12]. As an illustrative example, consider the network shown in Figure 4.1. A standard modularity maximization when applied to the toy network yields two clusters indicated by vertex shapes (circles and squares). Interestingly, both clusters have the same quality of 0.1775. This is counter intuitive as one cluster is a clique (maximal connectivity) and the other is a tree (minimal connectivity). Intuitively, one would expect to find five clusters (indicated by the node colors) consisting of the clique and four singleton clusters. The inconsistency is a result of the global nature of modularity function i.e quality of a cluster depends on
Figure 3.1. A toy network illustrates the importance of detecting robust clusters. There are two clusterings shown - by color and by shape. The standard modularity maximization detects two clusters indicated by the shapes (circles and squares). The proposed robust modularity maximization approach detects five clusters indicated by the colors.

global network characteristics (Eg. number of edges). Such artifacts seem common to all global quality function and not just modularity. It has already been demonstrated by Karrer et al [52] that the authenticity of a community structure can be inferred by its resilience to perturbations/uncertainty. They performed sensitivity analysis on several real world networks to test the robustness of partitions to graph perturbations. They propose a graph perturbation scheme and study the relationship between the variation in optimal partitions and the level of perturbation. As expected, they conclude that random graphs (with non-robust partitions) exhibit far greater variation in communities as a result of perturbation when compared to non-random networks. Hence, detecting robust graph partitions can be posed as a problem of modularity maximization with uncertain network structure.

3.2 Robust partitions

Loosely, robust partitions can be defined as partitions which retain high $Q$-value even when subjected to perturbation. One way to discover robust partitions is to consider $Q$-value corresponding to the worst possible network perturbation scenario. To illustrate the notion of a robust community structure, consider a hypothetical network whose modularity scores are plotted for each of the possible partitions (See Figure 3.2, solid black line). The optimal deterministic partition (partition-5) maximizes modularity of the original graph. The dotted lines refer to the perturbations of the original graph. Notice that, partition-5, although optimal, cannot sustain its optimality under perturbation. In fact, partition-10 has a competing maxima which is more stable to perturbations. This phenomenon can be
captured if we plot only the minimum modularity score over all perturbations (see solid red highlighted line). We refer to the optimal solution of the minimum modularity plot, partition-10, as a \textit{robust graph partition}. There are several traditional methods which one could employ to deal with uncertain optimization problems - (i) Sensitivity Analysis (ii) Stochastic Programming (iii) Sampling/Scenario based analysis. Sensitivity analysis can quantify the robustness of an optimal solution but does explicitly identify alternative solutions which are more robust. Moreover, the computation effort required to solve multiple instances of a problem is prohibitively large. Most research concerning robustness or the stability of community structures [52, 53] is based on sensitivity analysis. Sampling based methods are useful when there are multiple factors (varying over a certain range) causing uncertainty. Although sampling based methods have been proven useful in practise, they share the drawbacks of sensitivity analysis. Stochastic programming based approaches, however, identify explicit robust solutions but often require probability distributions of uncertainties. Recently, Robust Optimization (RO) methodology has been introduced as an alternative approach for optimization under uncertainty. There have been several successful applications of RO in areas like engineering design, transportation, finance and so on [54]. In order to define a robust graph partition, we must first describe the perturbation scheme. Here we consider edge deletions as the only source of perturbations. Let $u_{ij} \in [0, w_{ij}]$ be the perturbation on edge $(i, j)$. As a result, we can express the edges of the perturbed graph as

$$w_{ij}(u_{ij}) = w_{ij} - u_{ij} \quad \forall (i, j) \in E$$

Typically, every edge in the network has an associated confidence score which can be interpreted as the cost of perturbation. Let $c_{ij} > 0$ be the confidence score associated with edge $(i, j)$. Then, an edge with large confidence/cost is less likely to be perturbed. In some cases, when cost is less than 1, it can be interpreted as the probability the an edge exists. Note that if all edges are allowed to be perturbed, then the robust partition will correspond to a trivial case of singleton clusters. In order to avoid such a situation, we impose a \textit{budget of perturbation} ($\Gamma$) so that we can easily control the total number of perturbed edges. We can now
Figure 3.2. Modularity plots depicting the concept of robust partition. Variation of modularity across graph perturbations (dotted lines). The solid line refers to the original graph which peaks at partition-5. The minimum modularity plot across all perturbations. The robust partition (partition-10) maximizes the minimum modularity.

define a perturbation set \( (U_\Gamma) \) in which all possible perturbations exist,

\[
U_\Gamma = \{ u : u \in [0, w], \sum_{(i,j) \in E} c_{ij} u_{ij} \leq \Gamma \} \tag{3.4}
\]

Modularity function under perturbation, \( Q_u(P) \), is a non-linear function of uncertainty. In order to maintain tractability, we linearize the perturbed modularity function by using a first order taylor series approximation around \( u = 0 \) (see Equation (3.5)).

**Assumption 1.** For small perturbations, \( u \), and large network size \( m \), the perturbed modularity function, \( Q_u(P) \) can be approximated as follows

\[
Q_u(P) \approx Q_0(P) - \frac{\sum_{C \in P} \sum_{(i,j) \in E} c_{ij} u_{ij}}{m} \tag{3.5}
\]

Note that the assumption is based on small perturbations which can be guaranteed by controlling the budget of uncertainty \( \Gamma \). Smaller perturbations also ensure that the underlying network is not altered drastically which would make robust clusters irrelevant.
We now define the robust partition detection as a solution of a max-min optimization problem.

**Definition 2.** The $\Gamma$-robust partition of graph $G$ corresponds to the optimal solution of the following max-min problem

$$\max_{\mathcal{P}} \min_{u \in U_{\Gamma}} Q_u(\mathcal{P})$$

It follows from the definition that when $\Gamma = 0$, the uncertainty is forced to be zero and therefore the robust modularity modularity maximization boils down to the original modularity maximization.

### 3.3 Algorithm to detect robust partitions

In this section we discuss some of the properties of the robust modularity maximization problem which will lay the foundation for an algorithm which can detect robust partitions. We introduce some additional notation which is helpful in illustrating the properties of robust partitions. Suppose the partition $\mathcal{P}$ is represented by a binary variable $x_{ij}$ indicating if an edge $(i, j)$ is included inside a cluster i.e

$$x_{ij}(\mathcal{P}) = \begin{cases} 
1 & \exists C \in \mathcal{P} \text{ s.t } i \in C, j \in C \\
0 & \text{else}
\end{cases}$$

Firstly, we show that the max-min problem can be reformulated as a pure maximization problem (Theorem 2).

**Theorem 2.** The max-min modularity maximization (Definition 2) is equivalent to the following maximization problem

$$\max_{\mathcal{P}, \alpha} \tilde{Q}(\mathcal{P}, \alpha) = Q_0(\mathcal{P}) - R(\mathcal{P}, \alpha)$$

where

$$R(\mathcal{P}, \alpha) = \sum_{(i,j) \in E} w_{ij} x_{ij}(\mathcal{P}) \left( \frac{1}{m} - c_{ij} \alpha \right)^+ + \Gamma \alpha$$
**Proof.** The proof proceeds in a similar fashion as that shown in [55].

We can write the max-min optimization problem (3.6) as the following

$$\max_{x \in X} \min_{u \in U} Q_0(P) - \frac{1}{m} \sum_{(i,j) \in E} u_{ij}x_{ij}(P)$$

$$\sum_{(i,j) \in E} c_{ij}u_{ij} \leq \Gamma \quad (\alpha)$$

$$u_{ij} \leq w_{ij} \quad (\beta_{ij}) \quad \forall (i,j) \in E$$

$$u_{ij} \geq 0 \quad \forall (i,j) \in E$$

Using strong duality, we can claim that the dual formulation of the inner minimization is equivalent to the following

$$\max_{\alpha \geq 0, \beta \geq 0, x \in X} -\alpha \Gamma - \sum_{(i,j) \in E} w_{ij}\beta_{ij} + Q_0(x)$$

$$-c_{ij}\alpha - \beta_{ij} \leq -\frac{1}{m}x_{ij}(P) \quad \forall (i,j) \in E$$

It follows from the above formulation that optimal $\beta_{ij} = \max(0, \frac{1}{m}x_{ij}(P) - c_{ij}\alpha)$. Moreover, as $x_{ij} \in \{0, 1\}$, we can rewrite $\beta_{ij} = (\frac{1}{m} - \alpha c_{ij})^+ x_{ij}(P)$. Note that this transformation is possible only due to the binary nature of the $x_{ij}$ variables. By exploiting this transformation, we can reformulate the dual maximization problem as

$$\max_{x \in X, \alpha \geq 0} \left[ Q_0(x) - \sum_{(i,j) \in E} w_{ij}x_{ij}(P) \left(\frac{1}{m} - c_{ij}\alpha\right)^+ x_{ij} - \Gamma \alpha \right]$$

Due to the addition of the new variable $\alpha$, the complexity of the robust modularity maximization increases dramatically as compared to the original modularity maximization. However, this reformulation provides interesting insights into the structure of robust modularity maximization problem. Firstly, we observe that for a given $\alpha = \bar{\alpha}$, maximizing $\tilde{Q}(P, \bar{\alpha})$ can be solved by leveraging existing modularity maximizing heuristics. We propose a modified greedy modularity maximization heuristic (Algorithm 1) originally proposed by Clauset et al [19]. The algorithm
recursively merges clusters until further merging does not increase modularity. The complexity of the original greedy modularity maximization is $O(d|E|\log(|V|))$ where $d$ is the number of merges. Note that we can retain the same computational complexity as the original algorithm as we are only affecting the initialization of the original algorithm.

**Algorithm 1** Modified Greedy Modularity Maximization

**Input:** Graph $G = (V, E)$, $\alpha$

**Output:** $P$

**Initialization:** Set $P = \{C_i\}_{i \in V}$ where $C_i = \{i\}$

Set $\Delta Q^{\text{merge}}(C_i, C_j) = \frac{2}{m} (w_{ij} - w_{ij}(1 - mc_{ij}\alpha)^+ - \frac{s_{i}s_{j}}{4m}) \forall (i, j) \in E$

Set $s(C_i) = s_i \forall i \in V$

while $\Delta Q^{\text{max}} = \max_{(C', C'')} \Delta Q^{\text{merge}}(C', C'') > 0$ do

Let $(C_k, C_l) = \arg \max \Delta Q^{\text{merge}}(C', C'')$

If both $C_k$ and $C_l$ are connected to $C_p$, then $\Delta Q^{\text{merge}}(C_p, C_k) \leftarrow \Delta Q^{\text{merge}}(C_p, C_k) + \Delta Q^{\text{merge}}(C_p, C_l)$

If only $C_k$ is connected to $C_p$, then $\Delta Q^{\text{merge}}(C_p, C_k) \leftarrow \Delta Q^{\text{merge}}(C_p, C_k) - \frac{1}{2m} s(C_p) s(C_k)$

If only $C_l$ is connected to $C_p$, then $\Delta Q^{\text{merge}}(C_p, C_k) \leftarrow \Delta Q^{\text{merge}}(C_p, C_l) - \frac{1}{2m} s(C_p) s(C_k)$

Update cluster strengths, $s(C_k) \leftarrow s(C_k) + s(C_l)$

Merge clusters, $C_k \leftarrow C_k \cup C_l$ and $P = P - C_l$

end while

On the other hand, for a given partition $P$, $\tilde{Q}(P, \alpha)$ can be solved efficiently as $R(P, \alpha)$ is piecewise linear and convex in $\alpha$. The function $R(P, \alpha)$ is convex in $\alpha$ as it is a non-negative linear combination of convex functions. Moreover, $R(P, \alpha)$ is piecewise linear function with break points in set $U_\alpha$ where

$$U_\alpha = \{0\} \cup_{(i,j) \in E} \{ (mc_{ij})^{-1} \}$$

The break points are the points in $\mathbb{R}^1$ where the gradient of the function $R(P, \alpha)$ with respect to $\alpha$ is not defined. As $R$ is convex and non-differentiable, we use the concept of a sub-differential to establish optimality conditions. It is straight
forward to show that the sub-differential, $\partial R(\mathcal{P}, \alpha)$, is given by

$$\partial R(\mathcal{P}, \alpha) = \left[ \Gamma - \sum_{(i,j) \in E} 1_{\alpha \leq (c_{ij} \cdot m)^{-1} w_{ij} c_{ij} x_{ij}(\mathcal{P})}, \Gamma - \sum_{(i,j) \in E} 1_{\alpha < (c_{ij} \cdot m)^{-1} w_{ij} c_{ij} x_{ij}(\mathcal{P})} \right]$$

(3.10)

The necessary and sufficient condition for a point, $\alpha^*$, to be optimal is that the sub-differential must contain 0 i.e

$$0 \in \partial R(\mathcal{P}, \alpha^*)$$

(3.11)

Clearly, the sub-differential is a singleton set for all $\alpha \notin U_\alpha$. Therefore, the optimal solution must be a break point i.e $\alpha^* \in U_\alpha$, as the slope changes only at the break points. A simple bisection search algorithm can be used to detect $\alpha^*$ in $O(\log(|U_\alpha|))$ time. In the worst case, all edges will have unique costs which will imply that $|U_\alpha| = |E| + 1$. As a result, the worst case complexity of minimizing $R(\mathcal{P}, \alpha)$ is $O(\log(|E| + 1))$.

**Algorithm 2** Detecting $\Gamma$-Robust Partitions

**Input:** Graph $G = (V, E)$, $\Gamma, \alpha_0$

**Output:** $\mathcal{P}^*$

**Initialization:** $Q_{best} = -\infty$, $k = 1$

**loop**

Solve $\mathcal{P}^k = \arg \max_{\mathcal{P}} \tilde{Q}(\mathcal{P}, \alpha^{k-1})$

Solve $\alpha^k = \arg \min_{\alpha \in U_\alpha} R(\mathcal{P}^k, \alpha)$

if $\tilde{Q}(\mathcal{P}^k, \alpha^k) > Q_{best}$ then

$\mathcal{P}^* = \mathcal{P}^k$

$Q_{best} = \tilde{Q}(\mathcal{P}^k, \alpha^k)$

else

Exit the loop

end if

$k = k + 1$

end loop

These two observations directly hint towards an alternating direction maximization algorithm. An intuitive algorithm (Algorithm 2) to detect a robust partition is to alternatively optimize $\mathcal{P}$ and $\alpha$ until there is no improvement in objective function. The convergence of Algorithm 2 immediately follows from the fact that
the objective function, $\tilde{Q}(\mathcal{P}, \alpha)$, strictly increases after each iteration. As $\tilde{Q}(\mathcal{P}, \alpha)$ is bounded from above, the algorithm has to converge to some $(\mathcal{P}^*, \alpha^*)$. For each iteration, the overall complexity of the algorithm is $O(d|E| \log(|V|) + \log(|E| + 1))$.

The alternating direction maximization approach can get stuck at local optima. Therefore, we run Algorithm 2 multiple times with randomly chosen $\alpha \in U_\alpha$ as starting points and pick the best solution.

### 3.4 Related Work

There are several graph partitioning algorithms which have been proposed in the literature which heuristically maximize modularity function. A trivial way to account for graph perturbation and the resulting uncertainty is to delete edges whose cost is below than a certain threshold cost. One can then use existing graph partitioning algorithms on the resultant graph. A major challenge, however, is the identification of the threshold. In our model, the dual variable $\alpha$ can be interpreted as a threshold whose value is automatically computed by the algorithm.

Other approaches include sensitivity analysis where edges in the network are randomly perturbed and graph partitioning algorithm is re-run on perturbed graphs [52, 53]. The disadvantage of such approaches is that it does not identify a robust partition. They only test the robustness of a given partition. A separate stream of work has been going which address the topic of data mining with uncertain data[56, 57]. Typically, data resides in metric spaces (Ex: Euclidian space) and the location of each data point is uncertain (often specified by a pdf). Then, standard deterministic data mining algorithms can be adapted to minimize the expected distance between cluster members[58]. Ngai et al [59] propose Uncertain K-means algorithm as a variant of the standard K-means which works on expected distance between cluster centers and data points. The difficulty in adapting such algorithms is due to the technical challenges resulting from a lack of a simple distance measure between vertices of the graph. For instance, any distance measure based on path length between two nodes will break down if the graph perturbation resulted in isolated components. A work around this situation is to use a similarity measure which is valid under all possible realizations of uncertainty. Recently, kollios et al [60] use a simple similarity measure based on random null model which
is well defined for all perturbations. A limitation of this approach is that the similarity measure used in the study is naive and is known to be less effective than modularity measure. On the other hand, modularity based similarity measure is not well defined for all possible realizations of uncertainty (Ex: if all edges of the network are deleted). This is precisely the advantage of our proposed robust optimization based model where we ensure the modularity function is well defined by excluding extreme cases.

3.5 Results

We will now illustrate the usefulness of robust partitions on both real and artificial data sets. The implementation of the algorithm was done in Java. All experiments were performed on a standard desktop PC with 2 GB RAM and 1.66 Ghz Intel processor. In several data sets, the confidence level on edges is not readily available. So we propose a technique to infer edge confidence levels from topology of the graph. We claim that edges between nodes with fewer common neighbors have lower confidence levels (easier to perturb). This follows from a previous study [61] that has shown that future links can be predicted reasonably well based on the number of common neighbors. This observation indicates that an edge \( e_1 \) is more likely to be spurious than edge \( e_2 \) if end points of edge \( e_1 \) have fewer common neighbors than those of \( e_2 \). In other words, one can interpret similarity of an edge as the cost of perturbing that edge. The similarity of an edge can be captured by the following expression

\[
c_{ij} = c_{min} + \sum_{k \in N_i \cap N_j} w_{ik} w_{jk} \frac{\min(s_i, s_j)}{m} \quad \forall (i,j) \in E
\]

(3.12)

where \( c_{min} \in (0,1] \) is a constant. Unless specified otherwise, for all of our experiments we set \( c_{min} = 0.01 \) and \( \Gamma = 0.01 m \bar{c} \) where \( \bar{c} \) is the average weighted cost of perturbing an edge i.e.

\[
\bar{c} = \frac{\sum_{(i,j) \in E} c_{ij} w_{ij}}{m}
\]

Also, the alternating direction algorithm is executed with 10 randomly chosen starting points. In order to compare a robust partition to a non-robust partition
we employ the variation of information measure \((VI)\). Suppose \(\mathcal{P}_1\) and \(\mathcal{P}_2\) are any two partitions, then distance between partitions is defined as

\[
VI(\mathcal{P}_1, \mathcal{P}_2) = \frac{1}{N \log N} \sum_{C \in \mathcal{P}_1} \sum_{D \in \mathcal{P}_2} -|C \cap D| \log \left( \frac{|C \cap D|^2}{|C||D|} \right)
\]  

(3.13)

In this section, we compare and contrast the non-robust partitions to robust partitions under stylized artificial networks. Our objective is to show how certain drawbacks of traditional modularity maximization can be alleviated using robust partitions.

### 3.5.1 Artificial Networks - Insensitivity to noise

We construct a graph with \(N\) vertices. The vertices are divided in \(c\) clusters each containing \(n = \frac{N}{c}\) vertices. All vertices within each group are connected with probability \((p)\) and vertices belonging to different groups are connected with a probability \(q\). Given this graph construction process, the expected degree of a node is \(k = (n - 1)p + (N - n)q\). For all our experiments, the average degree of nodes is held constant. Moreover, \(q\) is set to a fixed small fraction \((f)\) of \(p\) so as to ensure that most edges lie within groups i.e. \(q = fp\). The only variable \(f\) controls the number of noisy edges in the network in addition to the edges emanating from the outlier nodes. We are interested in comparing the accuracy of robust and non-robust partitions with respect to the ideal partition as noise level \((f)\) is varied. The noise level \(f \in [0, f_{\text{max}}]\) where \(f_{\text{max}}\) is the noise level when each node will have more edges outside of its cluster. It is easy to see that

\[
f_{\text{max}} = \frac{n - 1}{N - n}
\]

We tested on a network with \(N = 100, c = 10\) and \(k = 8\) and plotted the accuracy of robust and non-robust partitions at different noise levels (Figure 3.3). Note that higher accuracy is equivalent to smaller variation of information between detected partition and the true partition. When \(f \approx 0\), both robust and non-robust partitions correctly identify the \(c\) clusters as there are very few edges between clusters. As the noise level increases, robust partitions are found to be significantly more accurate as compared to the non-robust partition. The proposed methodology is
Figure 3.3. Accuracy of robust partitions as compared to non-robust partitions at varying noise levels. Each data point is average over 10 independent replications.

able to lessen the impact of noise by selecting the partition which has a better worst case performance. When \( f \approx f_{max} \), although robust partitions have better accuracy on an average, they are not significantly better than non-robust partitions. This observation can be explained due to the lack of an inherent modular topology in the network at higher noise levels. Therefore, it is difficult for both robust and non-robust modularity functions to differentiate between inter and intra cluster edges. Our analysis on artificial noisy graphs clearly demonstrates the utility of detecting robust partitions.

3.5.2 Artificial Networks - Resolving clusters at different scales

As we have already discussed, modularity based partitioning in large networks tend to merge several small clusters into one large cluster. We want to show that robust partitions can avoid this situation. Consider a graph consisting of a circular chain of \( p \) cliques of size \( n \). Each clique is connected to two other cliques via a single
Figure 3.4. The ability of robust partitions to detect clusters at higher levels of resolution as compared to non-robust partitions. The adjacency matrix is constructed by setting $n = 3$ and $p = 10$

edge. Let $m = 0.5n(n - 1)$ be the number of edges in each clique. Therefore, the total number of nodes $N = np$ and edges $M = mp + p$. Clearly, the best partition of the graph should identify each clique of the graph to be a cluster. Let us refer to this partition as $P_1$. Also, let $P_2$ represent another partition consisting of $\frac{p}{2}$ clusters where a cluster is a combination of two connected cliques. It has already been shown that $Q_0(P_1) \geq Q_0(P_2) \iff p \leq 2(m + 1)$. Therefore, as the number of cliques ($p$) increases beyond a certain limit, cliques will be merged together. We will now show that by incorporating robustness, such a situation can be avoided. We construct a graph consisting of $10(= p)$ cliques of size $3(= n)$ (See Figure 3.4). Note that the non-robust partition contains only 5 clusters which merges two consecutive cliques. On the other hand, the robust partition produces the expected 10 cluster partition. The main reason is because the cost of perturbing edges connecting two cliques is less than $c_{min}$ - as their endpoints do not share any common neighbors. Therefore, even a small budget of perturbation is sufficient to produce large decrement in robust modularity value $\tilde{Q}$. Although we have used a highly stylized example, we expect similar findings in a more general case.

3.5.3 Real World Network- Football Network

We now show the applicability of the robust partitions on a real world network. The vertices of a football network are the football teams representing various american universities. An edge between two teams indicates a game between them. There
are a total of 179 football teams in the 2006 NCAA Division-I which played 760 regular season games among themselves. Only 119 teams belong to Division-I and the other 60 belong to lower divisions. Moreover, among all Division-I teams, 115 teams are divided into 11 conferences with 4 independent teams (not affiliated to any conference). Figure 3.5 shows the topology of the network and also illustrates the 11 conferences and the four independent teams. All nodes in the periphery of the network are lower division teams. The network consists of only 759 edges as two teams - Houston and Southern Mississippi - played each other twice. As teams in a conference play more games with each other, a natural set of clusters corresponding to 11 conferences emerge. The original modularity maximization results in a partition which has a modularity score of 0.60, Figure 3.6. However, there are several conferences that are incorrectly merged together. Moreover, all lower division teams are clustered into one of seven clusters detected instead of identifying them as isolated clusters. Its distance from the true partition (as measured by variation of information) is 0.357. On the other hand, robust partition has a lower modularity score of 0.54 but detects more sensible clusters, Figure 3.7. It detects 10 out of the 11 conferences correctly. The Mountain West and the Western Athletic Conference (WAC) are incorrectly merged together as they have several games between each other. All but six of the lower division schools are identified as isolated clusters. The distance between the true partition and robust partition is 0.052 which is much lower than the non-robust partition. Clearly, incorporating robustness improves the accuracy of clustering process.

3.6 Conclusion

In this paper, we have proposed a robust optimization based framework to define robust graph partitions and a scalable algorithm to detect them. We maximize modularity when the underlying network is subjected to edge deletions. We show that the accuracy of detected clusters increases significantly by incorporating robustness. Another advantage of our model is that it can incorporate user-defined information (through edge costs) into the clustering process. We have shown the utility of our approach by experimenting on both real and artificial networks. In the future, we want to consider both edge deletions and additions as the source
Figure 3.5. Football Network consisting of 179 vertices and 759 edges. All 11 conferences are annotated (and color coded) along with the 4 independent teams in the middle.

of perturbations. We want to extend our robust optimization based approach to other cluster quality functions.
Figure 3.6. Non-Robust Partition of Football Network ($Q = 0.60$ and $VI = 0.357$)

Figure 3.7. Robust Partition of Football Network ($Q = 0.548$ and $VI = 0.052$)
4.1 Introduction

Many biological, social and technological networks tend to naturally decompose into densely connected subgraphs which often act as functional units of the system. Hence, such subgraphs (also known as clusters/communities) provide insights into the behavior of the network as a whole [62]. As the modular nature of complex networks seems to be universal, detecting such clusters could be of utmost importance. A cluster or community is defined as a set of nodes which have a much higher number of connections between them than expected from the average connection density of the network. Clearly, there is an ambiguity in the definition of a cluster due to lack of a proper threshold on density. As a result researchers have proposed several definitions of clusters which in turn lead to a variety of algorithmic approaches [9, 17]. Here, our definition of cluster is based on modularity measure \( Q \) [20] which quantifies the quality of the cluster. The modularity measure is proportional to the excess number of observed edges inside the cluster when compared to number of expected edges in a random graph with the same degree distribution. We are interested in detecting a collection of clusters in a graph (referred to as clustering). The quality of the clustering is just the sum of qualities of individual clusters. We will need the following notation to introduce modularity function.
Let us consider a general weighted directed graph $G = (V, W)$ where $V$ denotes the set of vertices s.t $|V| = n$ and $W = [w_{ij}]$, $w_{ij}$ being the strength of connection between nodes $i$ and $j$. In addition, we define out and in strength vectors of $G$ as $S^{\text{out}} = W \mathbf{1}$, $S^{\text{in}} = \mathbf{1}^T W$. The total strength of the network, $S = \mathbf{1}^T W \mathbf{1}$ where $\mathbf{1}$ represents a $n \times 1$ vector of ones. Any clustering, $c$, is obtained by assigning each vertex $i$ in graph $G$ a label $c_i$ where the set of vertices with the same label belong to the same cluster. The quality of clustering $c$ defined on graph $G$ is given by

$$Q(c, G) = \frac{1}{S} \sum_{(i, j)} b(i, j) \delta(c(i), c(j)),$$  \hspace{1cm} (4.1)$$

where $B = [b_{ij}] = W - \frac{1}{S} S^{\text{out}} S^{\text{in}}$ is the modularity matrix. More details on the derivation of modularity measure can be obtained from [20]. It has been observed that sensible clusterings in several real world networks also correspond to clusterings with the optimal or near-optimal $Q$ values [63]. Hence, one can pose the problem of community detection as an optimization problem i.e. finding a clustering which maximizes the quality. Not surprisingly, modularity maximization is hard and is in fact strongly NP-Complete [50]. Currently, there is a plethora of existing heuristics which try to maximize modularity measure [23, 64, 9]. However, the output of most existing algorithms is a single clustering which is not sufficient to gain a complete understanding of the underlying graph. In extreme cases, considering only the optimal clustering may be even misleading. We illustrate the above discussion through a toy example of a network having 12 nodes as shown in Figure 4.1. The optimal clustering, shown in Figure 4.1(a), corresponds to three clusters of the network with a respectable modularity score of 0.3947. Clearly, the nodes 9-12 (green circles) hardly constitutes a cluster as it has more edges leaving the group than the ones within the group. Such an anomaly is observed because of the global nature of modularity maximization. The expected number of edges in a group decreases as the network becomes larger. For this reason, the expected number of edges between nodes 9-12 is less than the observed number of edges which incorrectly suggests that it is a modular partition. On the other hand, an alternative clustering, shown in Figure 4.1(b) with two clusters is more meaningful although it has a slightly lower $Q$ score of 0.3934. One can only expect the situation to worsen as the size of the network increases. In fact, Good et al [63] showed
the existence of alternative partitions in large networks which are very diverse but still have large $Q$ values. These partitions remain largely unexplored by traditional algorithms because the space of all possible partitions is prohibitively large for an exhaustive search. It now becomes imperative that one needs to uncover an ensemble of partitions in order to confidently draw inferences about the underlying graph structure.

There are two main conditions under which detecting alternative clusterings is useful: (a) near-optimal modularity value and (b) dissimilar structure. In this paper, we will show that potentially meaningful clusterings can be extracted by perturbing the graph and optimizing modularity on the perturbed graph. Intuitively, the graph perturbations will alter the landscape of the modularity function. As the level of perturbation increases, various alternative clusterings emerge as optimal solutions of modularity maximization.

The organization of this paper is as follows: Section 4.2 talks about existing literature, Section 4.3 introduces the graph perturbation, Section 4.4 discusses some analytical properties of our proposed perturbation followed by numerical results on large scale networks and conclusion in Sections 4.5 and 4.6 respectively.

## 4.2 Related Work

Initial work on alternative clusterings was primarily directed towards feature based clustering although some of them can be readily extended to graph clustering [65, 66, 67, 68]. We focus more on the graph ensemble clustering literature. One
of the earliest attempts at ensemble clustering was performed using randomization techniques like Simulated annealing[69]. Typically, new cluster samples are generated as a result of moving vertices from one community to another. A drawback of this approach is the prohibitively large number of samples required to ensure diversity of alternate solutions. More recently, diversity constraint was explicitly modeled in an integer programming formulation of modularity maximization[70]. The new constrained modularity maximization was solved using the techniques proposed by Aggarwal and Kempe[71]. Although this approach did provide encouraging results, it is not scalable for large networks. A more scalable approach was suggested by Duggal et al [72] who develop an algorithm to recursively split and merge existing clustering and obtain alternative near-optimal clusterings. Their method is flexible enough to incorporate diversity as an additional factor in choosing new clusterings. The advantage of their approach is that they can prove the new clusterings generated are the next best solutions in a constrained tree subspace of clusterings. Their method, however, works only on hierarchical clustering algorithms. In fact all the previously discussed algorithms are algorithm dependent. Our work departs from the existing literature in this aspect. We propose a graph perturbation technique which is independent of any algorithm. Moreover, we do not explicitly add constraints to obtain diverse clusterings. We will show analytically that diversity is implicit in our perturbation scheme by proving that the clustering solution quality will decrease monotonically as the perturbation increases.

The use of graph perturbations has already been employed in the past[53, 52, 73] but to study a closely related concept of network robustness. It is possible that if the optimal clustering is robust then all of the alternative clustering will be similar to one another. For example, a graph consisting of isolated cliques will result in each clique being identified as a separate cluster. This clustering will likely retain its optimality under all graph perturbations. However, this is seldom the case in real world networks as there are no obvious clusterings which motivates the need to detect alternative clusterings.
4.3 Graph Perturbation

All graph perturbation schemes discussed in the literature\cite{52,73}, generate a family of random graphs which often share certain topological graph features like total number of expected edges or expected degrees of nodes. But modeling stochastic uncertainty in edges and its non-linear dependence on modularity function will definitely compromise any analytical tractability. Therefore, we propose a deterministic family of perturbed graphs which will give way to tractable alternative clustering detection methodology. In order to describe the family of deterministic graph perturbations, we need a simple graph transformation technique. The original graph $G = (V,W)$ is modified by adding a dummy isolated node $i_{dummy}$. Let us refer to this new graph as $G_0 = (V_0,W_0)$ where $V_0 = V \cup \{i_{dummy}\}$ and $W_0$ is defined in Eq.(4.2).

$$w_0(i,j) = \begin{cases} w(i,j) & i \neq i_{dummy}, j \neq i_{dummy} \\ 0 & \text{else} \end{cases} \quad (4.2)$$

We observe that isolated nodes do not affect the modularity of original graph $G$ i.e.

$$Q(c,G_0) = Q(c',G) \forall c \text{ where } c'(i) = c(i) \forall i \in V \quad (4.3)$$

Due to this property, detecting clusterings on graph $G_0$ is equivalent to that on graph $G$. We will restrict our discussion to graph $G_0$ as our graph perturbation scheme will be based on $G_0$. The role of the dummy node is to induce uncertainty in graph structure. As the degree of perturbation increases, non-dummy nodes weaken their connections with existing neighbors and form new connections with the dummy node. In the extreme case, this perturbation will result in star graph, $G_1$, with the dummy node as the hub i.e $G_1 = (V_0,W_1)$ where $W_1$ is defined in Eq.(4.4).

$$w_1(i,j) = \begin{cases} 0.5S_{out}^i & i \neq i_{dummy}, j = i_{dummy} \\ 0.5S_{in}^j & i = i_{dummy}, j \neq i_{dummy} \\ 0 & \text{else} \end{cases} \quad (4.4)$$

Note that the total strength of $G_1$ is the same as $G_0$ but the original degree distributions are not preserved. The motivation for this perturbation scheme is
Figure 4.2. Figures illustrating the family of perturbed graphs when $v = \{0, 0.5, 1\}$. Width of the edge represents the strength of connection. A perturbed graph is constructed by (1) adding a dummy node and (2) adding new edges between the dummy and all non-dummy nodes. The weight on any edge at perturbation level, $v$, can be computed from Eq(4.6).

drawn from the fact that the modularity of any clustering $c$ with respect to a star graph $G_1$ is always non-positive.

$$Q(c, G_1) \leq 0 \forall c$$  (4.5)

In fact, the maximum modularity for a star graph corresponds to grouping all nodes in a single cluster. One can view graphs $G_0$ and $G_1$ as two extremes of a family of graphs which exhibit varying levels of perturbations (see Figure 4.2). This observation leads to a natural definition of a perturbed graph, $G_v(V_0, W_v)$, as a linear combination of $G_0$ and $G_1$ for some $v \in [0, 1]$ i.e.

$$W_v = (1 - v)W_0 + vW_1, v \in [0, 1]$$  (4.6)

The parameter $v \in [0, 1]$ controls the degree of perturbation as a total of $Sv$ links of the original graph are perturbed for a given value of $v$ where $S$ is the total strength of the network. Notice that as the perturbation increases, more edges leave a partition than lie within it. Therefore, optimal clustering is progressively weakened and alternative clusterings emerge.

It is useful to define a notion of perturbed clustering as shown in Definition 3.

**Definition 3.** Perturbed clustering $(c_v)$ corresponds to the optimal modularity clus-
tering on graph $G_v$

Note that the perturbed clusterings are defined on graphs containing the dummy node. As discussed earlier, one can simply ignore the effect of dummy node on unperturbed graph $G_0$. Also, a precise value of $v$ is not critical as we will be interested in extracting an ensemble of clusterings at various perturbation levels. However, large perturbation levels will result in trivial partitions. For example, $v = 1$ will result in trivial solutions such as grouping all nodes in the same group.

We will now study some of the properties of perturbed clustering in the next section.

### 4.4 Properties of Perturbed Clusterings

We first show an interesting result describing the dependence of $Q(c, G_v)$ on perturbation level, $v$, in Theorem 3. This will prove helpful in establishing several important properties of perturbed clusterings.

**Theorem 3.** The relation, $Q(c, G_v) = (1 - v)Q(c, G_0) + v^2Q(c, G_1)$, holds true for any clustering $c$

**Proof.** Consider graph $G_v$ whose connectivity matrix $W_v = (1 - v)W_0 + vW_1$. It follows that $S_v^{in} = (1 - v)S_0^{in} + vS_1^{in}$ and $S_v^{out} = (1 - v)S_0^{out} + vS_1^{out}$. The modularity matrix, corresponding to graph $G_v$, is given by

$$B_v = W_v - \frac{1}{S_v^{out}}S_v^{in} \quad \text{(4.7)}$$

We will first evaluate the second term.

$$S_v^{out}S_v^{in} = (1 - v)^2S_0^{out}S_0^{in} + v^2S_1^{out}S_1^{in} + v(1 - v)(S_0^{out}S_1^{in} + S_1^{out}S_0^{in})$$

$$= (1 - v)^2S_0^{out}S_0^{in} + v^2S_1^{out}S_1^{in} + v(1 - v)(S_0^{out}S_1^{in} + SW_1)$$

$$= (1 - v)S_0^{out}S_0^{in} + v^2S_1^{out}S_1^{in} + v(1 - v)SW_1 \quad \text{(4.8)}$$

Combining Eq.(4.7) and Eq.(4.8), we get

$$B_v = (1 - v)(W_0 - \frac{1}{S_0^{out}}S_0^{in}) + v^2(W_1 - \frac{1}{S_1^{out}}S_1^{in}) = (1 - v)B_0 + v^2B_1$$
Therefore, $Q(c, G_v) = (1 - v)Q(c, G_0) + v^2 Q(c, G_1)$

In order to examine the dependence of perturbed clustering ($c_v$) on the perturbation parameter ($v$), we first observe an important property that $Q(c, G_v)$ corresponding to two different clusterings can intersect at most once in $v \in [0, 1]$.

**Property 1.** The modularity functions, $Q(c, G_v)$ and $Q(c', G_v)$, can intersect at most once in $[0, 1]$ for two distinct clusterings $c$ and $c'$.

*Proof.* We show that $z(u)$ can have at most one real root in $[0, 1]$ where $z(u)$ is defined as

$$z(u) = Q(c, G_u) - Q(c', G_u)$$

$$= u^2(Q(c, G_1) - Q(c', G_1)) - u(Q(c, G_0) - Q(c', G_0)) + Q(c, G_0) - Q(c', G_0)$$

There are three possible cases:

1. $Q(c, G_0) = Q(c', G_0)$ and $Q(c, G_1) = Q(c', G_1)$

   This case contradicts our original assumption that $c_1 \neq c_2$.

2. $Q(c, G_0) \neq Q(c', G_0)$ and $Q(c, G_1) = Q(c', G_1)$

   The only real root of $z(u)$ corresponds to $u = 1$.

3. $Q(c, G_1) \neq Q(c', G_1)$

   For this case, $z(u) = 0 \iff u^2 - u\alpha + \alpha = 0$ where $\alpha = \frac{Q(c, G_0) - Q(c', G_0)}{Q(c, G_1) - Q(c', G_1)}$.

   The only real root in $[0, 1]$ occurs at $u = \frac{\alpha}{2} + \sqrt{\frac{\alpha^2}{4} - \alpha}$ and $\alpha \leq 0$.

The above mentioned property indicates that once a clustering ceases to be optimal, it cannot regain its optimality as the perturbation level $v$ increases (Property 3). Alternatively, if a clustering retains its optimality at two distinct perturbations levels, $v_1$ and $v_2$, then it will remain optimal throughout the interval $[v_1, v_2]$ (Property 2). These results seem to indicate that perturbed clusterings are optimal for certain perturbation intervals. One can interpret the length of such intervals as an indication of the robustness of the clustering. In theory, one can make use of this observation to quantify robustness of clustering. A more detailed analysis is required which is not the focus of this paper.
**Property 2.** Suppose $c_{v_1} = c_{v_2}$ for some $0 \leq v_1 < v_2 \leq 1$. Then, $c_v = c_{v_1} = c_{v_2}$ $\forall v \in [v_1, v_2]$.  

*Proof.* We will prove by contradiction. Suppose $\exists c_{v_3} \neq c_{v_1}$ for some $v_3 \in (v_1, v_2)$. This immediately follows from Definition 3 that $Q(c_{v_3}, v_3) > Q(c_{v_1}, v_3)$. We already know that $Q(c_{v_3}, v_1) \leq Q(c_{v_1}, v_1)$ and $Q(c_{v_3}, v_2) \leq Q(c_{v_1}, v_2)$. As $Q(c_{v_1}, u)$ and $Q(c_{v_3}, u)$ are continuous in $u$, both functions should intersect at two distinct points in $[v_1, v_2] \subseteq [0, 1]$. This is in contradiction with Property 1.  

**Property 3.** Suppose $c_{v_2} \neq c_{v_1}$ for some $0 \leq v_1 < v_2 \leq 1$. Then, $c_v \neq c_{v_1}$ $\forall v \in [v_2, 1]$.  

*Proof.* We will prove by contradiction. Suppose $\exists c_{v_3} = c_{v_1}$ for some $v_2 < v_3 \leq 1$. As $c_{v_1}$ is optimal when $v = v_1$ and $v = v_3$, it follows from Property 2 that $c_{v_1}$ is optimal when $v = v_2$ i.e. $c_{v_1} = c_{v_2}$. This contradicts the fact that $c_{v_1} \neq c_{v_2}$.  

We now want to show that diversity of alternative partitions increases with perturbation levels. One can provide indirect evidence for diversity by showing that the quality of perturbed clusterings w.r.t the original graph ($G_0$) decreases with level of perturbation(Property 4). Note that similar clusterings will have similar quality ($Q$ scores). Therefore, it is very unlikely that a significant decrease in clustering quality will correspond to a similar clustering.  

**Property 4.** The quality of perturbed clustering, $Q(c_v, G_0)$, is non-increasing in $v$.  

*Proof.* We will prove by contradiction. Suppose $Q(c_{v_2}, G_0) > Q(c_{v_1}, G_0)$ where $v_2 > v_1$. By definition, $Q(c_{v_2}, G_{v_1}) < Q(c_{v_1}, G_{v_1})$ and $Q(c_{v_2}, G_{v_2}) > Q(c_{v_1}, G_{v_2})$. As $Q(c_{v_1}, G_u)$ and $Q(c_{v_2}, G_u)$ are continuous in $u$, they must intersect twice in $[0, v_2]$ which is in contradiction with Property 1.  

We now turn our attention to the mechanics of alternate community detection. More specifically, we provide insights the dependence of the topology of detected clusters on perturbation level. To this effect, we introduce some additional notation. Suppose the optimal clustering $c_v$ detected at perturbation level $v$ partitions
the vertices $V_0$ into $\mathcal{P}_v$, i.e. $\bigcup_{P \in \mathcal{P}_v} P = V_0$ and $P \cap R = \{\} \ \forall P, R \in \mathcal{P}_v$. We define the strength of a cluster $P$ as

$$S(P) = \sum_{i \in P} S^{in}(i) + S^{out}(i),$$

where the in and out-strength of a cluster is defined on the original graph (when $v = 0$). Finally, let $P^* \in \mathcal{P}_v$ represent the strongest cluster in $\mathcal{P}_v$, i.e.

$$S(P^*) \geq S(P) \ \forall P \in \mathcal{P}_v$$

The transformations observed in $\mathcal{P}_v$ (w.r.t perturbation) can be explained by combination three fundamental operations: (i) Merging of two disjoint clusters (ii) Splitting of a cluster and (iii) Moving vertices from one cluster to another. We show the effect of above mentioned operations on the quality of clustering (modularity). We introduce additional notation which are useful to prove properties (5-7)

- Let $S_v(A,B) = \sum_{i \in A,j \in B} w_{ij}(v)$ denote the strength of all edges starting in set $A \subseteq V_0$ and ending in $B \subseteq V_0$ at perturbation level $v$. It is straightforward to show that

$$S_v(A,B) = (1 - v)S_0(A,B) + \frac{v}{2}\delta(i_{dummy} \in B)S_0(A,V_0) + \frac{v}{2}\delta(i_{dummy} \in A)S_0(V_0,B) \quad (4.9)$$

- Let $P^* \in \mathcal{P}_v$ be the strongest cluster in $\mathcal{P}_v$. In other words,

$$S_0(P^*,V_0) + S_0(V_0,P^*) \geq S_0(P,V_0) + S_0(V_0,P) \ \forall P \in \mathcal{P}_v \quad (4.10)$$

- Suppose $A, B$ are two disjoint clusters of a partition $\mathcal{P}$, then the change in modularity resulting from a merger of $A$ and $B$ is given by

$$\Delta Q^\text{merge}_v(A,B) = \frac{S_v(A,B) + S_v(B,A)}{S_0(V_0,V_0)} - \frac{S_v(A,V_0)S_v(V_0,B) + S_v(B,V_0)S_v(V_0,A)}{S_0(V_0,V_0)^2} \quad (4.11)$$
Firstly, at any perturbation level \( v \), it is not optimal to move the dummy node \((i_{dummy})\) from the cluster which has the largest strength \((P^*)\) to any other cluster (see Property 5). The intuition behind this observation is the fact that the strength of connections between the dummy node and a cluster is proportional to the strength of the cluster.

**Property 5.** Let \( P^* \in \mathcal{P}_v \) be the strongest cluster detected at perturbation level \( v \). Then, \( i_{dummy} \in P^* \)

**Proof.** Suppose \( i_{dummy} \in P^* \). We will show that the change in modularity as a result of moving \( i_{dummy} \) from \( P^* \) to \( P \) \((\Delta Q^{move}_{v}(i_{dummy},P^*,P))\) is always negative. It follows from from Theorem (3) that,

\[
\Delta Q^{move}_{v}(i_{dummy},P^*,P) = (1-v)\Delta Q^{move}_{0}(i_{dummy},P^*,P) + v^2\Delta Q^{move}_{1}(i_{dummy},P^*,P) \\
= v^2\Delta Q^{move}_{1}(i_{dummy},P^*,P) \quad [\text{As } i_{dummy} \text{ is isolated when } v = 0]
\]

We now show that \( \Delta Q^{move}_{1}(i_{dummy},P^*,P) \) is always negative.

\[
\Delta Q^{move}_{1}(i_{dummy},P^*,P) = \Delta Q^{merge}_{1}(i_{dummy},P) - \Delta Q^{merge}_{1}(i_{dummy},P^* - i_{dummy}) \\
= \frac{1}{2} \left[ S_0(P,V_0) + S_0(V_0,P) - S_0(P^*,V_0) - S_0(V_0,P^*) \right] \\
\leq 0 \quad [\text{using Eqns (4.9), (4.11)}]
\]

Clearly, the modularity will decrease by moving the dummy node to any cluster with lesser strength. Therefore, the dummy node must belong to the cluster with highest strength.

As the perturbation level increases, existing edges are weakened and new edges are strengthened with the dummy node. It follows that the intra cluster strength decreases for all clusters expect for the strongest cluster. The intra cluster strength of the strongest cluster will remain invariant as it contains the dummy node - all perturbed edges remain within the cluster. This leads us to two interesting properties of the clusters: (a) Splitting the strongest cluster even at higher perturbations will decrease modularity (Property 6) (b) Merging any two weaker (not strongest)
clusters even at higher perturbation levels will decrease modularity (Property 7). In fact, splitting weaker clusters may result in an increase in modularity as the perturbations increase. Note that a combination of operations can still result in an increase in modularity even if one of the operations leads to a decrease in modularity. Therefore, an operation which decreases modularity is not necessarily sub optimal. However, these properties still provide useful insights into the mechanics of alternate cluster detection.

Property 6. Let $P^* \in P_v$ be the strongest cluster detected at perturbation level $v$. Then, splitting $P^*$ at any perturbation level $u \geq v$ will decrease modularity.

Proof. We show that the change in modularity by splitting $P^*$ into $P (\subset P^*)$ and $P^*-P$ is negative. Without loss of generality, we can assume that $i_{\text{dummy}} \notin P$ which implies that $i_{\text{dummy}} \in P^*-P$. Let $\Delta Q^{\text{split}}_u(P, P^*)$ denote the change in modularity due to splitting $P^*$. It follows from Theorem (3) that,

$$\Delta Q^{\text{split}}_u(P, P^*) = (1-u)\Delta Q^{\text{split}}_0(P, P^*) + u^2 \Delta Q^{\text{split}}_1(P, P^*)$$

Before we proceed, we show the following result

$$\Delta Q^{\text{split}}_1(P, P^*) = -\Delta Q^{\text{merge}}_1(P, P^*-P)$$

$$= -\frac{1}{4} \frac{S_0(P, V_0)}{S_0(V_0, V_0)} \left( 1 - \frac{S_0(V_0, P^*-P)}{S_0(V_0, V_0)} \right)$$

$$+ \frac{S_0(V_0, P)}{S_0(V_0, V_0)} \left( 1 - \frac{S_0(P^*-P, V_0)}{S_0(V_0, V_0)} \right) \leq 0$$

There are two possible cases:

1. If $\Delta Q^{\text{split}}_0(P, P^*) \leq 0$, then it is straightforward to show that $\Delta Q^{\text{split}}_u(P, P^*) \leq 0 \ \forall u \in [0, 1]$

2. If $\Delta Q^{\text{split}}_0(P, P^*) > 0$, then

$$\frac{d\Delta Q^{\text{split}}_u(P, P^*)}{du} = -\Delta Q^{\text{split}}_0(P, P^*) + 2u \Delta Q^{\text{split}}_1(P, P^*) \leq 0$$

It follows that for this case, the modularity can only decrease as a result
of split at higher perturbation levels. We already know that $\Delta Q^\text{split}_v \leq 0$, therefore $\Delta Q^\text{split}_u \leq 0 \forall u \in [v, 1]$.

In both cases, the modularity will decrease by splitting the strongest cluster at any perturbation level higher than $v$.

**Property 7.** Let $P^* \in \mathcal{P}_v$ be the strongest cluster detected at perturbation level $v$. Then, merging any two clusters $P \neq P^*, R \neq P^*$ at any perturbation level $u \geq v$ will decrease modularity.

**Proof.** We show that the change in modularity ($\Delta Q^\text{merge}_u(P, R)$) due to merging any two weaker clusters $(P, R)$ is negative. As seen earlier,

$$\Delta Q^\text{merge}_u(P, R) = (1 - u)\Delta Q^\text{merge}_0(P, R) + u^2 \Delta Q^\text{merge}_1(P, R)$$

Before we proceed, we show the following result

$$\Delta Q^\text{merge}_1(P, R) = -\frac{1}{4} \left[ S_0(P, V_0)S_0(V_0, R) + S_0(V_0, P)S_0(R, V_0) \right] \leq 0$$

The above result makes use of the fact that $i_{\text{dummy}} \notin P, R$. The rest of the proof is exactly the same as that of Property (6).

Properties (6,7) hint that the alternative clusterings are a result of the strongest cluster expanding with perturbation level and absorbing vertices from other weaker clusters. In the extreme case ($v = 1$), all vertices are absorbed into a single cluster. Moreover, weaker clusters are less likely to merge but can potentially split into smaller clusters. Therefore, the clusterings detected at higher perturbation levels are not monotonically coarser or finer.

We verify our claims on a toy network with 17 vertices and 38 edges. The network is divided into 4 connected cliques: nodes 1-6 ($C_1$), nodes 7-11 ($C_2$), nodes 12-14 ($C_3$) and nodes 15-17 ($C_4$). Figure 4.3 depicts the toy network and various clusterings solutions found at different levels of perturbation. The clustering solutions were detected by maximizing modularity to optimality using Cplex solver. Clearly, each optimal clustering is optimal for a particular perturbation range (Properties 2,3). Also, the quality of clusterings detected decreases with perturbation level (Property 4). Notice that the strongest cluster ($C_1$) expands with the perturbation level until
it absorbs all nodes in the network (Property 6). Also, none of the weaker clusters merge as the perturbation increases (Property 7). In fact, weaker clusters split into smaller clusters confirming that alternate clusterings are not monotonically coarse or fine. When $v > 0.55$, node 7 changes its cluster membership as it has equal connections to both clusters $C_1$ and $C_2$. Previously, GFeller et al [53] made similar observations and identified such nodes as unstable nodes as they lie at the border of two well defined clusters. Finally, when $v \geq 0.6$, clusters $C_3$ and $C_4$ are correctly split into individual clusters which has a modularity within 10% of the optimal. This indicates that alternate optimal clusterings can capture near-optimal clusterings which may yield stronger insights than just the optimal clustering. Similar results were obtained by Navlakha and Kingsford [70] who explicitly model diversity constraint in modularity maximization. We now apply this methodology to identify alternative clusterings in large scale complex networks.
4.5 Experimentation and Analysis

In the previous section, we have shown the tractability of our approach in theory and verified it on karate club network. Notice that clusterings detected were quite similar to each other. Although minor modifications of partitions may sometimes provide useful information, detecting the optimal clustering for the unperturbed graph suffices for networks like karate club. We will now show that this is not the case in general networks as there exists quite different solutions with near optimal modularity scores. But detecting optimal solutions by solving modularity maximization to optimality requires huge computational resources (even for small networks). Therefore, we have to employ one of the existing modularity maximizing heuristics to handle large graphs which of course cannot guarantee optimality of detected clusterings. Specifically, we use the Louvain algorithm [74] for its speed and accuracy. Unfortunately, the analytical results capturing the dependence of perturbed clustering on perturbation level (Properties 2-4) hold only for optimal solutions. This implies that, for instance, clustering solutions found by heuristics at higher perturbation level may exhibit higher modularity and therefore contradicting property 4. Indeed, such cases did occur during our experiments but the general trend still seems to follow the theoretical predictions. We have experimented on both real and artificial networks. For each network, we detect alternative clusterings \(c_v\) at various perturbation levels, \(v \in [0, 1]\). For each network, we are interested in: (a) Quality \(Q(c_v, G_0)\) and (b) Similarity/Dissimilarity of clustering solutions found at different values of \(v\). The distance between clusterings is captured by an information theory based measure, Variation Information (VI) [38], which measures the fraction of nodes which change their cluster membership w.r.t their true clusters. Let \(X = \{x_1 \ldots x_r\}\) and \(Y = \{y_1 \ldots y_s\}\) be two clusterings with \(r\) and \(s\) clusters respectively. The measure \(VI(X, Y)\) (see Eq.(4.12)) captures the fraction of nodes that have changed their cluster memberships between the two clusterings. It is easy to verify that the variation between two exactly same clusterings is 0. On the other hand, the maximum variation of 1 is achieved when one clustering includes all nodes in a single cluster while the other involves \(n\) different
clusters each containing a single node.

\[ VI(X, Y) = \frac{1}{n \log n} \sum_{x \in X, y \in Y} n_{xy} \log \frac{n_x n_y}{n_x n_y} \tag{4.12} \]

where \( n_x \) is the number of nodes in cluster \( x \) and \( n_{xy} \) is the number of nodes that are common to clusters \( x \) and \( y \).

We expect that for a network with robust clusterings, the perturbations in topology will only result in similar perturbed clustering (small distance). Therefore, we expect to find large dark blocks along the diagonal if the networks have clusters which are robust to perturbations. On the other hand, existence of very different clustering solutions will result in formation of small diagonal blocks. Although the perturbed clusterings are bound to be different at higher perturbation levels, modularity is also expected to decrease. We are specifically interested in cases where the decrease in modularity is not significant and yet the clusterings detected are very different.

4.5.1 Artificial networks

The advantage of testing on artificial networks is that the results obtained can serve as benchmarks for real networks. We use the standard planted community model to generate artificial networks. Suppose there are \( N \) nodes which are divided into equal sized communities of size \( n \). Moreover, any two nodes belonging to the same group will have an edge between them with a probability of \( p \). Similarly, any two nodes belonging to different groups will be connected with a probability \( q \). Therefore, \( q \) can be interpreted as noise level in the network. The values of \( p \) and \( q \) are adjusted so that the average degree of a node (\( \bar{k} \)) remains a constant i.e. \( \bar{k} = (n - 1)p + (N - n)q \). Figures 4.4(a) to 4.4(d) indicate the distance between clusters found at various levels of perturbation when noise level \( q \in \left\{ 0, \frac{\bar{k}}{2(N - n)}, \frac{\bar{k}}{2(N - n)}, \frac{\bar{k}}{N - 1} \right\} \). We generate networks by setting the parameters as \( N = 128, n = 32, \bar{k} = 16 \). When noise levels are low (Figures 4.4(a) and 4.4(b)), big diagonal blocks occur indicating the existence of a clear cluster in the network. The size of the diagonal block decreases as \( q \) increases (Figure 4.4(c)) indicating that the existence of very different clusters with close to optimal modularity val-
ues. Note that at this noise level, the number of expected edges inside a cluster is equal to the number of expected edges outside it. The extreme case of \( q = \frac{k}{N-1} \) results in a random graph with no clear clusters and consequently small diagonal blocks can be observed (Figure 4.4(d)).

### 4.5.2 Real networks

We considered six undirected and unweighted networks for our analysis: Social Networks Jazz Musicians (Nodes = 198, Edges = 5484) [41], Facebook Caltech Students (Nodes = 769, Edges = 16656) [42], Biological Networks - Metabolic Network of C.elegans (Nodes = 453, Edges = 4596) [44], Neural Network of C.elegans (Nodes = 297, Edges = 2359) [75] and Technological networks - Electronic circuit network (Nodes = 512, Edges = 819) [76], Power Grid network (Nodes = 4941, Edges = 6594) [77]. The objective of these experiments is to test the existence of alternative clusterings across various domains. Let us first study the social networks. The social network of jazz musicians is constructed by placing edges between musicians who have played in the same band. Figure 4.5(a) shows that the original clustering of musicians detected is quite robust to perturbations. All other clusterings detected within 90% of optimal modularity are very similar to each other as less than 20% of the vertices have changed their memberships. The robustness of the original clustering is primarily due to a strong racial and geographical divide among musicians in early 20\(^{th}\) century [41]. On the other hand, more diverse alternate clusterings appear in the network of online social network among students at Caltech, Figure 4.5(b). It has already been reported that students are primarily clustered according to their dorm affiliations [42]. However, friendship relationships formed due to commonality between other attributes (field of study, gender and so on) may result in alternate clusters. Similar results are observed for biological networks. The neural network (Figure 4.6(a)) has a single clustering which seems robust to perturbations. This may be due to a coarse initial clustering - nearly 30% of all vertices are grouped in one cluster [75]. In the case of the metabolic network (Figure 4.6(b)), there exists multiple distinct clusterings with similar modularity scores. The alternate clusterings could be a result of cluster members associated with multiple functions. A completely different behavior is exhibited by techno-
logical networks as compared to social and biological networks. Figures 4.7(a), 4.7(b) clearly indicate existence of a single clustering which quickly disintegrates under perturbations. As seen earlier in section 4.5.1, the small blocks along the diagonal is a tell tale sign of a random graph. In summary, the results indicate the existence of alternative clusters in several different real world networks. Moreover, the number and diversity of alternative clusters seems to depend on the network domain.

4.5.3 Case study: Amazon co-purchase network

In order to highlight the utility of detecting alternative clusters, we perform a detailed analysis on a network of co-purchased books. The books are political in nature and were sold at amazon.com during 2004 United States presidential elections. An edge is placed between two books if they have been bought by the same customer. The network data, collected by Valdis Krebs (http://www.orgnet.com), consists of 105 books and 495 edges. The original cluster partitions the network into three groups consisting predominantly of conservative, liberal and neutral books. The categorization of books was originally performed by Newman (http://www-personal.umich.edu/mejn/netdata/) by looking into the metadata (author and reviews available at amazon.com). These three clusters are depicted in Figure 4.8 by solid lines. We apply our methodology of detecting alternative clusters on this data set. When perturbation level $\nu = 0.4$, we find that the original conservative cluster further splits into two additional clusters as indicated by the dashed circles in Figure 4.8. These clusters, although have a conservative outlook, focus on (a) National security and (b) George W Bush’s personal life. The sub-clusters are formed because conservatives are interested in a variety of issues such as social policies, fiscal policies and national security policies. On the other hand, when $\nu = 0.6$, an additional cluster is formed within the liberal cluster. The authors of books in this cluster are well known liberal commentators who have strong views against conservative principles. As with the previous case, the sub-cluster formed indicates a section of liberals who have extreme views against conservatives. A detailed list of books found in all three sub-clusters is shown in Table 4.1. Clearly, the additional clusters provides much deeper insights into the buying patterns of
<table>
<thead>
<tr>
<th>Cluster</th>
<th>Members</th>
</tr>
</thead>
<tbody>
<tr>
<td>G.W. Bush’s Personal Life</td>
<td>The bushes, The perfect wife, The faith of George W Bush</td>
</tr>
<tr>
<td>Extreme Left Wing</td>
<td>Rush limbaugh is a big fat idiot, We’re right they are wrong, It’s still the economy, stupid!, Buck up suck up, Shrub, Had Enough?, Dude, where’s my country?, The best democracy money can buy, Stupid white men, The culture of fear, Downsize this!</td>
</tr>
<tr>
<td>Liberal (Selected Books)</td>
<td>The Price of Loyalty, Plan of Attack, The Sorrows of Empire, All the Shah’s Men, Perfectly Legal, The Great Unraveling, Living History</td>
</tr>
</tbody>
</table>

Table 4.1. Detailed list of books contained in each of the additional clusters detected customers as compared to the original coarse division.

4.6 Conclusions

We have proposed a methodology to identify near optimal clusterings which are also different from each other. A novel deterministic graph perturbation scheme has been proposed which is algorithm independent. A detailed investigation into various real world networks suggests that alternative clusterings at different perturbation levels often yield rich insights into the topology of the network. Future work includes a systematic procedure to combine and represent the knowledge gained from diverse clusterings. Currently, we run the modularity maximization algorithms from scratch on each of the perturbed graphs which may not be feasible in large scale networks. One can improve the computational complexity by using the current clustering as an initial solution for next perturbed graph. The performance of our proposed perturbation scheme on non-modularity based alternative clustering detection will be an interesting exercise.
Figure 4.4. Analysis of perturbed clusters in artificially generated networks under various levels of noise ($q$). For each noise level, alternative clusterings ($c_v$) are detected when $v \in \{0, 0.01, \ldots, 1\}$. A pair of similar (dissimilar) clusterings will result in a dark/light spot in the distance matrix.
Figure 4.5. Analysis of alternative clusters in social networks. The alternative clusterings \( (c_v) \) are detected when \( v \in \{0, 0.01, \ldots, 1\} \). A pair of similar (dissimilar) clusterings will result in a dark (light) spot in the distance matrix.

Figure 4.6. Analysis of alternative clusters in biological networks. The alternative clusterings \( (c_v) \) are detected when \( v \in \{0, 0.01, \ldots, 1\} \). A pair of similar (dissimilar) clusterings will result in a dark (light) spot in the distance matrix.
Figure 4.7. Analysis of alternative clusters in technological networks. The alternative clusterings \((c_v)\) are detected when \(v \in \{0, 0.01, \ldots, 1\}\). A pair of similar (dissimilar) clusterings will result in a dark/light spot in the distance matrix.

Figure 4.8. Alternative clusters in political book network. The shape indicates the political inclination of the book (circle-conservative, square-liberal, triangle-neutral).
Chapter 5

A Game-theoretic approach to Graph Clustering

5.1 Introduction

The interactions in most complex systems can be conveniently captured as graphs with components of the system as vertices. One can then infer several interesting properties about the system by studying the topology of the graph. Formation of clusters in graphs where groups of vertices are densely connected with each other but sparsely connected to the rest of the graph is a phenomenon commonly observed across several domains. There are several useful applications of detecting such clusters. In biological systems, clusters typically correspond to set of biological components (proteins, metabolites, genes) which are responsible for a particular function. The functions performed by any newly identified biological component can be inferred by observing the functions performed by its cluster members. Social networks is another area where graph clustering has tremendous business value. Most of the online human activity, especially on social networking sites, is mainly captured as interactions between people. The clusters in social graphs can identify people with similar interests/preferences. Such information is invaluable for organizations which can develop recommender systems as one can essentially predict customer interests by analyzing the preferences of other closely related customers. There are several other applications in various domains ranging
from wireless service providers to financial services. There has been considerable multi disciplinary research directed at developing algorithms to detect clusters (or communities). However, there has been little agreement in the definition of a cluster. This is understandable as every domain has unique requirements for a group of vertices to become a cluster. There are, however, certain common requirements which have to be satisfied by any definition of a cluster - internal homogeneity and external inhomogeneity\cite{78}. The internal homogeneity refers to the fact that every member of the cluster should be closely connected to the rest of the cluster members. The degree of closeness can vary depending on the domain. The external inhomogeneity requires each member of the cluster to be sparsely connected to members of any other cluster. These natural conditions must form the basis of any clustering algorithm. There is an uncanny resemblance between traditional notions of equilibrium defined in game theory and that of a cluster as discussed above. Consider simple analogy where each vertex is an agent which must decide on a membership cluster and will receive a reward proportional to the number of its neighbors in the cluster. In this context, let us analyze the notion of a Nash Equilibrium (NE). Loosely, NE states that no agent can benefit by unilaterally changing its action. Analogously, each vertex will join a cluster with most neighbors (internal condition) when compared to other clusters (external condition). This relationship has been pointed out in various forms and contexts in the past \cite{79, 80}. We will show that there are several advantages of posing a clustering problem in a game theoretic context. Primarily, we can devise simple algorithms to learn an equilibrium leveraging existing tools from game theory literature. Also, several issues in clustering such as determining optimal number of clusters, overlapping clusters and stability of clusters have natural game theoretic interpretations. Finally, scalability can be vastly improved by devising decentralized algorithms drawing inspiration from the rapidly expanding field of Algorithmic Game Theory \cite{81}. In this paper, we provide evidence on some of the issues that we discussed above.

5.1.1 Our Contributions

We feel the following are the highlights of this paper
• We develop a novel set of conditions to define an overlapping cluster (Sec 5.2)

• Prove the existence of a game whose equilibrium conditions coincide with that of a cluster (Sec 5.3, Sec 5.4)

• Propose two convergent algorithms to compute the equilibrium which have a nearly linear (in the number of edges) running time (Sec 5.5)

• Outperform clique percolation based algorithm [82] for detecting overlapping clusters when tested on artificial networks (Sec 5.7)

5.2 Problem Definition

Consider an undirected graph $G = (V, E)$ where $E$ is the set of edges and $V$ is the set of vertices. In addition, let $N_v = \{w : (v, w) \in E\}$ be the neighborhood of vertex $v$. We represent a clustering $\mathcal{P}$ as a collection of clusters $\{X_i\}_{i=1,...,H}$ where the number of clusters, $H$, is typically unknown. Additionally, we impose a coverage constraint on $\mathcal{P}$ so that all vertices are included in at least one cluster i.e $\bigcup_{X \in \mathcal{P}} X = V$. Note that the clusters need not be disjoint implying that vertices could belong to multiple clusters. As discussed before, we need an internal and external condition to define a cluster. For the sake of simplicity, suppose each vertex can belong to at most one cluster. In order to derive an internal condition, let us define a fitness function $f_v(X)$ as

$$f_v(X) = |N_v \cap X| - \rho|X\setminus\{v\}|,$$  \hspace{1cm} (5.1)

which captures the goodness of membership of vertex $v$ with respect to a cluster $X \in \mathcal{P}$. The higher the fitness, the better connected is the vertex $v$ to other members in $X$. Clearly, the definition of $f_v(X)$ follows directly from that of a $\rho$-quasi clique. A $\rho$-quasi clique [83] is defined as a group of vertices where each element of the group is connected to at least a given fraction ($\rho \in [0, 1]$) of the other members of the group. It is clear that for $\rho = 1$, the definition of a quasi-clique coincides with that of a clique. Note that a quasi clique is less strict when compared to related clique relaxation concepts ($k$-plex, $k$-core), especially for large
subgraphs. This property is useful when the range of cluster sizes is large which is often the case in several real world networks. An intuitive internal condition for a cluster is

\[ f_v(X) \geq 0 \quad \forall v \in X \]  

(5.2)

In other words, each cluster must necessarily be at least as connected as a quasi-clique to guarantee that it is a cluster. We now modify the internal condition to account for overlapping clusters. To that effect, we introduce a new variable called total overlap of vertex \( v \), \( o_v(\mathcal{P}) \), which is simply the total number of clusters in which vertex \( v \) has a membership i.e

\[ o_v(\mathcal{P}) = |\{X \in \mathcal{P} : v \in X\}| \]

As all vertices are covered by at least one cluster, each vertex will have a minimum overlap of 1. The modified internal condition should be more strict for overlapping nodes as compared to non-overlapping nodes. Therefore, the modified internal condition is defined as

\[ f_v(X) \geq \lambda_v(o_v(\mathcal{P}) - 1) \quad v \in X, \forall X \in \mathcal{P} \]  

(5.3)

where parameter \( \lambda_v > 0 \) can be interpreted as the minimum additional fitness required for a vertex \( v \) to belong to one additional cluster. We can adjust its value to control the degree of overlap of vertices. For example, when \( \lambda_v = \infty \), a clustering can satisfy internal condition only if it contains disjoint clusters i.e \( o_v(\mathcal{P}) = 1 \). Now we have to specify an external condition which will ensure sparse connectivity of cluster members to the rest of the network. A natural external condition is that each vertex should have higher fitness with respect to its membership clusters as compared to clusters in which it is not a member i.e.

\[ f_v(X') \leq f_v(X) \quad \forall X, X' \in \mathcal{P} \text{ s.t } v \in X, v \notin X'. \]  

(5.4)

We define the conditions (5.3) and (5.4) together are both necessary and sufficient for any clustering \( \mathcal{P} \) to be a clustering. We can formally define a clustering according to Definition 4.
Definition 4. For a given $\rho, \lambda, \forall v \in V$, a clustering $P$ of graph $G$ is valid if it satisfies an internal condition, Eq.(5.3), and an external condition, Eq.(5.4).

We deal with the problem of detecting a set of clusters (possibly overlapping) in the graph $G$. It is worth while to note the potential non-uniqueness of clusterings. Also, we do not explicitly place any restrictions on the size or number of clusters to be detected.

5.3 Clustering Game

We introduce a game theoretic approach to compute valid graph clustering. We show that a natural connection exists between internal and external conditions of a cluster and equilibrium conditions of the proposed game. More specifically, our objective is to design a game such that the NE of the game will correspond to the clusters of the graph. Consider a non-cooperative clustering game, $\Gamma = (V, A, [U_v]_{v \in V})$, where each player is a vertex in graph $G$. Each vertex is supposed to choose a set of labels which will ultimately determine its memberships in various clusters. The utility functions of vertices will be defined such that the set of vertices which belong to a cluster will choose the same labels. Let $L$ be the set of infinitely countable labels, $L = \{l_1, l_2, \ldots\}$. The infiniteness of $L$ is due to the fact that the number of clusters is unknown before hand. As we want to allow overlapping clusters, each vertex is allowed to choose an arbitrary non-empty subset of labels from the set $L$. Therefore, a feasible action set $A$ for any vertex is the power set of $L$ i.e $A = 2^L \setminus \{\}$. Let $a_v \in A$ be the action chosen by vertex $v$.

Suppose vertex $v$ chooses an arbitrary subset of labels from label set $L$ i.e. $a_v \subseteq L$. The utility of such an action consists of two components - reward and cost. The reward, $r_v(l; a_{-v})$, gained by vertex $v$ by choosing label $l \in a_v$ is equal to the number of its neighbors who have chosen the same label minus the discounted total number of vertices which have chosen label $l$ i.e.

$$r_v(l; a_{-v}) = \sum_{w \in N_v} |l \cap a_w| - \rho \sum_{w \in V \setminus v} |l \cap a_w| \quad (5.5)$$

It is not hard to notice a correspondence between the fitness function and the reward. On the other hand, the cost of choosing additional labels accounts for the
loss of fitness due to overlap. The cost is a quadratic function of the cardinality of set \( a_v \). In specific, we choose the cost function \( c_v(a_v) \) as follows

\[
c_v(a_v) = \frac{1}{2} \lambda_v |a_v|(|a_v| - 1)
\]

The utility of vertex \( v \), \( U_v \), is obtained by combining reward and cost as shown in Equation 5.6.

\[
U_v(a_v; a_{-v}) = \sum_{l \in a_v} r_{vl}(l; a_{-v}) - c_v(a_v)
\]  

(5.6)

We can now define the NE of the clustering game which corresponds to action profile \( a^* = \{a_1^*, \ldots, a_n^*\} \) satisfying equation 5.7.

\[
U_v(a_v^*; a_{-v}^*) \geq U_v(a_v; a_{-v}^*) \quad \forall a_v \in A \quad \forall v \in V
\]  

(5.7)

The next section discusses various properties of the clustering game and its relation to the original graph clustering problem.

### 5.4 Properties of clustering game

In this section, we show (a) clustering game has certain desirable properties which guarantees the existence of a pure NE and (b) relationship between equilibrium and a valid graph clustering. To that effect, we first show that the clustering game is also an exact potential game. A potential game is defined as shown in Definition 5.

**Definition 5.** Consider a game played by agents in set \( V \), each having an action set \( A \) and utility functions \( [U_v : A \rightarrow \mathbb{R}]_{v \in V} \). This game is an exact potential game if \( \exists \Phi : A^{|V|} \rightarrow \mathbb{R} \) such that \( \forall a_v, a_v' \in A \) and \( a_v \neq a_v' \)

\[
\Phi(a_v, a_{-v}) - \Phi(a_v', a_{-v}) = U_v(a_v, a_{-v}) - U_v(a_v', a_{-v}) \quad \forall v \in V
\]

We now show that our clustering game also has an exact upper bounded potential function which is just the sum of all utility function minus the cost function.
Lemma 1. The clustering game $\Gamma$ has an exact potential function

$$\Phi(a) = \frac{1}{2} \sum_{v \in V} [U_v(a_v, a_{-v}) - c_v(a_v)].$$

Moreover, the potential function is bounded from above.

Proof. The proof follows directly from the definition of potential game (Definition 5). The utility function can be written as

$$U_v(a_v, a_{-v}) = \sum_{w \in N_v} |a_v \cap a_w| - \rho \sum_{w \in V \setminus v} |a_v \cap a_w| - c_v(a_v).$$

Therefore,

$$\Phi(a_v, a_{-v}) - \Phi(a'_v, a_{-v}) = \frac{1}{2} [U_v(a_v, a_{-v}) - U_v(a'_v, a_{-v})] - \frac{1}{2} [c_v(a_v) - c_v(a'_v)]$$

$$+ \frac{1}{2} \sum_{w: v \in N_w} (|a_v \cap a_w| - |a'_v \cap a_w|) - \rho \frac{1}{2} \sum_{w \in V \setminus v} (|a_v \cap a_w| - |a'_v \cap a_w|)$$

Due to the symmetric nature of the graph, the third term of the above expression can be rewritten as

$$\sum_{w: v \in N_w} (|a_v \cap a_w| - |a'_v \cap a_w|) = \sum_{w: \in N_v} (|a_v \cap a_w| - |a'_v \cap a_w|)$$

Based on this observation, the last three terms of the above expression can be combined into $\frac{1}{2} (U_v(a_v, a_{-v}) - U_v(a'_v, a_{-v})).$

Now we show that the potential function upper bounded. In order to show boundedness of potential function, it is sufficient to show that the utility functions are bounded. Now consider the utility function of vertex $v$,

$$U_v(a_v; a_{-v}) = \sum_{w: v \in N_w} |a_v \cap a_w| - \rho \sum_{w: v \notin w} |a_v \cap a_w| - \frac{\lambda_v}{2} |a_v|(|a_v| - 1)$$

$$\leq \sum_{w: v \in N_v} |a_v \cap a_w| - \frac{\lambda_v}{2} |a_v|(|a_v| - 1)$$

$$\leq |N_v| |a_v| - \frac{\lambda_v}{2} |a_v|(|a_v| - 1)$$

$$\leq \left(\frac{|N_v| + \frac{\lambda_v}{2}}{2\lambda_v}\right)^2 \text{[finite for any } \lambda_v \in (0, \infty)]$$
Note that the existence of the potential function relies heavily on the symmetric interactions between agents i.e. undirected nature of the graph. The existence of an exact potential function offers several advantages. Primarily, the existence of a pure NE is guaranteed for a upper bounded potential game [84] even if the strategy sets are infinite. Another property of potential games is that they are dominance solvable i.e. several learning dynamics (Ex: Best Response Dynamics, Fictitious play) will converge to a pure NE. In this paper, we are specifically interested in analyzing best response (BR) dynamics. In order to compute the BR, every vertex must maximize its utility function (Eq.(5.8)) assuming every other agent has frozen its strategy i.e.

$$\max_{a \in A} U_v(a; a_{-v})$$

(5.8)

More about the BR dynamics and its convergence is discussed in the next section. We now show necessary and sufficient optimality conditions for an action to be the optimal BR. These conditions will form the basis for a simple algorithm (see Algorithm 3) which can efficiently compute the best response.

**Lemma 2.** For any given action profile \((a_v, a_{-v})\), the necessary and sufficient conditions for an action \(a_v^{BR}\) to be a best response solution of vertex \(v\) are

\[
\begin{align*}
    r_v(l) &\geq \lambda_v(|a_v^{BR}| - 1) \quad \forall l \in a_v^{BR} \\
    r_v(l') &< \lambda_v|a_v^{BR}| \quad \forall l' \notin a_v^{BR} \\
    r_v(l) &\geq r_v(l') \quad \forall l \in a_v^{BR}, l' \notin a_v^{BR}
\end{align*}
\]

(5.9) (5.10) (5.11)

**Proof.** (Conditions are necessary)

We will prove by contradiction for each of the three cases:

- Suppose \(\exists l \in a_v^{BR} : r_v(l) < \lambda_v(|a_v^{BR}| - 1)\). Then, deleting \(l\) from \(a_v^{BR}\) will strictly improve utility i.e

$$U_v(a_v^{BR}\setminus \{l\}; a_{-v}) - U_v(a_v^{BR}; a_{-v}) = \lambda_v(|a_v^{BR}| - 1) - r_v(l) > 0$$

Therefore, contradiction.
• Suppose $\exists l \notin a_v^{BR} : r_v(l) \geq \lambda_v |a_v^{BR}|$. Then, adding $l$ to $a_v^{BR}$ will improve utility i.e

$$U_v(a_v^{BR} \cup \{l\}; a_{-v}) - U_v(a_v^{BR}; a_{-v}) = -\lambda_v |a_v^{BR}| + r_v(l) \geq 0$$

Therefore, contradiction.

• Suppose $\exists l \in a_v^{BR}, l' \notin a_v^{BR} : r_v(l) < r_v(l')$. Then, adding $l'$ and deleting $l$ from $a_v^{BR}$ will strictly improve utility i.e

$$U_v(a_v^{BR} \backslash \{l\} \cup \{l'\}; a_{-v}) - U_v(a_v^{BR}; a_{-v}) = r_v(l') - r_v(l) > 0$$

Therefore, contradiction.

(Conditions are sufficient)

We will again prove by contradiction. Suppose $a_v^{BR}$ satisfies conditions (5.9)-(5.11) but is not optimal. Therefore, there must exist an optimal solution $a_v \neq a_v^{BR}$ such that $U_v(a_v; a_{-v}) > U_v(a_v^{BR}; a_{-v})$. As $a_v$ is optimal, it must also satisfy necessary conditions Eqs (5.9)-(5.11). There are three cases:

• $|a_v| = |a_v^{BR}|$

The optimal solution $a_v$ will contain the $|a_v|$ labels from $L$ with the highest reward (due to necessary condition Eq(5.11)). If there are no ties i.e $r_v(l; a_{-v}) > r_v(l'; a_{-v}) \ \forall l \in a_v, l' \notin a_v$, then $a_v^{BR} = a_v$. On the other hand, if there are ties i.e $\exists l \in a_v, l' \notin a_v : r_v(l) = r_v(l')$, one can construct an alternative optimal solution ($\tilde{a}_v$) by swapping an unselected label with a selected label of equal reward so that $\tilde{a}_v = a_v^{BR}$. Either of the above two cases leads to a contradiction as $a_v^{BR}$ cannot be optimal.

• $|a_v| > |a_v^{BR}|$

Using the same argument as before, one can construct an alternative optimal solution ($\tilde{a}_v$) such that $a_v^{BR} \subset \tilde{a}_v$. It follows that $\exists l \in \tilde{a}_v, l \notin a_v^{BR}$ such that $r_v(l; a_{-v}) \geq \lambda_v(|\tilde{a}_v| - 1)$ (condition 5.9) and $r_v(l; a_{-v}) < \lambda_v |a_v^{BR}|$ (condition 5.10) which is a contradiction as $|\tilde{a}_v| = |a_v| \geq |a_v^{BR}| + 1$.

• $|a_v| < |a_v^{BR}|$

Using the same argument as before, one can construct an alternative optimal
solution ($\tilde{a}_v$) such that $\tilde{a}_v \subseteq a_v^{BR}$. It follows that $\exists l \notin \tilde{a}_v, t \in a_v^{BR}$ such that $r_v(l; a_{-v}) \geq \lambda_v(|a_v^{BR}| - 1)$ (condition 5.9) and $r_v(l; a_{-v}) < \lambda_v|\tilde{a}_v|$ (condition 5.10) which is a contradiction as $|\tilde{a}_v| = |a_v| \leq |a_v^{BR}| - 1$.

There are two steps to the algorithm. Firstly, all the labels which are in user by at least one vertex are sorted in decreasing order of their rewards (labels are ordered according to increasing index incase of a tie). The algorithm exploits the fact that only labels which are currently selected by at least one vertex have a potential of yielding non-negative rewards. Suppose all the used labels $L' = \cup_{w \neq v} a_w$ are ordered such that $r_v(l_1) \geq r_v(l_2) \cdots \geq r_v(l_{|L'|})$. A trivial scenario arises when $r_v(l_1) < 0$. In this case, the optimal best response is to select an arbitrary unused label from $L \setminus L'$ and thereby achieving a maximum utility of 0. Secondly, the vertex selects the sorted labels consecutively until the utility strictly decreases. Algorithm 3 terminates only when all three necessary and sufficient conditions are met - conditions (5.9) and (5.10) are ensured due to termination condition; condition (5.11) is a result of sorting the labels according to decreasing reward. In general, BR need not be unique but one can impose any deterministic tie breaking rule to ensure uniqueness. We will now show that the optimality conditions for

```
Algorithm 3 Optimal Best Response

Input: $a_{-v}, \rho, \lambda_v$
Output: $a_v^{BR}$
Let $L' = \cup_{w \neq v} a_w$ be the set of all used labels
Compute $r_v(l; a_{-v}) \forall l \in L'$ and sort labels in the descending order of rewards using any deterministic tie breaking rule
Set $a_v^{BR} = \{\}$
for $i = 1: |L'|$ do
    if $r_v(l_i; a_{-v}) < \lambda_v(i - 1)$ then
        break
    end if
    $a_v^{BR} \leftarrow a_v^{BR} \cup \{l_i\}$
end for
if $a_v^{BR} = \{\}$ then
    Set $a_v^{BR}$ to any unused label from $L \setminus L'$
end if
```

best response directly translate to internal and external conditions of a valid graph
clustering. In order to establish the relationship between the clustering game and a valid clustering of the original graph, we first define a mapping (Definition 6) from an action profile in the clustering game to a clustering in the original graph.

**Definition 6.** For any action profile \( a \) of the clustering game \( \Gamma \), let \( L' \subseteq L \) be the set of all used labels which is defined as \( L' = \cup_{v \in V} a_v \). Then, a clustering \( \mathcal{P}(a) \) can be constructed by defining a cluster as a collection of vertices with a common label i.e \( X_l(a) = \{ v \in V : l \in a_v \} \forall l \in L' \). In other words, \( \mathcal{P}(a) = \{ X_l(a) \}_{l \in L'} \).

Similarly one can define a reverse mapping (Definition 7) from a clustering to an action profile of the clustering game. In such a mapping, all vertices which belong to the same cluster will share a common label.

**Definition 7.** For any clustering \( \mathcal{P} = \{ X_1, \ldots, X_H \} \) of \( G \), an action profile \( a(\mathcal{P}) \) of the clustering game \( \Gamma \) can be defined as \( a_v(\mathcal{P}) = \{ l \in X_i \} \forall v \in V \)

We show, in Theorem 4, that any equilibrium of the clustering game can be transformed into a valid clustering in graph \( G \). Unfortunately, the converse is not always true i.e given an arbitrary valid clustering of graph \( G \), it may not necessarily correspond to an equilibrium solution of the clustering game. However, we show that a one to one correspondence between equilibrium and valid clusterings exists when clusters are disjoint. A clustering \( \mathcal{P} \) is said to be disjoint if \( X \cap Y = \{ \} \) for any clusters \( X, Y \in \mathcal{P} \).

**Theorem 4.** For any equilibrium action profile \( a^* \) of the clustering game \( \Gamma \), \( \mathcal{P}(a^*) \) (as defined in Definition 6) is a valid clustering of graph \( G \). Moreover, for any valid disjoint clustering \( \mathcal{P} \) of graph \( G \), \( a^*(\mathcal{P}) \) (as defined in Definition 7) is an equilibrium action profile of clustering game \( \Gamma \) with \( \lambda_v = \infty \forall v \in V \).

**Proof.** For the first part of the theorem (equilibrium \( \implies \) valid clustering), we start with two observations

- The fitness of vertex \( v \) w.r.t cluster \( X_l(a^*) \) is the same as the reward component of the utility function, \( r_v(l) \) i.e \( r_v(l) = f_v(X_l(a^*)) \). Also, by construction of the clustering \( \mathcal{P}(a^*) \), we have \( |a_v^*| = o_v(\mathcal{P}(a^*)) \)

- Finally, \( a_v^* \) must be a best response as \( a^* \) equilibrium. Therefore, the necessary conditions Eqns (5.9)-(5.11) must be satisfied by all vertices.
Combining all three observations, we have

\[ f_v(X_l(a^*)) \geq \lambda_v(\circ_v(\mathcal{P}(a^*)) - 1) \quad \forall l : v \in X_l(a^*) \]  

(5.12)

\[ f_v(X_l(a^*)) < \lambda_v(\circ_v(\mathcal{P}(a^*))) \quad \forall l : v \notin X_l(a^*) \]  

(5.13)

\[ f_v(X_l(a^*)) \geq f_v(X_{ll'}(a^*)) \quad \forall (l, l') : v \in X_l(a^*), v \notin X_{ll'}(a^*) \]  

(5.14)

Clearly, Eqns (5.12) and (5.14) are equivalent to the internal and external conditions (Eqns (5.3) and (5.4)) respectively.

For the second part of the theorem (valid disjoint clustering \( \Rightarrow \) equilibrium), we start with two observations

- The fitness of vertex \( v \) w.r.t. cluster \( X_i \in \mathcal{P} \) is the same as the reward component of the utility function, \( r_v(l_i) \) i.e \( f_v(X_i) = r_v(l_i) \). Also, by construction of the action profile \( a^*(\mathcal{P}) \), we have \( \circ_v(\mathcal{P}) = |a_v(\mathcal{P})| = 1 \)

- As \( \mathcal{P} \) is a valid clustering, it must satisfy the internal and external conditions (Definition 4).

Combining all the above observations, we have

\[ r_v(l) \geq 0 \quad \forall l \in a_v(\mathcal{P}) \]  

(5.15)

\[ r_v(l) \geq r_v(l') \quad \forall l \in a_v(\mathcal{P}), l' \notin a_v(\mathcal{P}) \]  

(5.16)

Clearly, the necessary and sufficient conditions, Eq(5.9) and Eq(5.11), are equivalent to Eq(5.15) and Eq(5.16) respectively when \( |a_v| = 1 \). Note that the sufficient condition, Eq(5.10), becomes redundant when \( \lambda_v = \infty \). As all the sufficient conditions are satisfied, \( a_v(\mathcal{P}) \) is a best response for vertex \( v \), given \( a_{-v}(\mathcal{P}) \). Therefore, \( a(\mathcal{P}) \) is an equilibrium action profile.

Note that sometimes it is possible for all nodes in a cluster to choose two different labels resulting in duplicate clusters. Although duplicacy is not a major issue, it might sometimes prevent vertices from joining smaller clusters. One can easily avoid such scenarios by setting the cost of choosing an additional label to a reasonably large value (discussed more in Sec 7.2). It is clear that computing
the NE is the key to discovering clusters in graphs. The next section presents two algorithms which can be used to compute the NE of the clustering game.

## 5.5 Algorithm

Both algorithms proposed in this section primarily use the best response dynamics to compute NE. A BR dynamics involves agents either sequentially or simultaneously updating their actions to best responses. We introduce some additional notation which will be useful in the remainder of the paper. A *round* is said to be complete after each agent gets a chance to play its best response. Let \( a^r \) be the action profile of all agents in \( r^{th} \) round. Also, let \( S_{NE} \) be the set of all pure NE. We know that the \( S_{NE} \neq \emptyset \) as at least one pure NE exists.

### 5.5.1 Sequential Best Response

One of the most intuitive ways to learn an equilibrium is when all agents play best responses sequentially. This is used as the basis of our first algorithm (Algorithm 4) called Sequential Best Response (SEBR). However, such dynamics do not always converge to a NE. Fortunately, in our case the existence of a potential function ensures convergence of BR dynamics (Theorem 5).

---

**Algorithm 4 SEBR: Sequential Best Response**

Input: \( G = (V, E), \rho, \lambda_v, \) vertex ordering rule  
Output: \( a^*_v \forall v \in V \)  
set \( a_v = \{l_v\} \forall v \in V \)  
All vertices are ordered according to vertex ordering rule  
loop  
    Each ordered vertex computes its best response \( a^B_v \) and updates its action  
    \( a_v \leftarrow a^B_v \)  
    If there are no updates then Exit the loop  
end loop  
return \( a^*_v = a_v \)

---

Theorem 5. *(Monderer and Shapley [85]*) Starting from any initial action profile
(a)

(b)

Figure 5.1. (a) Cycling of states in simultaneous best response in a two node network (b) The states and their transition probabilities as a markov chain.

(a\(^0\)), the SEBR algorithm converges to \(S_{NE}\) in finite number of rounds i.e

\[\exists \bar{r} < \infty \text{ s.t } a^\bar{r} \in S_{NE}.\]

This convergence result was observed in the seminal work of Rosenthal [84] and later generalized by Monderer and Shapley [85]. The proof of Theorem 5 uses the fact that each time an agent plays a best response, its utility increases which implies that the potential function increases as well (see Definition 5). But, this improvement cannot occur infinitely often as the potential function is bounded from above. Therefore, one can claim convergence of sequential BR dynamics.

5.5.2 Simultaneous Best Response

We now study a variant of the SEBR (Algorithm 4) which is obtained by updating the best responses simultaneously. We refer to such an algorithm as Simultaneous Best Response (SMBR). A major advantage of SMBR over SEBR is that the former can be easily parallelized. However, this minor variation has adverse effects on the convergence of the best response dynamics. The simultaneous updates may cause cycles in action profiles. For example, consider a simple two node network as shown in Figure 5.1(a) with label set \(L = \{l_1, l_2\}\). Suppose initial labels are assigned such that \(a_v = l_v\) \(v \in \{1, 2\}\). Clearly, the actions of vertices oscillate between \(\{l_1, l_2\}\) and \(\{l_2, l_1\}\). In order to avoid such cyclic behavior, we introduce randomness in the dynamics. We postulate that by randomly forcing agents not to play their best responses we will be able to break the cycle and reach a NE. In
other words, in any given round, with a probability $\epsilon$, an agent $v$ will not play its best response. Based on this idea, we propose the SMBR algorithm (Algorithm 5).

We will now discuss the convergence properties of SMBR. To better understand SMBR algorithm, let us think of the stochastic best response dynamics as a markovian process. Each state of the markov chain refers to an action profile across all agents. See Figure 5.1(b) for an illustration of states and their transition probabilities for the two-node toy example. In general, any state corresponding to a pure NE will be an absorbing state (by definition of NE). Also, we know that at least one pure NE exists (Lemma (1)). Therefore, any markovian process will have at least one absorbing state. A best response cycle corresponds to a closed communicating class (CCC) of states i.e set of communicating states from which the process cannot leave. For instance, in Figure 5.1(b), when $\epsilon = 0$ the states $(l_1, l_2)$ and $(l_2, l_1)$ form a CCC. Interestingly, the CCC disappears for any $\epsilon \in (0, 1)$. Moreover, it is also clear that, after sufficiently large number of rounds, the markovian process will reach an absorbing state (or NE) almost surely. We now show, in Theorem (6), that our observation on the toy network holds true for any general network.

**Theorem 6.** Starting from any initial action profile $(a^0)$, for any $\epsilon \in (0, 1)$, the SMBR algorithm converges to $S_{NE}$ almost surely i.e

$$\lim_{r \to \infty} Pr(a^r \in S_{NE}) = 1$$

**Proof.** The proof relies heavily on the fact that from any state (say $a$), there exists a sequential best response path to some element in $S_{NE}$ (Theorem (5)). On the other hand, for any $\epsilon \in (0, 1)$, we can associate a non-zero probability with the aforementioned sequential best response path starting from $a$. Therefore, there exists a non-zero probability path starting from every state to some absorbing state in $S_{NE}$. This implies that the markov chain itself is absorbing and therefore must get absorbed into $S_{NE}$ in the long run. \( \Box \)

Although SMBR converges in theory, we need a practical termination criterion. Note that non-improvement of action profile does not necessarily mean convergence to NE as agents in non-equilibrium may decide not to be active. The only way to ensure convergence to NE is to compute best responses of all agents which may
prove to be costly if performed very often. Therefore, we wait for \( T = \lfloor \frac{1}{1-\epsilon} \rfloor \) rounds before we check for convergence. The reasoning behind the choice of \( T \) is as follows. During each round, on an average, \(|V|(1 - \epsilon)\) vertices are active. Assuming that there is no overlap of active vertices between two rounds, it will take precisely \( T \) rounds before each agent gets a chance to be active and update its actions. Although the computation of \( T \) involves strong assumptions, it is a good starting point.

\[ \text{Algorithm 5 SMBR: Simultaneous Best Response} \]

**Input:** \( G = (V, E) \), \( \rho, \lambda_v, \epsilon \)

**Output:** \( a_v^* \forall v \in V \)

set \( T = \lfloor \frac{1}{1-\epsilon} \rfloor \)

set \( a_v = \{l_v\} \forall v \in V \)

**loop**
- Set \( V_{\text{active}} \) (active vertices) by randomly selecting vertices from \( V \) with probability \( 1 - \epsilon \)
- All active vertices compute their best responses \( a_v^{BR} \) simultaneously (this operation can be parallelized)
- All active vertices update their actions \( a_v \leftarrow a_v^{BR} \)
  - if no updates for \( T \) consecutive rounds and \( a \in S_{NE} \) then
    - Exit the loop
- end if

end loop

**return** \( a_v^* = a_v \)

5.5.3 Computational Complexity

In order to analyze the computational complexity of both algorithms, we need to know the (a) complexity of computing a best response and (b) number of best responses computed. The former can be analyzed easily as there are three main operations in Algorithm 3. Firstly, note that only labels adopted by neighbors need to be considered as all other labels will carry negative reward. In order to compute the reward, we need total number of neighboring vertices and total number of vertices who have chosen a given label. The former requires \( O(|N_v|) \) operations while the latter is constant time if we maintain and update an label frequency distribution. Secondly, sorting can also be done in log linear time, \( O(|N_v| \log |N_v|) \).
Thirdly, $O(o_v)$ computations are required to choose the right number of labels. Therefore the total complexity of vertex $v$ to compute a BR is $O(|N_v| \log |N_v| + o_v)$. In each round, total BR computation requires $O(|E| \log |V| + |V|\bar{o})$ where $\bar{o}$ is the average overlap of the graph.

The total number of rounds, however, is difficult to analytically estimate in a general setting. However, for a special case of disjoint clusters ($\lambda_v = \infty$) and $\rho = 0$, we can show that the SEBR algorithm converges in at most $|E|$ rounds.

**Lemma 3.** Starting from any initial action profile $(a^0)$ and when $\lambda_v = \infty \ \forall v$, $\rho = 0$, the SEBR algorithm converges to $S_{NE}$ in at most $|E|$ rounds.

**Proof.** Under the specified conditions, it is easy to see that the potential function is given by $\Phi(a) = \frac{1}{2} \sum_v U_v(a_v; a_{-v})$ where $U_v(a_v; a_{-v}) = \sum_{w \in N_v} |a_v \cap a_w|$ and $|a_v| = 1$.

The key observation is that the utility function is integer valued. We already know that in each round, utility function value of at least one vertex must strictly increase. In this case, utility must increase by 1. As the potential is exact, the value of the potential function must also increase at least by 1 each round. Additionally, one can show that

$$\Phi(a) = \sum_{(v,w) \in E} |a_v \cap a_w| \leq \sum_{(v,w) \in E} 1 = |E|$$

As potential function is upper bounded by $|E|$, it will take at most $|E|$ rounds to reach an equilibrium. \qed

In more general settings, we empirically estimate the number of best responses taken by both algorithms when tested on various real world data sets whose sizes range from 34 to 9875. The results are plotted in Figure 5.2.

It appears that the number of best responses required are nearly linear in the number of vertices for both algorithms. We find that the number of best responses required by SMBR is always greater than that of SEBR. Moreover, the difference increases as the network size grows. Our conjecture is that the number of best responses required vary as $O(|V| \log |V|)$ and $O(|V| \log |V|^2)$ for SEBR and SMBR respectively. In other words, we empirically observe that SEBR and SMBR will
reach equilibrium in $\log |V|$ and $\log |V|^2$ rounds respectively which maintains the near linear (in number of edges) complexity.

5.6 Related Work

There have been relatively few game theoretic approaches to graph clustering suggested in the literature. Torsello et al [86] explore the relationship between evolutionary game theory and clustering. They define a clustering game on a graph such that a maximal clique of the graph will correspond to an Evolutionary Stable State (ESS) of the game. They propose replicator dynamics to compute the ESS. Recently, Chen et al [87] propose a community formation game which is similar in spirit to the clustering game defined in this paper. In the community formation game, each agent maximizes a personalized modularity[49] function. They propose an algorithm to detect a local equilibrium which is defined to be a NE with respect to a reduced strategy space. They report impressive results on artificial networks.
when compared to existing algorithms. However, it is not clear how the notion of local equilibrium and their choice of fitness function translates into graph clusters. Also, their motivation for using game theory is to understand community formation itself whereas we are looking at game theory purely from an algorithmic perspective.

There have been several non-game theoretic approaches suggested from both physics and computer science literature to detect overlapping clusters. Perhaps one of the most popular technique for detecting overlapping clusters is the Clique Percolation Method (CPM)[82]. Their algorithm starts by enumerating all $k$-cliques which are cliques of size $k$. Then, a cluster is defined as a collection of adjacent $k$-cliques. Any two $k$-cliques are adjacent if they share $k−1$ vertices. Li et al [88] propose a two-phase algorithm to cluster named entities - people, organizations, etc. The first phase is similar to CPM (with $k = 3$). In the second phase, any remaining cliques are combined based on the content. Another interesting approach based on seed expansions is proposed by Wei et al [89]. The main idea is to compute initial non-overlapping clusters using existing spectral modularity optimization techniques. Then each of seed clusters are expanded in a locally optimal sense leading to overlapping clusters. Note that all the methods described above first detect dense subgraphs and then combine them to obtain overlapping clusters.

On the other hand, one can directly compute overlapping clusters. Zhang et al [90] propose a fuzzy $c$-means modularity maximizing algorithm to detect overlaps in clusters. The central idea is to map each vertex of the graph into a $n_c$-dimensional euclidian space where $n_c$ is the expected number of clusters in the graph. The embedding is done by computing the top $n_c$ eigenvectors of a generalized eigen system $Ax = tDx$ where $A$ is the adjacency matrix, $D = diag(A.1)$ and $t$ is the eigenvalue. They propose a modified modularity measure capable of handling soft clustering. The optimal number of clusters is picked based on the highest modularity score. In another effort, Nepusz et al [91] propose a continuous constrained optimization approach to overlapping community detection. The degree of belonging of vertex $v$ in various clusters is captured by vector $x_v$ and the objective is to minimize $\sum_{(v\neq u)} w_{uv}(a_{uv} - x^T_vx_u)^2$. A gradient descent algorithm is used to solve the quadratic program.
There are another class of algorithms based on Betweenness Centrality (BC)\[92\]. Gregory proposes an algorithm CONGA[93] which is an extension of the BC based clustering algorithm proposed by Girvan and Newman\[94\]. A high BC of edge implies that many shortest paths pass through the edge and that it is an inter cluster edge. In each step of the algorithm either an edge with highest BC is removed or a vertex is split into two based on the vertex BC. A novel concept of a vertex split-BC is defined for each possible way in which a vertex can be split. The split of vertices resulting in highest split-BC is carried out. As split vertices can belong to multiple groups, overlapping clusters are obtained. In order to speed up the computation of split-BC, a heuristic called CONGO[95] is introduced to approximate the best vertex split. Pinney and Westhead [96] also adopt a BC based approach. They remove an edge with highest BC only if both ends of the edge have similar BC values. Otherwise the vertex with the highest BC is temporarily removed until the component splits into multiple subcomponents. After the split, a copy of the temporarily split vertices are added to all the subcomponents. The biggest disadvantage of this class of algorithms is that the number of clusters need to be specified as an input. It is worthwhile to mention that the nature of the potential function (Lemma 1) indicates close relationship of our algorithm to potts model [?, 97] based cluster detection techniques.

5.7 Experimental Results

We now evaluate the performance of SEBR and SMBR algorithms under a variety of parameter settings and artificially generated networks. The advantage of artificial networks because the ground truth (true clusters) is known and strong claims can be made about accuracy in comparison to other algorithms. Performance is defined both in terms of accuracy and computational complexity. In order to measure the the accuracy of an algorithm we compare the overlapping clusters generated by an algorithm to the true clusters using a modified Normalized Mutual Information (NMI) measure [98]. The computational complexity is captured by the total number of best responses which have to be solved to reach an equilibrium.
5.7.1 Generating Artificial Networks

Lancichinetti et al.[99] describe the procedure that can generate overlapping clusters with varying degrees of connectivity and overlap. We will not describe the procedure here but only describe the parameters used to generate the networks. The networks consisting of \( N \) vertices are generated with the average and maximum degree of the graph set to \( k_{\text{avg}} \) and \( k_{\text{max}} \) respectively. Both the node degrees and cluster sizes follow a power law distribution with negative exponents \( \tau_1 \) and \( \tau_2 \). An upper and lower bound can be specified on the size of the clusters (\( s_{\text{min}}, s_{\text{max}} \)). An additional parameter (\( \mu \)) specifies the fraction of edges connected to a vertex which do not belong to any of its clusters. One can interpret \( \mu \) as noise which makes the cluster membership more fuzzy. The number of overlapping nodes (\( \text{on} \)) and their number of cluster memberships (\( \text{om} \)) can also be specified. Each of the \( \text{on} \) overlapping nodes will have the same overlap of \( \text{om} \).

5.7.2 Parameter Selection

There are a total of four parameters involved in SEBR and SMBR - \( \rho, \lambda_v, \epsilon \) and vertex ordering rule. We generate artificial networks and record the performance of both algorithms under a variety of parameter settings.

The value of density parameter \( \rho \) depends on the network properties. For instance, it is known that social networks have denser clusters when compared to communication networks [100] and therefore warrants a higher value of \( \rho \). In general, small values of \( \rho \) lead to bigger and weakly connected clusters. But, larger values of \( \rho \) will result in smaller but strongly connected clusters. Similar story is observed in Figure 5.3(a) where the accuracy of cluster detection is highest in a certain range of values of \( \rho \). It seems that \( \rho \approx \rho^G \) yields good results where \( \rho^G \) is the density of graph \( G \). It has the natural interpretation that clusters must be denser than the overall graph. However, in extremely sparse networks, the clusters are usually orders of magnitude denser than the overall graph. In such cases, setting \( \rho = \rho^G \) might result in poor cluster quality. A guideline for the choice of \( \rho \) is still unclear.

We set \( \lambda_v = \lambda(1 - \rho) \forall v \in V \) so that all vertices incur the same cost of joining multiple clusters. Note that \( \lambda \) has a natural interpretation of minimum number of neighbors that a vertex should have in a cluster before it can decide to join an
Figure 5.3. The effect of parameters \((\rho, \lambda, \epsilon)\) on the computational complexity and the accuracy of SEBR and SMBR. The networks are generated with parameters \((N = 100, k_{\text{avg}} = 10, k_{\text{max}} = 20, s_{\text{min}} = 3, s_{\text{max}} = 20, on = 10, om = 2)\) under five different scenarios of varying noise levels \((\mu)\). Each data point is obtained by averaging over 10 random graph realizations.

additional cluster. It follows that large values of \(\lambda\) will discourage formation of overlapping clusters and small values of \(\lambda\) will result in too many trivial clusters (of size 2 and 3). The accuracy of SEBR is insensitive to the value of \(\lambda\) (see Figure 5.3(b)). We find that \(\lambda = 2\) works well in practice.

Also, we have already established that sequential best responses by vertices will result in convergence to NE. But, the sequence in which players will update their strategies is not clear. We find that the order of best response updates does not affect the SEBR algorithm.

Finally, we want to capture the effect of \(\epsilon\) on both accuracy and computational complexity of SMBR. It turns out that the accuracy of SMBR is not affected by \(\epsilon\) (Figure 5.3(c)) and 5.3(d). Also, the accuracy index exhibits higher variation when the noise levels in the networks are higher. This is because of the existence
of more number of pure NE - some of them being potentially bad. On the other hand, the computational complexity is sensitive to the choice of $\epsilon$. There is drastic increase in the number of best responses for small values of $\epsilon$ which tend to settle down as $\epsilon \to 1$. An explanation for this observation is that the chances of best response cycling is greater when $\epsilon \approx 0$ as there are more players playing the game simultaneously. In summary, $\epsilon$ must be chosen as close to 1 as possible yielding better computational efficiency without affecting the accuracy.

5.7.3 Performance on Artificial Networks

We compare the accuracy of clusters detected by our algorithm (SEBR) to the well known clique percolation method (CPM) when tested on artificially generated networks. In our experiments, we set the values of $(N, k_{avg}, k_{max}, \tau_1, \tau_2, om)$ to $(1000, 20, 100, 2, 1, 2)$ respectively. We vary the number of overlapping nodes from 0 to 0.5$N$ under various settings of $(\mu, s_{min}, s_{max})$ (see Figure 5.4). Both SEBR and SMBR do not differ significantly in any of the four scenarios indicating that both algorithms tend to converge to similar NE. Hence, we can safely conclude that SMBR is preferable over SEBR only if SMBR can be sufficiently parallelized. Based on our conjectures on algorithm complexity (Sec 5.3), then we need at least $O(\log|V|)$ parallel threads for SMBR to be a viable option. Let us now compare the performances of SEBR and CPM. One can see a general dip in the performance of both algorithms as the noise level ($\mu$) is increased which is understandable. Moreover, SEBR performs marginally better than CPM when the range of cluster size is reasonably small ($[s_{min}, s_{max}] = [10, 50]$). But as we increase the range of cluster sizes to $[20, 100]$, performance of CPM degrades drastically. This is mainly due to a rather strong definition of a cluster which is implicit in CPM - a cluster is a collection of highly overlapping cliques. Such a definition is very unlikely to hold when the cluster size increases. On the other hand, performance of SEBR and SMBR is fairly stable and shows significant improvement over CPM.
Figure 5.4. Comparing the accuracy of SEBR, SMBR and Clique Percolation Method (CPM). The parameters of SEBR/SMBR and CPM are set as $(\rho = 0.05, \lambda = 2, \epsilon = 0.99)$ and $(k = 4)$ respectively. Each data point is obtained by averaging over 10 random graph realizations.

5.8 Applications

We now demonstrate the applicability of our approach in Natural Language Processing (NLP) and segmenting World Wide Web (WWW). The data for both experiments can be downloaded from http://www.CFinder.org. As SEBR and SMBR tend to give similar quality solutions, we use only SEBR algorithm for our analysis.

5.8.1 Word Sense Disambiguation

Word sense disambiguation is one of the oldest problems in the area of natural language processing. The basic idea is that words can have different meaning based on the context in which they are used. We use the word association network [101] which is constructed by asking users for the most related word when given a specific word. An edge between two words indicates their frequent co-occurrence. After pre processing, there are a total of 7207 words and 31784 edges. There is a
Figure 5.5. Clusters around the words PLAY and COLD in the word association network. An edge is placed between two words if their total frequency of occurrence is at least 0.025. Parameters used ($\rho, \lambda, c$)=(0.1,2,2) high degree of overlap in such a network as words can used in different senses - each of which will correspond to a cluster. Figure 5.5 shows the different senses of words PLAY (musical instruments, theater, children and toys) and COLD (common cold - disease, common cold - symptoms, weather - hot, weather - cold) which are discovered by SEBR. Also, close to 20% of the total words have a overlap of at least 2 indicating existence of clusters with large overlaps. Based on our analysis, we can safely conclude that SEBR can detect coherent clusters in graphs with
Figure 5.6. This is segmentation of the web pages in Google’s domain. The parameter settings are $(\rho, \lambda, c) = (0.5, 2, 1)$ potentially large overlaps.

5.8.2 Segmenting the World Wide Web

We now turn our attention to detecting clusters in the World Wide Web (WWW). The knowledge of topological patterns such as clusters can help tremendously to improve both speed and accuracy of information retrieval. For this analysis, we use the data collected by Palla et al[102] of all webpages around http://www.google.com. There are a total of 946 vertices and 1817 edges. One would expect that the web pages will correspond to various functionalities provided by google to the end users. Moreover, some of the web pages will belong to multiple functions. For example, the search web page (www.google.com) will belong to multiple functions (accounts, jobs, support) of google as shown in Figure 5.6. Several web pages with large overlaps exist (Ex: sitemap.html, about.html). The existence of such overlapping modules improves information retrieval as one can quickly move from one dense group to the other using the highly overlapping web pages as a bridge. Similar observations have been made by Palla et al[102].

5.9 Conclusion

We have proposed a game theoretic approach to detect overlapping clusters in graphs. We developed two convergent algorithms, SEBR and SMBR, whose com-
plexity seems to be nearly linear in the number of edges. The accuracy of our approach clearly outperforms clique percolation method when tested on artificial networks. There are several future directions of this work. It will be interesting to compare running times of SEBR to a parallel implementation of SMBR algorithm. Another direction of work could be along the non-uniqueness of NE. As there could be multiple NE, can we extract other NE which could provide alternative understanding of the network? The extension of this framework to directed graphs seems non-trivial as the existence of an exact potential function is not guaranteed anymore. Also, the effect of alternative learning dynamics (e.g. fictitious play) and fitness functions will yield interesting insights. Finally, studies are required to gain a more fundamental understanding of the relationship between game theory and clustering.
A major drawback of all existing approaches is the lack of adaptability to different domains. For instance, clusters in biological networks may have a different topological features when compared to clusters in social networks. Although some clustering algorithms have tunable parameters, it is often not clear how they affect the clusters detected. In some cases, parameters are easier to understand but they can only control trivial cluster properties like density. One can overcome this issue by considering an alternative approach to clustering. Suppose each vertex of the input graph has an associated deterministic but unknown label. The objective of clustering is to group vertices which have the same label. An underlying assumption is that the edges in the graph somehow depend on the hidden labels of vertices. Note that if the assumption were not true then edges observed in the graph are completely independent of the vertex labels and we have no hope of detecting true partitions. Suppose we know the rule based on which edges are formed. A simplest edge formation rule can be: *An edge is placed between two vertices if they have the same hidden label.* In this case, the resulting graph will be a collection of isolated cliques where all the members of a clique will have the same label. In order to detect clusters, one just has to detect components of the graph which is trivial. As the complexity of the edge formation rule increases, detecting clusters becomes more difficult. The dependence of edges on labels can be deterministic or stochastic. Clearly, one can then characterize the applicability of existing clustering algorithms based on the edge formation rules. As different domains have different edge formation rules, one can develop custom clustering algorithms which fit the
needs for a particular domain. This approach is in direct contrast with all of the existing approaches which rely on quality functions. It is assumed that maximizing existing quality functions like modularity and conductance are assumed to predict hidden labels independent of the domain being modeled. Another advantage of this approach is that it allows for elegant analytical treatment. Most of the existing quality functions do not allow for strong analytical treatment as obtaining algorithms with provable error bounds seems quite impossible. The questions which need to be answered are: Are there any edge formation rules for which cluster detection is impossible? Can we provide approximation guarantees for a class of edge formation rules? Another major aspect of clustering is the parallelizability of algorithms. As the size of the datasets grows, it becomes more and more important for algorithms to leverage ever increasing computational resources at hand. It can be extremely valuable to study existing clustering algorithms and modify them for parallel execution. Also, stability considerations are extremely important as noisy data is becoming common. The proposed algorithms must output clusters which are insensitive to small perturbations in network topology. In this thesis, we considered edge deletions as the only source of perturbations. However, one can consider vertex deletions, edge additions and degree preserving edge rewiring as additional perturbation sources. In summary, the intersection of techniques from operations research and data mining can potentially give rise to interesting modeling and methodological insights for researchers in both communities.
Bibliography


URL http://stacks.iop.org/1742-5468/2007/i=06/a=P06016

URL http://adsabs.harvard.edu/abs/2010PhR...486...75F


URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4358966


URL http://pre.aps.org/abstract/PRE/v70/i2/e025101

URL http://www.pnas.org/cgi/content/abstract/103/23/8577

URL http://www.citeulike.org/group/1476/article/1105595

URL http://portal.acm.org/citation.cfm?id=42512.42513

URL http://arxiv.org/abs/0709.2938
URL http://pre.aps.org/abstract/PRE/v69/i6/e066133

URL http://pre.aps.org/abstract/PRE/v70/i6/e066111

URL http://pre.aps.org/abstract/PRE/v74/i3/e036104

URL http://www.computer.org/portal/web/csdl doi/10.1109/ICDM.2007.72

URL http://stacks.iop.org/1742-5468/2005/i=02/a=P02001

URL http://pre.aps.org/abstract/PRE/v72/i2/e027104

URL http://arxiv.org/abs/0711.0491

URL http://www.springerlink.com/content/d0p361972ggg76465/

URL http://stacks.iop.org/1742-5468/2005/i=09/a=P09008


100

URL http://www.citeulike.org/user/filippone/article/883684

URL http://www.springerlink.com/content/p693214864j078v2/

URL http://portal.acm.org/citation.cfm?id=1232959.1233220


URL http://www.jstor.org/stable/3629752


URL http://arxiv.org/abs/0809.0690v1

URL http://pre.aps.org/abstract/PRE/v67/i2/e026112

URL http://dx.doi.org/10.1038/35036627

URL http://pre.aps.org/abstract/PRE/v68/i6/e065103


URL http://www.springerlink.com/index/10.1140/epjb/e2008-00425-1


URL http://arxiv.org/abs/0902.3331

URL http://stacks.iop.org/1742-5468/2008/i=10/a=P10008?key=crossref.46968f6ec61eb8f907a760be1c5ace52

URL http://dx.plos.org/10.1371/journal.pone.0009240

URL http://link.aps.org/doi/10.1103/PhysRevE.64.046119


URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=293490

URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=745730


URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4761386


URL http://dx.doi.org/10.1007/11871637_60


pp. 483–490.
URL http://linkinghub.elsevier.com/retrieve/pii/S0378437106008119

URL http://arxiv.org/abs/0707.1646

URL http://www.jstor.org/stable/3033543

URL http://www.springerlink.com/index/w670414828512227.pdf

URL http://arxiv.org/abs/cond-mat/0112110

URL http://www.springerlink.com/index/488601182072pm66.pdf


URL http://arxiv.org/abs/cond-mat/0402349

URL http://stacks.iop.org/1367-2630/11/i=3/a=033015

URL http://arxiv.org/abs/0904.3940


Vita
Supreet Reddy Mandala

EDUCATION

- Pennsylvania State University (Aug 06 Aug 12)
  PhD, Industrial Engineering and Operations Research (3.94/4.0)
  Thesis: Scalable and robust algorithms for clustering large scale networks

- Indian Institute of Technology, Bombay, India (Aug 01 - Aug 06)
  BS and MS, Mechanical Engineering (Dual Degree) (8.12/10.0)
  Thesis: Reinforcement learning algorithms for autonomous robot navigation

INDUSTRY EXPERIENCE

- IBM T. J. Watson Research Center, Research Intern (May 11 Aug 11)
- Intelligent Automation Inc. (IAI), Research Intern (May 07 Aug 07)

ACADEMIC EXPERIENCE

- Laboratory of Intelligent Systems and Quality, Research Assistant
- Department of Industrial Engineering, Teaching Assistant

RELEVANT COURSES


AWARDS HONORS

- Nominated for the INFORMS 2011 Future Practitioners Doctoral colloquium
- Recipient of best paper award in the Computers and Information Systems track of Industrial Engineering Research Conference (IERC, June 2009)
- Invited Member of International Golden Key International Honor Society

SELECTED PUBLICATIONS

- Mandala, S., Kumara, S., Yao, T., Detecting robust community structures using Robust Optimization Industrial Engineering Research Conference, Miami, June 2009 (Best Paper Award)
• Mandala, S., Kumara, S., Yao, T., Detecting alternative graph clusters
  Physics Review E (under review)

• Mandala, S., Kumara, S., Chatterjee, K., A game theoretic approach to
  graph clustering INFORMS Journal on Computing (submitted)

• Ben-Tal, A., Chung, B., Mandala, S., Yao T., Robust Optimization for Emer-
  gency Logistics Planning: Risk Mitigation in Humanitarian relief Supply
  Chains Transportation Research Part B: Methodological, Volume 45, Issue
  8, September 2011, Pages 1177-1189

• Yao T., Mandala S., Chung, B., Evacuation Transportation Planning Un-
  der Uncertainty: A Robust Optimization Approach Networks and Spatial
  Economics, Vol. 9, Issue 2, 171-189, 2009

• Ben-Tal, Mandala, S., Yao T., Robust Linear Programs for Evacuation under

SELECTED TALKS

• A game-theoretic approach to graph clustering, Mathematical Optimization:
  Theory and Applications (MOPTA), Lehigh University, August 2011

• Detecting robust community structures using Robust Optimization Indus-
  trial Engineering Research Conference (IERC), Miami, June 2009

PATENTS
Hernandez, M., Laredo, J., Mandala, S., Ping, Y.,Vukovic, M., A method for
hybrid role mining (Filed for U.S Patent)