NETWORK BASED APPROACHES FOR CLUSTERING AND LOCATION DECISIONS

A Dissertation

by

ANURAG VERMA

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

August 2012

Major Subject: Industrial Engineering

NETWORK BASED APPROACHES FOR CLUSTERING AND LOCATION DECISIONS

A Dissertation

by

ANURAG VERMA

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

August 2012

Major Subject: Industrial Engineering

ABSTRACT

Network Based Approaches for Clustering and Location Decisions. (August 2012)

Anurag Verma, B.Tech., IIT Madras; M.E., Texas A&M University

Co-Chairs of Advisory Committee: Dr. Sergiy Butenko
Dr. Gary M. Gaukler

The objective of this dissertation is to study commonly occurring location and clustering problems on graphs. The dissertation is presented as a collection of results in topics including finding maximum cliques in large graphs, graph clustering in large scale graphs, determining location of facilities for pre-positioning emergency relief supplies, and selecting nodes to form a virtual backbone in a wireless sensor network.

To begin with, a new clique relaxation called a *k-community* is defined as a connected subgraph such that endpoints of every edge have at least *k* common neighbors within the subgraph. It is used to develop scale reduction techniques to obtain the maximum clique on very large scale real life networks. Analytically, the technique is been shown to be very effective on power-law random graphs. Experimental results on real life graph instances (Collaboration networks, P2P networks, Social networks, etc) show our procedure to be much more effective than a regular *k*-core peeling approach.

Next, a general purpose network clustering algorithm based on the clique relaxation concept of *k*-community is presented. A salient feature of this approach is that it does not use any prior information about the structure of the network. By defining a cluster as a *k*-community, the proposed algorithm aims to provide a clustering of a network into *k*-communities with varying values of *k*. Even though the algorithm is not designed to optimize any particular performance measure, the computational results suggest that it performs well on a number of criteria that are used in literature to evaluate the quality of a clustering.

The third topic deals with choosing the locations of disaster response facilities for the storage of emergency supplies, which is critical to the quality of service provided in a large scale emergency like an earthquake. In the existing literature, large scale emergency facility location models have either assumed that disaster response facilities will always be functioning and available when required, or that the functioning of a facility is independent of a particular disaster scenario. In this paper new location models are presented that explicitly take into consideration the stochastic nature of the impact a disaster can have on the disaster response facilities and the population centers in surrounding areas. A comparison of the results obtained using our models with those from models available in literature using a case study suggests that the locations suggested by the model in this paper significantly reduce the expected cost of transportation of supplies when we consider the damage a disaster causes to the disaster response facilities and areas near it.

Lastly, a distributed approximate algorithm for forming the communication backbone in wireless sensor networks is presented. Some of the most popular routing protocols for wireless sensor networks require a virtual backbone for efficient communication between the sensors. Connected Dominating Sets (CDS) have been studied as a method of choosing nodes to be in the backbone. The traditional approach is to assume that the transmission range of each node is given and then minimize the number of nodes in the CDS representing the backbone. A recently introduced alternative strategy is based on the concept of k-bottleneck connected dominating set (k-BCDS), which, given a positive integer k, minimizes the transmission range of the nodes that ensures a CDS of size k exists in the network. This paper provides a 6-approximate distributed algorithm for the k-BCDS problem. The results of empirical evaluation of the proposed algorithm are also included.

DEDICATION

Dedicated to my sister, parents and GS.

ACKNOWLEDGMENTS

I would like to thank my advisors, Dr. Butenko & Dr. Gaukler, for the guidance, knowledge, and discussions they shared with me during the last five years. I cannot think of having better people and scholars to have as advisors. I would also like to thank the faculty of the department for the learning opportunities they availed to me. I thank the staff members in the department, especially Judy, for their constant help and support. I am grateful to the graduate student community in the department for their wonderful support and camaraderie throughout. Finally, I would like to thank Preeti, whose constant support ensured that I stay motivated and on course.

TABLE OF CONTENTS

LIST OF TABLES

TABLE Page

LIST OF FIGURES

1. INTRODUCTION

A graph, denoted by $G = (V, E)$, is defined as a set of nodes or vertices V, and a set of edges $E \subset V \times V$ representing the pairs of vertices that are connected. Graphs can be used to represent information in a very concise manner based on pairwise relationships between entities. In the course of this dissertation, this generic mathematical concept, and the tools built around it are used for representing systems – both abstract and physical – and solving some problems of interest.

The problems studied in this dissertation are classified into two sets – location and clustering. In location problems, the nodes in the graph represent a physical location. Consequently, geographical position is an important physical attribute of a node, often determining the presence and weight of an edge connecting two nodes. Selecting a subset of nodes to perform a task in effect amounts to choosing locations for a particular service from a given set of possibilities. On the other hand, in clustering problems the physical location of nodes is irrelevant, and the relationships a node has with other nodes, represented by edges, are of paramount importance. The task at hand is to find large subsets of nodes that are highly interconnected with each other.

The tools used for solving these problems vary depending on the needs of the problem and the computational resources available. For example, one of the topics addressed in this dissertation involves a long range planning problem of finding good locations for placing facilities with large quantities of disaster relief supplies. Considering the stochastic nature of the problem and the permanence of these facilities, a practical approach would be to model the system as well and in as much detail as possible and to apply appropriate tools irrespective of the computational requirements, as long as they can be solved in a few days. On the other extreme, another chapter addresses the problem of choosing nodes to be a part of a virtual communication backbone in ad-hoc wireless sensor networks in real-time. Each sensor is represented by a node, and their limited processing power calls

This dissertation follows the style of *IIE Transactions*.

for a distributed approach that requires very little computationally from each sensor. In this case, a practical approach is to forsake optimality for ease of computation. Although the tools used vary from computationally intensive stochastic programming and branch & bound to computationally inexpensive approximation algorithms, they all fall under the umbrella of optimization.

The overarching focus of this dissertation is to model systems as networks and to develop and apply relevant algorithms to obtain a solution to the problem being studied. In the following paragraphs, some of the systems and application under consideration are highlighted. Apart from providing a background for the chapters to follow, each of which is devoted to different topics under study, the following paragraphs also present a way of thinking of systems as networks. It is evident that networks capture the most basic underlying structure of a system where bilateral interactions are of principal importance.

1.1 Graph-Based Data Mining

Network science is an emerging field that studies network representations of data sets generated by an underlying complex system in order to draw meaningful conclusions regarding the system's properties. In a network representation of a complex system, the network's nodes typically denote the system's entities, while the edges between nodes represent a certain kind of similarity or relationship between the entities.

Social networks immediately come to mind – each person is represented by a node, and edges represent friendships. Although made popular by the online social networks like Facebook and LinkedIn all around us today, social network analysis has been used by anthropologists and social scientists for many decades. The concise representation of a complex social system as a network, and the various mathematical structures borrowed from graph theory allowed for the kind of quantitative analysis that would not possible otherwise. For example, the criticality of certain individuals and friendships to a social organization has been studied by using cut vertices and cut edges. The theory of "six degrees of separation" borrows from the idea of the diameter of a graph, while the concept of cliques has played a major role in measuring the cohesion within a social group. Studies on structural properties of networks have resulted in models for predicting the dissemination of information and the spread of epidemics in the society. The usefulness of network analysis, however, is in no way limited to social sciences. Many science and engineering disciplines use networks analysis for gaining insights and better understanding of the overall system. Consider the following network examples that have been conceptualized and used in the past:

Web graphs/Link graphs: Web graphs are made by considering html web pages as nodes, and connecting a node to another if the corresponding html page contains a hyperlink to the other one. In one of the more famous examples, web graphs were used by the founders of the Google search engine. The search engine uses web graphs to apply the PageRank algorithm [Page et al., 1999] to rank web pages based on their importance.

Protein interaction networks: Protein interaction networks are an intuitive way of mapping which proteins bind with which other proteins to form a biologically functional complex [Matsunaga et al., 2009]. Each protein is represented by a node, and an edge connects proteins that form a bind with each other. These networks are studied in the bio-informatics community to relate biological functions of different proteins.

Molecular correspondence graphs: Modeling molecules as three dimensional structures, researchers in molecular biology identify pairs of molecules with similar structure using graph isomorphism. To this end, a molecular correspondence graph is formed with nodes as pairs of atoms (one from each molecule being matched), and “nodes are joined by an edge if the interatomic distances between the pairs of atoms are the same within a user-defined tolerance” [Gardiner et al., 1998]. The size of the maximum clique in these correspondence graphs is a measure of the structural similarity of the two graphs. This technique has been found useful in drug discovery to identify alternative chemicals with similar structural properties.

Over the last two decades, the advent of information technology has resulted in an abundance of data. As a result, the networks that researchers in science and technology encounter today have hundreds of millions of nodes, far surpassing anything that was available before that. Improved computational resources have been key to coping with the increase in the scale of graphs available today, but not without advances in graph theory and algorithms. As a result of research in computer science and operations research, improved algorithms for famously hard problems such as graph partitioning, graph clustering, maximum independent set, vertex coloring, maximum clique, and traveling salesman have been found. The first part of this dissertation focusses on the maximum clique and graph clustering problems.

The term *clique* originates from Old French cliquer ("make a noise"), and, according to modern dictionaries, describes an exclusive group of people with a common purpose. The graph-theoretic concept of a clique, which was first introduced in the social science literature, aims to model "tightly knit" groups of elements, every pair of which share a common attribute, or are closely related in some clearly defined sense. This concept has played a key role in many important developments in diverse scientific fields, both theoretical and applied, including graph theory, computer science, mathematical optimization, operations research, and social science.

Not surprisingly, the most common modeling function of cliques is to represent clusters (cohesive groups, modules, communities) in various applied settings. If the studied entities in a real-life system of interest are described as vertices (nodes) and their pairwise associations are expressed as edges (links, arcs) between vertices, then cliques in this graph-theoretic representation of the system naturally represent clusters of entities. In general, one may be interested in analyzing all (large) clusters; however, in some cases detecting one largest clique in the graph is sufficient. The size of such a clique provides an upper bound on the number of elements in a cluster and can be thought of as a global characterization of cohesiveness of the system the graph represents.

The maximum clique problem, which aims to find a clique of the largest size in a graph, is a classical combinatorial optimization problem that has been extensively studied from different perspectives during the last several decades. Its extensive list of applications includes social network analysis, computational biology, study of human brain, coding theory, and finance, among others. Numerous exact and heuristic algorithms have been proposed for solving this computationally hard problem and successfully applied in practice. The maximum clique problem has a rich presence in a number of other mathematical fields. Although a fairly straightforward problem to state, it has resulted in a number of significant developments not just in graph theory, but also in integer programming, complexity theory, and global optimization, to name a few.

The importance of graph clustering in various applications has spawned a new area of clique relaxations, that are developed by relaxing some properties of a clique such as density and connectivity. Clique relaxations overcome the restrictive nature of cliques and help in identifying large clusters that may not be cliques, but will fit the definition of a cluster. The first two chapters in this dissertation will be devoted to the maximum clique problem and graph clustering, respectively.

1.2 Locating Disaster Response Facilities

Recent years have brought forth a number of devastating large-scale emergency situations, such as Hurricanes Rita and Katrina that affected the United States as well as portions of the Carribbean in 2005, or the earthquakes in Sichuan, China in 2008, and in Haiti in 2010. In the United States, this has prompted policy makers at the federal and at the state level to coordinate the establishment of disaster response facilities. At these facilities, supplies (food rations, medical items, rescue equipment) are stored for use in the event of a large-scale emergency. An example of this pre-positioning strategy is the creation of the Strategic National Stockpile (SNS) [Cdc, Last accessed: May 2011]. A crucial aspect of such pre-positioning strategies is the judicious choice of the pre-positioning sites, which is a facility location problem.

Many facility location models assume an underlying network structure to represent infrastructure. In a network representation, cities, potential facility locations and other points of interest become nodes, whereas the roads are represented by edges. This representation works well with the facility location problems because in most cases, the desired locations are close to the existing transportation infrastructure. Furthermore, the network representation allows for the use of existing algorithms and systems for solving the models.

An important consideration that makes emergency facility location significantly different from warehouse location, which has been studied extensively, is the dependence of the demand and availability of a facility on a particular disaster scenario. Under normal circumstances, there is no demand for emergency supplies. It is the occurrence of a disaster that generates a localized demand, while simultaneously reducing the supplying capabilities of facilities in that area. For example, it is quite possible for a facility that lies near the epicenter of an earthquake or on the path of the eye of a hurricane to be severely affected and not provide services when its immediate neighborhood requires them. This structure makes the problem very different from conventional facility location, where we usually consider all demands to be constantly active, and all facilities to be functioning at their full capacity. Modeling a disaster response facility location problem as a conventional facility location problem could result in a scenario where a facility is damaged and unable to serve precisely when it is required the most.

1.3 Virtual Backbone in Wireless Sensor Networks

Wireless sensor networks have applications in domains like military and environmental tracking to obtain information through the use of sensors. The primary tasks that each sensor in a network performs – which are critical to the usefulness of the sensor network – are collection and dissemination of information. The limited battery power supply in sensors curtails the usefulness of the sensor network by reducing its lifespan, and reducing the energy consumption of the sensors is a major challenge.

Communication between sensors, although a critical task, is a major drain on the battery of a sensor. Many sensor networks aggregate information collected from different sensors and communicate it to an external terminal where decisions can be made by processing the information obtained. A common strategy used in sensor networks for communication is to designate some nodes to perform communication/coordination tasks while the others can specialize in collection of information. For example, one set of nodes might be delegated the task of communicating with the external terminal, while another set might be given the task of facilitating communication between the sensors primarily collecting information. The set of nodes that facilitate communication are often collectively called a virtual communication backbone. The choice of nodes to be in the virtual backbone is important to the lifespan of the network.

Graphs provide a succinct representation of wireless sensor networks, with each node representing a sensor, and two nodes being connected if the corresponding sensors are within each other's transmission range. This representation facilitates the analysis of the performance of a given virtual backbone, and also allows us to specify the requirements from a virtual backbone more clearly.

1.4 Conclusion

To summarize, this dissertation comprises of topics in graph based data mining, wireless sensor networks, and locating disaster response facilities. The proposed method of research consists of developing new models, developing computationally efficient algorithms and evaluating the effectiveness of the new models. The diverse set of problems addressed result in a variety of solution algorithms and improvements – approximation algorithms, heuristics, scale reduction, branch & bound, and the L-shaped method – being developed and used. Chapter 2 will focus on a scale reduction technique for the maximum clique problem in large scale graphs, Chapter 3 provides a graph clustering algorithm using clique relaxations, Chapter 4 presents a new model and solution technique for the location of emergency response facilities, Chapter 5 develops an approximation algorithm for finding the virtual backbone in a wireless sensor network, and finally, Chapter 6 concludes the dissertation with a discussion on future research avenues.

2. MAXIMUM CLIQUE PROBLEM ON VERY LARGE SCALE SPARSE NETWORKS

2.1 Introduction

A graph, denoted by $G = (V,E)$, is defined as a set of vertices V, and a set of edges $E \subset V \times V$ representing the pairs of vertices that are connected. Graphs can be used to represent information in a very concise manner based on pairwise relationships between entities. Graph based data mining refers to obtaining information from a graph that could be pertinent to a particular task. Its applications have been explored in marketing and e-commerce for developing recommender systems, computational biochemistry & genomics, social network analysis, clustering, the analysis of financial markets and many other network science applications. For example, consider the network that could be built by an on-line retailer like Amazon or Overstock. Each node represents a buyer, and two nodes are connected if the corresponding buyers have purchased at least p items in common or visited pages of similar products. Finding dense clusters of nodes in this network can provide useful insight into the purchasing habits of buyers, and could lead to the development of better recommender systems.

A clique, defined as a subset of vertices such that all the vertices in it are pairwise adjacent, is a graph theoretical concept often used to represent these dense clusters. Cliques are critical components of graph-based data mining approaches for analysis of call graphs, social networks, communication networks, www graphs and collaboration networks amongst other examples. In particular, the application of maximum cliques has been explored in marketing for devising cross selling strategies [Cavique, 2007], computational biochemistry and data mining in genomics [Butenko and Wilhelm, 2006, Raymond and Willett, 2002, Matsunaga et al., 2009], analysis of financial markets [Boginski et al., 2005], protein structure alignment [Strickland et al., 2005] and CAD/VLSI [Corno et al., 1995].

The maximum clique problem is to find a clique of maximum cardinality in the given graph. It is a celebrated problem that has captured the attention of many researchers in computer science and operations research. In fact, it is one of Karp's original problems shown to be NP-complete.

The maximum clique problem has received significant attention from researchers in developing both exact methods like branch & bound, enumerative and scale reduction approaches [Carraghan and Pardalos, 1990, Corno et al., 1995, Balas and Xue, 1996, Wood, 1997, **?**, Tomita and Seki, 2003, Butenko and Trukhanov, 2007] and heuristics [Gendreau et al., 1993, Protasi et al., 1995, Abello et al., 1999, Katayama et al., 2005], amongst others. However, none of the exact methods have been thoroughly tested for graphs with more than 10,000 vertices. These methods have been well tested for small graphs ($n < 10,000$), especially graphs from the 2nd DIMACS implementation challenge [DIMACS2, Last accessed: November 2011]. On the other hand, the heuristic algorithms developed are not guaranteed to provide the maximum clique sizes.

Recent advances in information technology has resulted in data sets that are much larger than what most exact algorithms have been tested on. In fact, many real life networks of interest are very large, with tens of millions of nodes, and have extremely low densities, with the degrees of nodes often following a power law distribution. In this research, we devise an exact method to find the maximum clique in very large sparse graphs. The method has been tested on graphs with up to 18 million vertices originating from some real life graph instances. Abello et al. [1999] have attempted solving a maximum clique problem on a large real life network (AT&T Call graph with 53 million nodes, the authors could not obtain the dataset). However, they provide a heuristic solution with no guarantee of optimality. Some other researchers tackle the problem of enumerating all the maximal cliques of large scale graphs [Modani et al., 2010]. Though in the process they do find the maximum clique for large graphs, the fact that they are enumerating all other maximal cliques results in a large amount of time being spent before the maximum clique is found. Furthermore, we provide a theoretical analysis of some aspects of the algorithm when applied to power law random graphs.

2.2 k-Community

In this section, we define a new structure called k-community on an undirected graph $G = (V,E)$ and study some of its cohesiveness properties that will help us establish it as a clique relaxation. Before defining a k-community, we need the following two preliminary definitions.

Definition 2.2.1 (Neighbor of an edge)**.** *A node $t \in V$ is a neighbor of an edge $(u,v) \in E$ if it is connected to both u and v, i.e., $(v,t) \in E$ and $(u,t) \in E$.*

Definition 2.2.2 (Edge induced subgraph)**.** *An edge induced subgraph is a subset of edges of a graph G together with all the incident vertices.*

We are now ready to define a k-community, which can be seen as an edge analogue of the k-core as follows.

Definition 2.2.3 (k-Community)**.** *A k-Community of a graph G is the set of nodes in the edge induced subgraph with each edge having at least k neighboring vertices in the subgraph.*

This definition of a k-community is analogous to an edge based k-core (A k-core is a set of nodes such that in its induced subgraph each node has at least k neighboring nodes). In simpler words, a k-community of a graph G is the set of nodes in a subgraph such that the end-points of each edge have at least k common neighbors. Note that while a k-community of G will be a $(k+1)$-core of G, the converse is not true. Thus, the k-community is a strictly stronger clique relaxation when compared to a k-core.

As is the case with the k-core, the k-community of a graph can be found in polynomial time. Algorithm 1 describes a simple iterative procedure for finding the k-community of a graph G. In the first iteration, the algorithm removes all the edges from the graph that have less than k neighboring nodes. If no edges were removed, we have found the k-community of the graph. If some edges were in fact removed, we do another iteration as the connectivity of the nodes and edges could have changed.

This algorithm can be implemented more efficiently by examining an edge only if one of its neighboring edges was removed in the previous iteration. This limits the number of times the number of neighbors of an edge is calculated to $(2\Delta+1)$ times, Δ being the highest degree of a node in the graph. The overall complexity of finding the k-community can be shown to be $O(mk\Delta)$. A brief outline of the proof is as follows: suppose m_1 edges get deleted in the first iteration. Then it is $O(m_1\Delta)$ to delete those edges, and $O(m_1k\Delta)$ to investigate new ones. The second term arises because $O(\Delta)$ to investigate one edge, and there are at most $2m_1k$ edges to be investigated since each of the removed edges affects at most $2k$ edges. Similarly, suppose m_2 edges get deleted in the next iteration. Then it is $O(m_2\Delta)$ to delete those edges, and $O(m_2k\Delta)$ to investigate new ones. Similarly define m_3, m_4 and so on. Since $\sum_i m_i \leq m$, for any given k, the algorithm is $O(mk\Delta)$.

Algorithm 1 $k-$`Community`(G): Algorithm to Find k-Community of G

```
1: repeat
2:   for every $(i,j) \in E$ do
3:     if $|N(i) \cap N(j)| < k$ then
4:       $E \leftarrow E \setminus \{(i,j)\}$
5:     end if
6:   end for
7: until No edge is removed in the current iteration
8: $G(V_k,E_k) \leftarrow G[E]$      /* Edge induced subgraph */
9: return $V_k$
```

Some properties of the k-community that are worth noting for their usefulness in later sections are provided below.

Property 2.2.4. *A clique of size k is a $(k-t)$-community for any $t \geq 2$.*

Property 2.2.5. *If the k-community of G is empty, then size of maximum clique is $\leq k+1$.*

Note that the converse of property 2.2.5 is not true, for if it was, we could solve the maximum clique problem in polynomial time. The following result establishes the some cohesiveness properties of k-communities.

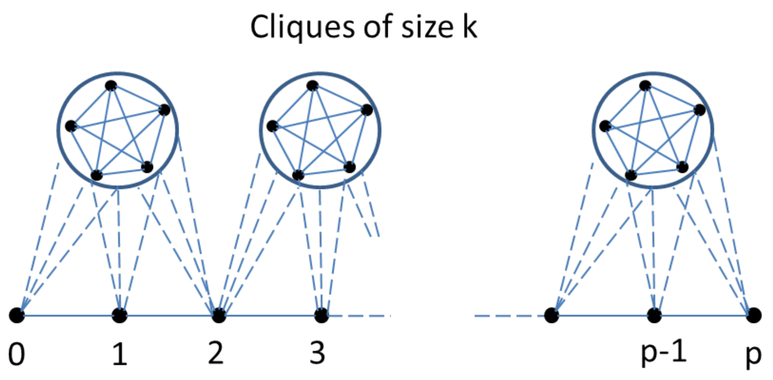


Fig. 2.1.: k-Community with $p(k+2)/2+1$ nodes and diameter p.

Theorem 2.2.6 (Cohesiveness properties of k-communities)**.** *A k-community S satisfies the following conditions:*

1. *The diameter of $G[S]$ is at most $\lfloor \frac{2(n-1)}{k+2} \rfloor$.*

2. *The minimum degree $\delta(G[S]) \geq k+1$.*

3. *The density of $G[S]$ is greater than $\frac{k+1}{|S|-1}$.*

Proof. For (a) let the diameter of a given k-community with n nodes be p. There exist vertices u,v in the k-community such that the shortest path between u and v is of length p. Let $u = x_0, x_1, ..., x_p = v$ be that shortest path. Note that each edge (x_i, x_{i+1}), $i = 0, ..., (p-1)$ should have at least k neighbors. Also, since $x_0, x_1, ..., x_p$ is the shortest path from x_0 to x_p, the edges (x_i, x_{i+1}) and (x_j, x_{j+1}) cannot have any common neighbors if $|i-j| \geq 2$. Hence, every alternate edge in the diameter should have at least k unique neighboring nodes. Thus, the number of nodes in the graph $n \geq (p+1) + k(\lceil \frac{p}{2} \rceil) \geq (p+1) + k(\frac{p}{2}) = 1 + p\frac{k+2}{2}$. Hence, $p \leq \lfloor \frac{2(n-1)}{k+2} \rfloor$.
Statement (b) is trivial, and (c) directly follows from (b). □

Figure 2.1 shows that the bound on the diameter is tight.

2.3 Scale Reduction Approaches

In this section, we devise a scale reduction method based on k-communities to find the maximum clique in a given graph. By scale reduction, we mean removal of vertices and edges from the graph which we are certain are not a part of the maximum clique, thus reducing the size of the graph we have to work with. An exact algorithm, like branch and bound, can then be applied on the smaller reduced graph. The most common scale reduction technique is peeling, where given a lower bound ω_{lb} on the clique number, we find the $(\omega_{lb}-1)$-core of the graph.

Algorithms 2 and 3, which are discussed next make use of Property 2.2.5 to find an upper bound of the clique number, while 4 uses Property 2.2.4 to find the maximum clique.

Algorithm 2 describes a simple binary search strategy for finding an upper bound ω_{ub} of the clique number. The binary search is performed in the interval $[\omega_{lb}-2, \Delta-2]$, where ω_{lb} is the size of a clique found by a greedy strategy (hence a lower bound on the maximum clique size), and Δ is a weak upper bound on the maximum clique size. The algorithm finds the smallest integer k' such that k'-community(G) is an empty set. By Property 2.2.5, $(k'+1)$ is an upper bound on the clique number. The worst case complexity of the algorithm is $O(m\Delta^2 log(\Delta))$. Since finding the k-community modifies the edge set, the whole edge set has to be duplicated from the original graph or a previously computed $(\omega_{lb}-2)$-community after each iteration of the while loop. While this does not affect the complexity, it can be a time consuming step for large graphs that take large amounts of memory.

As a result, Algorithm 3 is proposed as a linear search strategy that starts with $k = \omega_{lb}-2$, and increments k till the upper bound is found. Since k is incrementing, any edge that was removed in finding k-community will also be removed in finding $(k+1)$-community, and so we do not have to duplicate the whole graph in each iteration. Although the worst case complexity of this algorithm is $O(m\Delta^3)$, it might be a faster algorithm for large graphs because of lesser overhead cost of memory operations.

Algorithm 2 `UpperBoundKCommBinary(G,`ω_{lb}`,`Δ`)`: Binary Search Algorithm for ω_{ub}

```
1:  $k^u \leftarrow \Delta - 2,\ k^l \leftarrow \omega_{lb} - 2,\ k \leftarrow k^u$
2:  while $k^u - k^l > 1$ do
3:      if $k -$ Community(G) is empty then
4:          $k^u \leftarrow k$
5:      else
6:          $k^l \leftarrow k$
7:      end if
8:      if $k^u - k^l \leq 1$ then
9:          $k \leftarrow k^l$
10:     else
11:         $k = (k^u + k^l)/2$
12:     end if
13: end while
14: Return $k + 2$
```

Algorithm 3 `UpperBoundKCommLinear(G,`ω_{lb}`)` : Linear Search Algorithm for ω_{ub}

```
1:  $k \leftarrow \omega_{lb} - 2$
2:  while $k -$ Community$(G)$ is non-empty do
3:      $k \leftarrow k + 1$
4:  end while
5:  Return $k + 1$
```

The overall procedure for finding the maximum clique is described in Algorithm 4. To begin with, a greedy algorithm is employed to obtain a lower bound ω_{lb}. Next, either Algorithm 2 or 3 is used to obtain an upper bound ω_{ub} on the clique size. Subsequently, the $(\omega_{ub} - 2)$-community(G) is found and a lower bound ω_{lb} on the clique size is then obtained by using the procedure `FindMaxCliqueExact`, which can be any exact algorithm developed for finding the maximum cliques on smaller graphs. For this chapter, we use the algorithm developed by **?** as the exact algorithm. It should be noted that Property 2.2.5 does not guarantee that the maximum clique is in the $(\omega_{ub} - 2)$-community(G) because it might have been pruned away. This is true since if $\omega_{ub} - \omega_{lb} > 1$, there is a possibility that a clique of size $\omega \in [\omega_{lb} + 1, \omega_{ub} - 1]$ exists. Hence, we go a step further and find the $(\omega_{lb}-2)$-community(G), which is guaranteed to have the maximum clique. Finally, we either use an exact algorithm [**?**] on $(\omega_{lb}-2)$-community(G), or if the number of nodes is higher than what the exact algorithm can take, find the connected components of $(\omega_{lb}-2)$-community(G) and use the exact algorithm on each component to obtain the maximum clique of the graph G.

As mentioned earlier, steps 2 and 4 of Algorithm 4 use the maximum clique algorithm developed by **?**. We could as well have used any of the algorithms developed in [Butenko and Trukhanov, 2007, **?**, Carraghan and Pardalos, 1990, Corno et al., 1995, Wood, 1997, Tomita and Seki, 2003, Balas and Xue, 1996]. A commercial solver, CPLEX 11 was also tried but performed worse than [**?**]. It should be noted that none of these algorithms have been tested extensively on more than $10,000$ vertices, and hence cannot be used directly on the graphs with sizes that we are going to test our algorithms on. Abello et al. [1999] have attempted using a similar peeling approach using k-cores instead of k-communities, but could not reduce the size of the graph enough to employ an exact algorithm.

2.4 Extensions of k-Community

We can extend the idea of k-communities to obtain stronger clique relaxations. In each of the following definitions, consider a simple undirected graph $G=(V,E)$.

Definition 2.4.1. *A set of vertices $V' \subseteq V$ is said to be a vertex-neighborhood-k-core if for each vertex $v \in V'$ the subgraph $G[N(v)]$ contains a (nonempty) k-core.*

Definition 2.4.2. *A set of vertices $V' \subseteq V$ is said to be a vertex-neighborhood-k-community if for each vertex $v \in V'$ the subgraph $G[N(v)]$ contains a (nonempty) k-community.*

Definition 2.4.3. *Consider a set of edges $E' \subseteq E$ such that for each edge $e \in E'$ the subgraph $G[N(e)]$ induced by the edge-neighborhood $N(e)$ contains a (nonempty) k-core. Then the set of vertices $V' \subseteq V$ from the edge-induced subgraph $G[E']$ is said to be an edge-neighborhood-k-core.*

Algorithm 4 `FindClique`(G): Algorithm to find maximum clique of G

```
1: $\omega_{lb} \leftarrow$ GreedyClique$(G)$
2: if $(\omega_{lb} - 2) -$ Community$(G)$ has $< 12000$ vertices then
3:     return $\omega \leftarrow$ FindMaxCliqueExact$((\omega_{lb} - 2)-$Community$(G))$
4: end if
5: $\omega_{ub} \leftarrow$ UpperBound$(G, \omega_{lb}, \Delta)$
6: if $\omega_{lb} < \omega_{ub}$ then
7:     $\omega_{lb} \leftarrow \max(\omega_{lb},$ FindMaxCliqueExact$((\omega_{ub} - 2) -$ Community$(G)))$
8: end if
9: if $\omega_{lb} < \omega_{ub}$ then
10:    if $(\omega_{lb} - 2) -$ Community$(G)$ has $< 12000$ vertices then
11:        $\omega \leftarrow$ FindMaxCliqueExact$((\omega_{lb} - 2) -$ Community$(G))$
12:    else
13:        Find the connected components $\mathfrak{C}$ in $(\omega_{lb} - 2) -$ Community$(G)$
14:        for each connected component $C \in \mathfrak{C}$ do
15:            $\omega \leftarrow \max(\omega,$ FindMaxCliqueExact$(G[C]))$
16:        end for
17:    end if
18: else
19:    $\omega \leftarrow \omega_{lb}$
20: end if
```

Definition 2.4.4. *Consider a set of edges $E' \subseteq E$ such that for each edge $e \in E'$ the subgraph $G[N(e)]$ induced by the edge-neighborhood $N(e)$ contains a (nonempty) k-community. Then the set of vertices $V' \subseteq V$ from the edge-induced subgraph $G[E']$ is said to be an edge-neighborhood-k-community.*

These enhancements can be used to obtain both tighter upper bounds and better scale reduction using the lower bound of the clique size.

2.5 *k*-Community in Power-Law Random Graphs

A graph G is called a power law graph if the the number of nodes with degree q is proportional to $q^{-\alpha}$, where $\alpha \in (1,3)$ is a constant. Power-law graphs are ubiquitous in nature, and many graphs studied in literature have been found to follow this structure.

Shen et al. [2011] show that the maximum clique cannot be approximated within any constant factor on large power-law graphs unless NP=ZPP. In this section we characterize the quality of the upper bound obtained by algorithm 2 for large power-law random graphs. For this purpose, we use the hidden variable ensemble model for generating random power law graphs [Bianconi and Marsili, 2006]:

1. Assign a hidden continuous variable q_i to each node i according to the power law distribution.

2. Each pair of nodes with hidden variables q and q' are linked with probability

$$r(q,q') = \frac{qq'}{\bar{q}n} \tag{2.5.1}$$

   where $\bar{q}$ is the expectation of q, equivalently the average degree.

To ensure that the linking probabilities $r(q,q')$ are less than 1, we introduce a cutoff $Q = \sqrt{\bar{q}n}$ on the power law distribution. Hence, the hidden variable distribution is as follows:

$$p(q) = \begin{cases} p_0 q^{-\alpha} & q \in [1,Q] \\ 0 & \text{otherwise} \end{cases} \tag{2.5.2}$$

Furthermore, the cutoff Q can be estimated as $Q \sim n^{1/\alpha}, \alpha \in (1,2]$ and $Q \sim n^{1/2}, \alpha \in (2,3)$. It has been shown by Bianconi and Marsili [2006] that,

$$\Omega(n^{1/3\alpha}) = \omega = O(n^{1/2\alpha}), \qquad \alpha \in [1,2) \tag{2.5.3}$$

$$\Omega(n^{(3-\alpha)/4}) = \omega = O(n^{(3-\alpha)/6}), \qquad \alpha \in (2,3) \tag{2.5.4}$$

With this model in mind, in the rest of this section we establish some asymptotic results that hold true with high probability – that is, the probability converges to 1 as the number of vertices in the random graph n goes to infinity.

Lemma 2.5.1. *For a power law random graph with exponent $\alpha \in [2,3)$ obtained using the hidden variable ensemble, the upper bound ω_{ub} of the clique number obtained by algorithm 2 is $O(n^{(3-\alpha)/2})$ with high probability. Furthermore, $\omega_{ub} = O(\omega^3)$ for $\alpha \in (2,3)$.*

Proof. Let E_k denote the set of edges in the k-community of G, and E_k^t denote the set of edges remaining after t iterations of removing edges with less than k neighboring nodes. Further, let G_k^t denote the subgraph $G[E_k^t]$ induced by the set of edges E_k^t. We proceed by showing that $|E_k| \longrightarrow 0$ with high probability for $k > \Theta(n^{(3-\alpha)/2})$.

Consider the probability that nodes i & j with hidden variables q_i & q_j are both connected to node m with hidden variable q_m in G:

$$\begin{aligned} w_n(i,j,q_m) = P((i,m) \in E,\ (j,m) \in E) = r(q_i,q_m)r(q_j,q_m) \quad &(2.5.5)\\ = \frac{q_i q_m}{\bar{q}n}\frac{q_i q_m}{\bar{q}n} \quad &(2.5.6)\\ = \frac{q_i q_j}{\bar{q}^2 n^2} q_m^2 \quad &(2.5.7)\end{aligned}$$

Probability that i & j are both connected to a randomly chosen node in G:

$$\begin{aligned} w_n(i,j) = \int_1^Q p(q_m) w(i,j,q_m) dq_m \quad &(2.5.8)\\ = \int_1^{\sqrt{n}} p_0 q_m^{-\alpha} \frac{q_i q_j}{\bar{q}^2 n^2} q_m^2 dq_m \quad &(2.5.9)\\ \simeq \frac{p_0 q_i q_j n^{-(\alpha+1)/2}}{3-\alpha} \quad &(2.5.10)\end{aligned}$$

Let $\eta(i,j)$ denote the number of common neighbors that i & j have in G. We can show that

$$\mathbb{E}[\eta(i,j)] = \mu_{\eta(i,j)} = \frac{p_0 q_i q_j n^{(1-\alpha)/2}}{3-\alpha} \tag{2.5.11}$$

$$\mathbb{V}ar[\eta(i,j)] = \sigma^2_{\eta(i,j)} \simeq \frac{p_0 q_i q_j n^{(1-\alpha)/2}}{3-\alpha} \tag{2.5.12}$$

Now, we can observe the following using the one-sided Chebyshev's inequality,

$$P(\eta(i,j) \geq k) \leq \begin{cases} \frac{\sigma^2}{(k-\mu)^2} & k > \mu_\eta \\ 1 & k < \mu_\eta \end{cases} \tag{2.5.13}$$

Note that since $q_i, q_j \in [1, \sqrt{n}]$, when $k > \Theta(n^{(3-\alpha)/2})$, we have $k \gg \mu_\eta$. Thus,

$$P(\eta(i,j) \geq k) \leq \frac{\sigma^2}{k^2} \tag{2.5.14}$$

Now consider the expected number of neighbors node i will have in the graph G_k^1 as $n \longrightarrow \infty$.

$$\mathbb{E}[|N_{G_k^1}(i)|] = \int_1^{\sqrt{n}} np(q_j) r(q_i,q_j) P(\eta(i,j) \geq k) dq_j \tag{2.5.15}$$

$$\leq \int_1^{\sqrt{n}} np_0 q_j^{-\alpha} \frac{q_i q_j}{\bar{q}n} \frac{\sigma_\eta^2}{k^2} dq_j \tag{2.5.16}$$

$$\leq \frac{n^{(1-\alpha)/2} p_0 q_i^2}{(3-\alpha)k^2} n^{(3-\alpha)/2} \tag{2.5.17}$$

$$\leq c \frac{q_i^2 n^{2-\alpha}}{k^2} = O(1) \tag{2.5.18}$$

since $k > \Theta(n^{(3-\alpha)/2})$ and $q_i^2 \leq n$, $\forall i$. Similarly, the variance of the number of neighbors can be found to be

$$\mathbb{V}ar[|N_{G_k^1}(i)|] \leq c\frac{q_i^2 n^{2-\alpha}}{k^2}(1 - \frac{q_i^2 n^{1-\alpha}}{k^2}) = O(1), \tag{2.5.19}$$

Thus, using the one-sided Chebyshev's inequality, we can claim that as $n \longrightarrow \infty$,

$$P(N_{G_k^1}(i) \geq k) \longrightarrow 0 \tag{2.5.20}$$

and further that

$$P(N_{G_k^2}(i) = 0) \longrightarrow 1 \tag{2.5.21}$$

Thus, with high probability all the edges in G_k will be deleted, leaving the k-community empty for any $k > \Theta(n^{(3-\alpha)/2})$.

Hence, the upper bound ω_{ub} is $O(n^{(3-\alpha)/2})$ with probability tending to one as $n \longrightarrow \infty$. From equation 2.5.3, we can deduce that $\omega_{ub} = O(\omega^3)$ with high probability. □

Lemma 2.5.2. *For a power law random graph with exponent $\alpha \in (1,2)$ obtained using the hidden variable ensemble, the upper bound ω_{ub} of the clique number obtained by algorithm 2 is $O(n^{1/\alpha})$ with high probability. Furthermore, $\omega_{ub} = O(\omega^3)$ for $\alpha \in (1,2)$.*

Proof. We proceed in a similar manner as lemma 2.5.1. Using the same notation, we can show that

$$w_n(i,j) \simeq \frac{p_0 q_i q_j n^{3/\alpha - 1}}{(3-\alpha)(\bar{q}n)^2} \tag{2.5.22}$$

$$\simeq \frac{p_0 q_i q_j n^{-1/\alpha - 1}}{(3-\alpha)} \tag{2.5.23}$$

and

$$\mathbb{E}[\eta(i,j)] = \mu_{\eta(i,j)} = \frac{p_0 q_i q_j n^{-1/\alpha}}{3-\alpha} \tag{2.5.24}$$

$$\mathbb{V}ar[\eta(i,j)] = \sigma^2_{\eta(i,j)} = \frac{p_0 q_i q_j n^{-1/\alpha}}{3-\alpha} \tag{2.5.25}$$

Note since $q_i, q_j \in [1, n^{1/\alpha}]$, when $k > \Theta(n^{1/\alpha})$, $k \gg \mu_\eta$, and we have

$$P(\eta(i,j) \geq k) \leq \frac{\sigma^2}{k^2} \tag{2.5.26}$$

Now consider the expected number of neighbors node i will have in the graph G^1_k as $n \longrightarrow \infty$.

$$\mathbb{E}[|N_{G^1_k}(i)|] = \int_1^{n^{1/\alpha}} np(q_j) r(q_i, q_j) P(\eta(i,j) > k) dq_j \tag{2.5.27}$$

$$\leq \int_1^{n^{1/\alpha}} \frac{n^{1-3/\alpha} p_0^2 q_i^2}{(3-\alpha)k^2} q_j^{2-\alpha} dq_j \tag{2.5.28}$$

$$\leq \frac{n^{1-3/\alpha} p_0^2 q_i^2}{(3-\alpha)^2 k^2} n^{3/\alpha - 1} \tag{2.5.29}$$

$$\leq c\frac{q_i^2}{k^2} = O(1) \tag{2.5.30}$$

since $k > \Theta(n^{1/\alpha})$ and $q_i < n^{1/\alpha}$. Finding $\mathbb{E}[|N_{G^1_k}(i)|]$ & $\mathbb{V}ar[|N_{G^1_k}(i)|]$ and using the one-sided Chebyshev's inequality as in the proof of Lemma 2.5.1, we can deduce that all the edges in G_k will be deleted, leaving the k-community of G empty for any $k > \Theta(n^{1/\alpha})$.

Thus, the upper bound we obtain, ω_{ub} is $O(n^{1/\alpha})$. We obtain $\omega_{ub} = O(\omega^3)$ by using equation 2.5.4. □

Theorem 2.5.1. *Given a power law random graph with coefficient* $\alpha \in (1,2) \cup (2,3)$*, the upper bound* ω_{ub} *on the maximum clique obtained by algorithm 1 is* $O(\omega^3)$ *with high probability.*

Proof. The theorem follows directly from lemmas 2.5.1 and 2.5.2. □

2.6 Computational Results

The test cases were obtained from the Stanford Large Network Dataset Collection [SNAP, Last accessed: November 2011], referred to as the SNAP, and the 10th DIMACS implementation challenge [DIMACS10, Last accessed: May 2012]. These databases have a collection of large network datasets from tens of thousands of nodes and edges to tens of millions of nodes and edges. They includes social networks, web graphs, road networks, internet networks, citation networks, collaboration networks, random geometric graphs, and communication networks. The multitude of domains these networks originate from, along with the very large sizes of the networks make the two datasets a suitable candidate for performing computational studies for our algorithm. For conciseness, we consider all the networks that have at least 30,000 vertices and a few with lesser number of nodes. The networks in the database that were directed graphs were converted to an undirected graph by considering the directed edges as undirected. Table 2.1 describes the networks from the two datasets that were used for this study. The networks in the DIMACS dataset were further classified into two categories based on whether they follow a heavy tail degree distribution or not.

Table 2.2 compares the upper bounds obtained by using the scheme described in Algorithm 2 against those found by using the k-cores in Algorithm 2 instead of k-community. The number of nodes remaining in the corresponding $(\omega_{ub}-1)$-core and $(\omega_{ub}-2)$-community are also provided. The table also provides the lower bounds ω_{lb} found in the course of Algorithm 4, along with the number of nodes remaining in the corresponding $(\omega_{lb}-1)$-core and $(\omega_{lb}-2)$-community.

It can be seen that compared to the k-core upper bound, the k-community upper bounds are significantly lower, almost by a factor of 2. This is because the k-community is a much tighter relaxation of a clique. Furthermore, the number of nodes remaining in the corresponding $(\omega_{ub}-1)$-core and $(\omega_{ub}-2)$-community provide further evidence of the

Table 2.1: Description of the various networks used for computations. More information about the graphs can be obtained from [SNAP, Last accessed: November 2011] and [DIMACS10, Last accessed: May 2012].

Network Type	Example/Description
SNAP	
Social Networks	Epinions.com: Who-trusts-whom network of Epinions.com.
	Slashdot Slashdot social network for a month.
	Wikipedia who-votes-on-whom network.
Communication	Email network from a EU research institution.
	Wikipedia talk (communication) network.
Citation Networks	Citation network among US Patents.
	Arxiv High Energy Physics paper citation network.
Web graphs	Web graph of Stanford.edu.
	Web graph from Google.
Product Co-purchasing	Amazon product co-purchasing network for a day.
Internet P2P	Gnutella peer to peer network for a day.
DIMACS10-HeavyTail	
Clustering	Used as benchmarks in the graph clustering.
Coauthors	Social networks are created from co-authorships and citations.
Random Geometric	Generated from random points in the unit square. Edges connect vertices whose Euclidean distance is below $0.55\log(n)/n$.
Kronecker	Synthetic graphs created with the Kronecker generator.
DIMACS10-QuasiRegular	
Matrix	Florida Sparse Matrix Collection.
Walshaw	Benchmarks for graph partitioning algorithms.

tightness of k-communities. The lower bounds ω_{lb} were obtained by running an exact algorithm on the corresponding $(\omega_{ub}-1)$-cores and $(\omega_{ub}-2)$-communities of the graphs. In many cases, especially when the degree distribution of the graph does not follow a power law, the $(\omega_{ub}-1)$-cores have a large number of nodes, and the lower bounds cannot be found. In such cases, the lower bound obtained from the greedy algorithm is reported (marked by an asterisk).

Table 2.3 provides the maximum clique sizes as found by the Algorithm 4, with the time taken by six variants of the algorithm. These variants differ in the clique relaxation used for scale reduction (k-core, k-comunity, and hybrid) and the search procedure used to find the upper bounds (linear and binary search). The hybrid scale reduction uses a k-core based in the beginning, and if a sufficient reduction is not found, resorts to a k-community reduction.

Table 2.2: Comparison of the upper bounds (ω_{ub}) and lower bounds (ω_{lb}) obtained by a k-core scheme vs a k-comm scheme. Comparison of the number of nodes in the corresponding $(\omega_{ub}-1)$-core, $(\omega_{ub}-2)$-community, $(\omega_{lb}-1)$-core, and $(\omega_{lb}-2)$-community, ($n_{\omega_{ub}}$ and $n_{\omega_{lb}}$) is also included. The $n_{\omega_{ub}}$ and $n_{\omega_{lb}}$ values are in bold when they are larger than $12,000$. An asterisk marks the ω_{lb} value when the best lower bound was obtained from the greedy algorithm.

			k-core				*k*-comm			
Graph	**n**	**m**	ω_{ub}	$n_{\omega_{ub}}$	ω_{lb}	$n_{\omega_{lb}}$	ω_{ub}	$n_{\omega_{ub}}$	ω_{lb}	$n_{\omega_{lb}}$
SNAP										
Wiki-Vote	7115	100762	54	336	17	2316	23	50	17	458
p2p-Gnutella04	10876	39994	8	365	3	8379	4	12	4	12
p2p-Gnutella25	22687	54705	6	6091	4	9764	4	25	4	25
p2p-Gnutella24	26518	65369	6	7480	4	11478	4	41	4	41
Cit-HepTh	27770	352285	38	52	22	7278	30	48	21	366
Cit-HepPh	34546	420877	31	40	18	11284	25	36	18	193
p2p-Gnutella30	36682	88328	8	14	3*	**20194**	4	42	4	42
p2p-Gnutella31	62586	147892	7	1004	4*	**24222**	4	57	4	57
soc-Epinions1	75879	405740	68	486	23	5004	33	61	23	402
Slashdot0811	77360	469180	55	129	26	5050	35	87	26	164
Slashdot0902	82168	504230	56	134	27	5043	36	96	27	165
Amazon0302	262111	899792	7	286	7	286	7	105	7	105
Email-EuAll	265214	364481	38	292	16	1691	20	62	16	157
web-Stanford	281903	1992636	72	387	18*	34325	62	64	61	128
web-NotreDame	325729	1090108	156	1367	155	1367	155	155	155	155
Amazon0312	400727	2349869	11	**27046**	9*	**244256**	11	4534	11	4534
Amazon0601	403394	2443408	11	**32886**	11*	**32886**	11	5361	11	5361
Amazon0505	410236	2439437	11	**32632**	8*	**295845**	11	4878	11	4878
web-BerkStan	685230	6649470	202	392	201	392	201	392	201	392
web-Google	875713	4322051	45	48	44	103	44	48	44	48
WikiTalk	2394385	4659565	132	700	26	**15807**	53	237	26	1559
cit-Patents	3774768	16518947	65	106	10	**354843**	36	83	10	3131
DIMACS10-HeavyTail										
as-22july06	22963	48436	26	71	17	144	17	45	17	45
cond-mat-2005	40421	175691	30	30	30	30	30	30	30	30
kron_g500-simple-logn16	65536	2456071	433	694	136	6885	285	676	136	2513
G_n_pin_pout	100000	501198	8	**74227**	3*	**99942**	4	4	4	4
preferentialAttachment	100000	499985	6	**100000**	6*	**100000**	6	7	6	7
smallworld	100000	499998	8	**99737**	5*	**100000**	6	**14749**	6*	**14749**
caidaRouterLevel	192244	609066	33	92	16*	4021	19	36	17	58
coAuthorsCiteseer	227320	814134	87	87	87	87	87	87	87	87
citationCiteseer	268495	1156647	16	67	10	**35093**	13	13	13	13
coAuthorsDBLP	299067	977676	115	115	115	115	115	115	115	115
cnr-2000	325557	2738969	84	86	84	86	84	86	84	86
coPapersCiteseer	434102	16036720	845	845	845	845	845	845	845	845
coPapersDBLP	540486	15245729	337	337	337	337	337	337	337	337
eu-2005	862664	16138468	389	405	387	405	387	391	387	391
in-2004	1382908	13591473	489	491	489	491	489	490	489	490
rgg_n_2_21_s0	2097152	14487995	19	19	19	19	19	19	19	19
rgg_n_2_22_s0	4194304	30359198	20	20	20	20	20	20	20	20
rgg_n_2_23_s0	8388608	63501393	21	22	21	22	21	22	21	22
rgg_n_2_24_s0	16777216	132557200	21	82	21	82	21	44	21	44
uk-2002	18520486	261787258	944	944	944	944	944	944	944	944
DIMACS10-QuasiRegular										
luxembourg.osm	114599	119666	3	**93000**	2*	**114599**	3	204	3	204
wave	156317	1059331	9	**119747**	5*	**156311**	7	9	6	389
audikw1	943695	38354076	48	**687633**	36*	**937779**	39	135	36	**185805**
ldoor	952203	22785136	35	**900844**	21*	**952203**	21	**952203**	21*	**952203**
ecology1	1000000	1998000	3	**1000000**	2*	**1000000**	2	**1000000**	2*	**1000000**
belgium.osm	1441295	1549970	4	5	3	**1238894**	3	7113	3	7113
333SP	3712815	11108633	5	**2261408**	3*	**3712815**	4	28	4	28
cage15	5154859	47022346	26	**27712**	6*	**5135355**	6	**520172**	6*	**520172**

Table 2.3: Maximum clique sizes as found by the Algorithm 4, and the time taken by variants *k*-comm, *k*-core, and hybrid scale reductions when using binary and linear search for finding the upper bound. The best time for each instance is in bold. A ~~strike~~ marks the cases in which optimality of the clique found could not be validated, and a under-wave marks those in which an extension of *k*-community was used to establish optimality. A * is marks cases where connected components of the $(\omega_{lb}-1)$-core or the $(\omega_{lb}-2)$-community had to be found.

				Binary			Linear		
Graph	**n**	**m**	ω	**k-Core**	**k-Comm**	**Hybrid**	**k-Core**	**k-Comm**	**Hybrid**
SNAP									
Wiki-Vote	7115	100762	17	0.19	0.23	0.22	0.19	0.20	0.19
p2p-Gnutella04	10876	39994	4	1.75	0.05	0.69	1.06	0.06	1.06
p2p-Gnutella25	22687	54705	4	3.65	0.08	1.56	3.62	0.06	1.48
p2p-Gnutella24	26518	65369	4	4.35	0.08	1.47	4.27	0.11	1.37
Cit-HepTh	27770	352285	23	1.70	1.06	0.55	1.45	1.08	1.61
Cit-HepPh	34546	420877	19	3.71	0.27	0.97	3.71	0.25	1.51
p2p-Gnutella30	36682	88328	4	~~0.25~~*	0.22	0.34	~~0.14~~*	0.20	0.28
p2p-Gnutella31	62586	147892	4	~~0.36~~*	0.02	0.23	~~0.19~~*	0.02	0.09
soc-Epinions1	75879	405740	23	1.08	1.14	0.87	1.47	1.06	1.51
Slashdot0811	77360	469180	26	1.11	0.23	0.64	1.56	0.20	1.61
Slashdot0902	82168	504230	27	1.15	0.23	0.59	1.68	0.25	1.67
Amazon0302	262111	899792	7	2.04	0.89	1.98	0.34	0.39	0.34
Email-EuAll	265214	364481	16	0.25	0.25	0.23	0.22	0.23	0.28
web-Stanford	281903	1992636	61	6.21	4.87	6.32	2.68	4.87	2.70
web-NotreDame	325729	1090108	155	0.30	0.27	0.31	0.30	0.25	0.31
Amazon0312	400727	2349869	11	~~14.93~~	3.31	6.27	~~10.76~~	2.43	3.85
Amazon0601	403394	2443408	11	5.77	2.09	6.44	1.48	2.03	3.40
Amazon0505	410236	2439437	11	~~17.64~~	3.29	5.91	~~13.78~~	2.01	2.79
web-BerkStan	685230	6649470	201	16.51	33.14	16.44	11.76	33.71	11.83
web-Google	875713	4322051	44	4.76	3.89	4.90	1.97	2.90	1.92
WikiTalk	2394385	4659565	26	~~13.26~~*	12.29	16.29	~~9.91~~*	12.20	12.84
cit-Patents	3774768	16518947	11	~~59.38~~*	23.76	37.11	~~43.06~~*	18.95	20.72
DIMACS10-HeavyTail									
as-22july06	22963	48436	17	0.02	0.02	0.03	0.02	0.02	0.02
cond-mat-2005	40421	175691	30	0.03	0.02	0.03	0.03	0.03	0.03
kron_g500-simple-logn16	65536	2456071	136	820.11	953.54	821.00	818.93	955.55	822.07
G_n_pin_pout	100000	501198	4	~~2.45~~*	0.23	0.81	~~2.20~~*	0.12	0.39
preferentialAttachment	100000	499985	6	1.12	0.09	1.11	0.16	0.11	0.38
smallworld	100000	499998	6	~~2.08~~*	0.41*	0.62*	~~1.76~~*	0.27*	0.39*
caidaRouterLevel	192244	609066	17	0.52	0.16	0.53	0.47	0.13	0.44
coAuthorsCiteseer	227320	814134	87	0.13	0.20	0.11	0.09	0.59	0.13
citationCiteseer	268495	1156647	13	~~2.59~~*	1.05	2.43	~~1.59~~	1.08	1.61
coAuthorsDBLP	299067	977676	115	0.13	0.14	0.14	0.14	0.14	0.14
cnr-2000	325557	2738969	80	23.99	17.83	23.85	4.76	17.83	4.59
coPapersCiteseer	434102	16036720	845	4.01	5.82	4.17	3.98	6.01	4.01
coPapersDBLP	540486	15245729	337	2.26	3.57	2.18	2.29	3.62	2.23
eu-2005	862664	16138468	387	104.71	351.32	104.66	16.97	2061.41	17.43
in-2004	1382908	13591473	489	17.54	67.53	17.75	17.66	67.27	17.68
rgg_n_2_21_s0	2097152	14487995	19	1.44	1.61	1.44	1.36	1.56	1.40
rgg_n_2_22_s0	4194304	30359198	20	3.04	3.09	3.09	3.00	3.21	3.00
rgg_n_2_23_s0	8388608	63501393	21	5.88	5.98	5.91	5.94	5.93	5.96
rgg_n_2_24_s0	16777216	1.33E+08	21	29.64	19.78	30.06	21.94	19.98	22.12
uk-2002	18520486	2.62E+08	944	225.89	472.05	225.28	162.69	5106.09	164.13
DIMACS10-QuasiRegular									
luxembourg.osm	114599	119666	3	~~0.38~~*	0.17	0.33	~~0.28~~*	0.08	0.27
wave	156317	1059331	6	~~1.47~~*	1.56	2.34	~~0.94~~*	0.97	1.89
audikw1	943695	38354076	36$^{*\wedge}$	50.16*	122.81*	175.97*	24.88*	120.65*	157.58*
ldoor	952203	22785136	21	~~15.43~~*	57.91	64.94	~~7.46~~*	31.06	27.61
ecology1	1000000	1998000	2	~~2.89~~*	1.42	2.08	~~2.06~~*	1.08	1.36
belgium.osm	1441295	1549970	3	~~3.85~~*	4.09	3.68	~~2.98~~*	2.15	2.78
333SP	3712815	11108633	4	~~89.37~~*	21.40	133.49	~~128.16~~*	10.76	72.40
cage15	5154859	47022346	6	~~994.55~~*	22.11	56.08	~~983.58~~*	15.79	34.88

It can be observed that for almost all the graphs, the maximum clique was found within ten minutes, which is remarkable considering the size of the graphs. The relative tightness of k-communities when compared to k-cores is apparent not just from the upper bounds found in Table 2.2, but also from the fact that many instances that could not be solved using a k-core reduction were solved by the k-community reduction. The linear upper bound search, which was introduced as a less memory intensive algorithm targeting large graphs, does prove to be effective in reducing the time taken by the algorithm for the larger instances. A glance at Table 2.2 also suggests that the upper bounds are fairly tight, and that the lower bounds obtained are very close to the clique number. Amongst the three variants, the hybrid scale reduction method with linear search seems to perform the best overall. The results presented in this table also highlights the main contribution of this research is that we are able to obtain the maximum cliques for very large scale graphs with a proof of optimality for all but one of the graphs tested. The pool of test instances taken is diverse, with both power law and fairly regular graphs present. In fact, the one graph for which the proof of optimality does not have a power law distribution.

Also note that although the residual graphs ($(\omega_{ub}-2)$-communities and $(\omega_{ub}-1)$-cores) are quite large for the DIMACS2 graphs, the greedy clique is the same size as the upper bound found, not requiring an exact algorithm at all. An exception is the graph audikw1 for which the upper bound obtained by k-community is 39, whereas the greedy lower bound is 36. However, the enhancement *edge-neighborhood-k-core* defined in section 2.4 is used to establish that the upper bound on the clique size is 36, and hence the optimalilty of the greedy clique is established.

2.7 Conclusion

This chapter introduces a new clique relaxation, and an algorithm based on it to find the maximum cliques in large low density graphs. Using the methodology developed in this chapter we were able to find the maximum cliques for graphs with up to 18 million nodes, which has not been achieved till date. Furthermore, any advancements in exact algorithms for dense but smaller graphs will directly impact the performance of this methodology in a positive manner. The scheme might be useful for clique relaxations like *k*-plex too. In future, we would like to explore clustering techniques based on the *k*-community and evaluate their effectiveness.

3. NETWORK CLUSTERING VIA CLIQUE RELAXATIONS: A COMMUNITY BASED APPROACH

3.1 Introduction

Network (graph) based data mining is an emerging field that studies network representations of data sets generated by an underlying complex system in order to draw meaningful conclusions regarding the system's properties. In a network representation of a complex system, the network's nodes typically denote the system's entities, while the edges between nodes represent a certain kind of similarity or relationship between the entities. Network clustering, aiming to partition a network into clusters of similar elements, is an important task frequently arising within this context. The form of each cluster in the partitioning is commonly specified through a predefined graph structure. Since a cluster is typically understood as a "tightly knit" group of elements, the graph theoretic concept of a *clique*, which is a subset of nodes inducing a complete subgraph, is a natural formalization of a cluster that has been used in many applications. This results in partitioning into "ideal" clusters, with the highest possible level of cohesiveness one can hope for.

The flawlessness of the clique structure as a theoretical formalization of a cohesive cluster turns into a "curse of perfection" when it comes to practical applications. Since each node in a clique is required to be connected to all other nodes in the clique, a highly cohesive structure might not get identified as a cluster by the mere absence of a few edges. In real life data sets, this is of critical importance because some edges could be missing either naturally or due to erroneous data collection. Moreover, given that networks arising in many important applications tend to be very large with respect to the number of nodes and very sparse in terms of the relative number of edges, the clique clustering usually results in meaninglessly large number of clusters in such situations. In addition, computing large cliques and good clique partitions are computationally challenging problems, as finding a maximum clique and a minimum clique partition in a graph are classical NP-hard problems Garey and Johnson [1979a].

To circumvent these drawbacks of cliques, researchers in several applied fields, such as social network analysis and computational biology, have defined and studied structures that relax some of the properties of cliques, and hence are aptly called clique relaxations. Some of the popular clique relaxations include s-plexes, that require each vertex to be connected to all but s other vertices Seidman and Foster [1978]; s-clubs, that require the diameter of the induced subgraph to be at most s Alba [1973]; and γ-quasi-cliques, which require the density of the induced subgraph to be at least γ Abello et al. [2002]. It should be noted that each of 1-plex, 1-club and 1-clique represents a clique. By relaxing the properties of a clique, namely the degree, diameter, and density, these clique relaxations capture clusters that are strongly but not completely connected. However, like the clique model, these clique relaxations still suffer from the drawback of being computationally expensive.

In 1983, Seidman Seidman [1983] introduced the concept of a *k-core* that restricts the minimum number k of direct links a node must have with the rest of the cluster. Using k-cores to model clusters in a graph has considerable computational advantages over the other clique clique relaxation models mentioned above. Indeed, the problem of finding the largest k-core can be easily solved in polynomial time by recursively removing vertices of degree less than k. As a result, the k-core model has gained significant popularity as a network clustering tool in a wide range of applications. In particular, k-core clustering has been used as a tool to visualize very large scale networks Alvarez-Hamelin et al. [2005], to identify highly interconnected subsystems of the stock market Idicula [2004], and to detect molecular complexes and predict protein functionsBader and Hogue [2003], Altaf-Ul-Amin et al. [2003]. On the downside, the size of a k-core may be much larger than k, creating a possibility of a low level of cohesion within the resulting cluster. Because of this, a k-core itself may not be a good model of a cluster, however, it has been observed that k-cores tend to contain other, more cohesive, clique relaxation structures, such as s-plexes, and hence computing a k-core can be used as a scale-reduction step while detecting other structures Balasundaram et al. [2011].

Most recently, the authors of the current paper proposed yet another clique relaxation model of a cluster, referred to as *k-community*, that aims to benefit from the positive properties of k-cores while ensuring a higher level of cohesion. More specifically, a k-community is a connected subgraph such that endpoints of every edge have at least k common neighbors within the subgraph. As seen in the previous chapter, the k-community structure has proven to be extremely effective in reducing the scale of very large, sparse instances of the maximum clique problem. This research explores the potential of using the k-community structure as a network clustering tool. Even though the proposed clustering algorithm does not aim to optimize any of the quantitative measures of clustering quality, the results of numerical experiments show that it performs quite well with respect to most of such measures available in the literature.

The remainder of this chapter is organized as follows. Section 3.2 provides the necessary background information. Section 3.3 outlines the proposed network clustering algorithm. Section 3.4 reports the results of numerical experiments on several benchmark instances, and Section 3.5 concludes the chapter.

3.2 Background

In this chapter, a network is described by a simple undirected graph $G = (V, E)$ with the set $V = \{1, 2, \dots, n\}$ of nodes and the set E of edges. We call a pair of nodes u and v such that $(u, v) \in E$ *adjacent* or *neighbors*. For a node u, let $N_G(u) = \{v : (u, v) \in E\}$ denote the neighborhood of u in G. Then the degree $deg_G(u)$ of u in G is given by the number of elements in $N_G(u)$. Let $\delta(G)$ denote the minimum degree of a node in G. For a subset C of nodes, $G[C] = (C, E \cap (C \times C))$ denotes the subgraph induced by C. Next we define two clique relaxation concepts, namely k-core and k-community, that play a key role in this research.

Definition 3.2.1 (k-core)**.** *A subset of nodes C is called a k-core if $G[C]$ is a connected graph and $\delta(G[C]) \geq k$.*

Algorithm 5 $k-\texttt{Community}(G)$: Algorithm to find the k-Communities of G

```
1: repeat
2:     for every $(i,j) \in E$ do
3:         if $|N_G(i) \cap N_G(j)| < k$ then
4:             $E \leftarrow E \setminus \{(i,j)\}$
5:         end if
6:     end for
7: until No edge is removed in the current iteration
8: $G(V_k,E_k) \leftarrow G_e[E]$      /* Edge induced subgraph */
9: return $\mathcal{V}_k \leftarrow$ Connected components of $G(V_k,E_k)$. /* Each set of connected vertices
   forms a $k$-community*/
```

Before defining a k-community, we need the following two preliminary definitions.

Definition 3.2.2 (Neighbor of an edge)**.** *A node $t \in V$ is a neighbor of an edge $(u,v) \in E$ if it is connected to both u and v, i.e., $(v,t) \in E$ and $(u,t) \in E$.*

Definition 3.2.3 (Edge induced subgraph)**.** *An edge induced subgraph, denoted by $G_e[F]$, is a subset of edges F of a graph G together with all the incident vertices.*

We are now ready to define a k-community, which can be seen as an edge analogue of the k-core as follows.

Definition 3.2.4 (k-Community)**.** *A k-Community of a graph G is the set of nodes in the connected edge induced subgraph $G_e[E_k]$ with each edge in E_k having at least k neighboring vertices in the subgraph $G_e[E_k]$. If $G_e[E_k]$ is disconnected, then each component forms a k-community by itself.*

Given a positive integer k, both of these structures are in essence trying to find a cluster of vertices that satisfies some minimum node degree requirements. In the case of k-core, the presence of each node has to be supported by the presence of at least k neighbors, while in the case of k-community, the presence of each edge has to be supported by the presence of at least k alternative edge-disjoint paths of length two. It is instructive to note that every k-community is also a $(k+1)$-core, but the converse is not true.

Algorithm 6 `Basic` k`-Community Clustering`(G): Basic algorithm to find clusters in G

```
1: $G' \leftarrow G$
2: $\mathscr{C} \leftarrow \emptyset$
3: **repeat**
4:     $k \leftarrow$ highest integer such that $k$-community($G'$) is non-empty.
5:     Find all the $k$-Communities in $G'$ and add them to $\mathscr{C}$.
6:     Find the set of vertices $L$ that are not yet clustered.
7:     $G' \leftarrow G[L]$.
8: **until** $k \leq l$ or $G'$ is empty
9: **for** every $v \in L$ **do**
10:     Add $v$ to the cluster $C \in \mathscr{C}$ which maximizes $|N(v) \cap C|$.
11: **end for**
12: **return** $\mathscr{C}$
```

Given a positive integer k, all the maximal k-communities of a graph G can be easily computed as outlined in Algorithm 5.

3.3 Clustering Algorithm

The algorithm described in this section is based on the idea of finding k-communities for large k and placing them in different clusters. To this end, we identify the largest k' such that the k'-community of G is non-empty, and place all k'-communities formed in distinct clusters. Once this has been done, all the nodes that have been placed in clusters are removed from G and the whole procedure is repeated till either k becomes small (reaches a lower bound l provided by the user) or no vertices are left to cluster. If any vertex is left to cluster, we attach it to the cluster that contains the most neighbors of that vertex. This basic procedure is described in Algorithm 6.

In this algorithm, we stop when k becomes small enough so that a k-community becomes meaningless. For example, any set of vertices that induce a tree will form a 0-community. While in some cases this might be the best possible option (the original graph is a forest), for most clustering instances we would like the vertices in a cluster to share

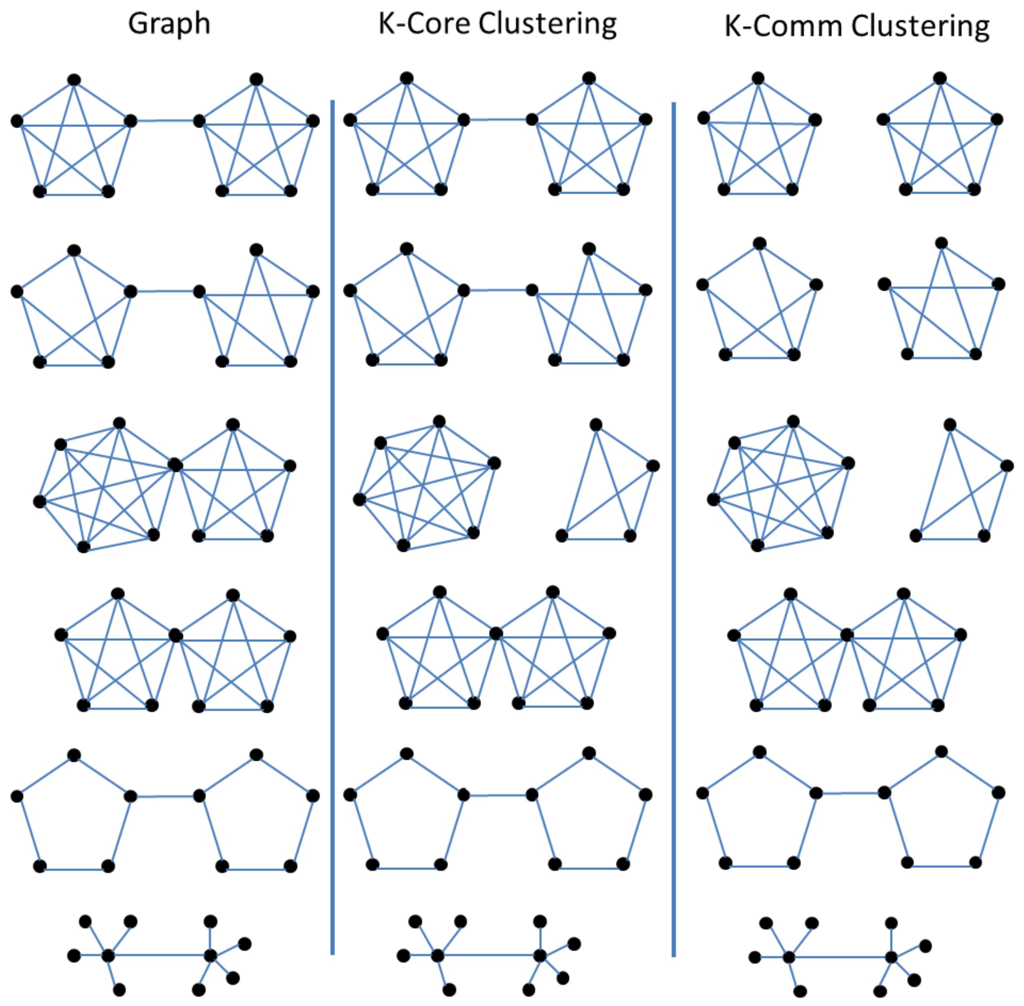


Fig. 3.1.: Clustering found by Algorithm 6 using the *k*-core and *k*-community-based approaches on some illustrative graphs. The diagram highlights the cases where community based approach is better than the core-based approach, and also when none of them perform well.

more than just one edge with the remaining nodes. For this chapter, the lower bound l was set to 1 in Algorithm 6.

It should be noted that the clustering provided by Algorithm 6 does not aim to optimize any criteria provided such as modularity, performance, average isolated inter-cluster conductance (aixc), average isolated inter-cluster expansion (aixe), and minimum intra-cluster

Algorithm 7 `Enhanced k-Community Clustering`(G): Enhanced algorithm to find clusters in G

```
1:  $G' \leftarrow G$
2:  $\mathscr{C} \leftarrow \emptyset$, best_mod $\leftarrow -1/2$
3:  **repeat**
4:      $k \leftarrow$ highest integer such that $k$-community($G'$) is non-empty.
5:      Find all the $k$-Communities in $G'$ and add them to $\mathscr{C}$.
6:      Find the set of vertices $L$ that are not yet clustered.
7:      $G' \leftarrow G[L]$.
8:      **if** $k \leq u$ **then**
9:          $\mathscr{C}^k \leftarrow \mathscr{C}$
10:         **for** every $v \in L$ **do**
11:             Add $v$ to the cluster $C^k \in \mathscr{C}^k$ which maximizes $|N(v) \cap C^k|$.
12:         **end for**
13:         **if** Modularity($\mathscr{C}^k$) $<$ best_mod **then**
14:             $\mathscr{C} \leftarrow \mathscr{C}^{k-1}$
15:             **break**
16:         **else**
17:             best_mod $\leftarrow$ Modularity($\mathscr{C}^k$)
18:         **end if**
19:     **end if**
20: **until** $k$=1 or $G'$ is empty
21: **for** every $v \in L$ **do**
22:     Add $v$ to the cluster $C \in \mathscr{C}$ which maximizes the increase in Modularity($\mathscr{C}$).
23: **end for**
24: LocalSearch($\mathscr{C}$)
25: **return** $\mathscr{C}$
```

density (mid) as described in the DIMACS 2011 challenge DIMACS10 [Last accessed: May 2012].

3.3.1 Enhancements

If optimizing a given measure is indeed the aim, an enhanced version of the basic algorithm is provided in Algorithm 7. The description of the enhanced Algorithm 7 uses modularity as a measure, but can as well have any other measure. A major improvement in Algorithm 7 over Algorithm 6 is that the decision of the what k is too small to be used for finding k-communities as clusters is made dynamically. Given a range $[l,u]$, the algorithm checks the modularity of the clustering found at each $k \leq u$ and stops as soon as reducing k also reduces modularity. In this manner, the formation of k-communities for small k that don't contribute to increasing modularity can be avoided. Furthermore, local search is done to increase modularity by moving vertices from one cluster to another cluster such that the increase in modularity is maximized. For the results in this chapter, the range $[l,u]$ was set to $[1,6]$, and the time spent in local search was restricted to $10,000$ seconds.

An advantage of both these algorithms is that they do not use any prior information about the graph such as the number of clusters, degree distribution, etc. This makes it a very general approach that is applicable even when no information about the structure of the graph is available. Furthermore, although we use k-core and k-community to define clusters, new structures that fit the users description of a cluster can be incorporated into the algorithm fairly easily.

In both the Algorithms 6 & 7, we can replace k-community in steps 4-5 with k-core, with the remaining steps of the algorithm as they are, to obtain a k-core-based clustering algorithm.

Some illustrations of clusterings found by the k-core and k-community approach described in this section are provided in Figure 3.1. It should be noted that, although k-communities are strictly stronger relaxations, the clustering formed by the core-based approach can in some cases be better than that obtained using the community-based approach.

3.4 Computational Results

In this section we provide computational results obtained by using the k-community and k-core clustering on the graph sets provided in the 10th DIMACS challenge DIMACS10 [Last accessed: May 2012]. The computational results were obtained on a desktop machine (Intel Xeon E5620@2.40GHz, 16 cores, 12GB RAM). All computations except for the final steps of attaching leftover vertices to already formed clusters and

Table 3.1: Modularity of clustering found by the basic Algorithm 6 using the *k*-community based and *k*-core based approaches. The modularity that is higher between the two methods is highlighted in bold.

			k-core-based			*k*-community-based		
Graphs	**n**	**m**	**Mod**	**Clusters**	**Time(s)**	**Mod**	**Clusters**	**Time(s)**
celegans_metabolic	453	2025	0.267	19	0.00	**0.331**	30	0.02
email	1133	5451	0.342	15	0.03	**0.394**	72	0.03
polblogs	1490	16715	**0.243**	8	0.03	0.219	32	0.06
power	4941	6594	0.295	24	0.05	**0.851**	189	0.09
PGPgiantcompo	10680	24316	**0.755**	398	0.47	0.732	655	0.64
astro-ph	16706	121251	**0.539**	918	1.70	0.538	1480	1.95
memplus	17758	54196	**0.555**	1238	0.56	0.554	1256	0.58
as-22july06	22963	48436	0.473	33	0.41	**0.519**	162	0.59
cond-mat-2005	40421	175691	**0.509**	2469	3.85	0.508	4016	4.99
kron_g500-simple-logn16	65536	2456071	-0.018	6	15.31	**-0.013**	28	38.60
preferentialAttachment	100000	499985	0.000	1	1.01	**0.145**	299	14.85
G_n_pin_pout	100000	501198	0.065	2	4.23	**0.136**	4479	33.93
smallworld	100000	499998	0.000	4	0.48	**0.570**	11129	9.64
luxembourg.osm	114599	119666	0.000	1	10.50	**0.955**	68	95.47
rgg_n_2_17_s0	131072	728753	**0.752**	7539	9.91	0.612	15572	13.64
caidaRouterLevel	192244	609066	**0.625**	5436	55.83	0.605	6005	78.57
coAuthorsCiteseer	227320	814134	**0.701**	17185	102.99	0.690	23562	127.65
citationCiteseer	268495	1156647	**0.481**	2145	91.69	0.433	11499	194.66
coPapersDBLP	540486	15245729	**0.670**	31213	1429.25	0.669	34267	1557.58
eu-2005	862664	16138468	0.304	18403	1965.33	**0.404**	30380	2570.01
audikw1	943695	38354076	0.241	10190	550.23	**0.389**	22076	1151.01
ldoor	952203	22785136	0.091	361	20.23	**0.392**	2	42.06
kron_g500-simple-logn20	1048576	44619402	-0.026	5	1554.64	**-0.025**	1788	3155.71
in-2004	1382908	13591473	**0.632**	29528	2774.93	0.625	43454	3416.69
belgium.osm	1441295	1549970	0.000	2	889.65	**0.983**	2326	7118.33
cage15	5154859	47022346	**0.813**	4958	14451.30	0.544	174163	259.33

the local search used only one core. The local search and attaching leftover vertices were parallelized using OpenMP with 16 threads.

Table 3.1 presents the modularity and number of clusters found by Algorithm 6 using the *k*-core and *k*-community clustering for 27 graphs. For each graph, the higher of the two modularities as found be the two methods is highlighted in bold. It can be seen that *k*-community clustering is better on about half of the instances (14 of the 27 graphs tested). However, a closer look suggests that when the *k*-community based clustering significantly outperforms (difference in modularity more than 0.2) *k*-core clustering in 5 of those 14 instances, while *k*-community based clustering is significantly outperformed by *k*-core clustering only once out of the remaining 13 instances. Some noteworthy examples are the *preferentialAttachment, smallworld, luxembourg.osm* and *belgium.osm* graphs, where the almost all nodes in the graph are identified as 4-, 6-, 1- & 1-cores respectively and placed in one huge cluster by the *k*-core clustering. On the other hand, the *k*-community clustering is able to identify a more meaningful clustering. The examples provided in Figure 3.1 point to some potential reasons why *k*-cores are not able to cluster these graphs as well as *k*-communities do.

Table 3.2: Modularity of clustering found by the enhanced Algorithm 7 using the *k*-community based and *k*-core based approaches. The modularity that is higher between the two methods is highlighted in bold. The improvement in modularity when compared to the basic Algorithm 6 and the time taken are also provided.

				k-core-based			**k-community-based**	
Graphs	**n**	**m**	**Mod**	**Improv**	**Time(s)**	**Mod**	**Improv**	**Time(s)**
celegans_metabolic	453	2025	0.360	0.092	0.16	**0.402**	0.071	0.17
email	1133	5451	0.477	0.134	0.98	**0.542**	0.148	0.62
polblogs	1490	16715	0.419	0.176	2.75	**0.426**	0.206	0.16
power	4941	6594	0.759	0.464	0.55	**0.860**	0.009	0.50
PGPgiantcompo	10680	24316	0.835	0.080	1.54	**0.848**	0.116	1.59
astro-ph	16706	121251	**0.651**	0.112	25.93	0.646	0.108	6.94
memplus	17758	54196	**0.537**	-0.017	4.62	**0.537**	-0.017	4.45
as-22july06	22963	48436	0.513	0.041	113.67	**0.603**	0.084	43.85
cond-mat-2005	40421	175691	**0.625**	0.116	273.29	0.620	0.112	16.13
kron_g500-simple-logn16	65536	2456071	**0.023**	0.040	10019.40	0.014	0.027	1700.88
preferentialAttachment	100000	499985	0.000	0.000	22.00	**0.243**	0.097	10041.40
G_n_pin_pout	100000	501198	0.065	0.000	131.02	**0.212**	0.076	10047.00
smallworld	100000	499998	0.000	0.000	19.63	**0.753**	0.184	43.99
luxembourg.osm	114599	119666	0.000	0.000	29.00	**0.958**	0.003	233.72
rgg_n_2_17_s0	131072	728753	**0.871**	0.119	35.77	0.800	0.188	72.29
caidaRouterLevel	192244	609066	0.776	0.151	5447.02	**0.821**	0.216	340.88
coAuthorsCiteseer	227320	814134	**0.823**	0.122	397.66	0.817	0.127	211.66
citationCiteseer	268495	1156647	0.639	0.157	10142.10	**0.709**	0.276	483.39
coPapersDBLP	540486	15245729	**0.716**	0.046	2581.11	0.715	0.046	2720.36
eu-2005	862664	16138468	0.671	0.367	15205.00	**0.757**	0.353	11874.90
audikw1	943695	38354076	0.325	0.084	10826.60	**0.637**	0.248	11231.10
ldoor	952203	22785136	0.092	0.001	6130.62	**0.392**	0.000	847.84
kron_g500-simple-logn20	1048576	44619402	-0.024	0.002	11626.20	**0.010**	0.036	13737.80
in-2004	1382908	13591473	0.924	0.292	6033.33	**0.926**	0.302	5887.41
belgium.osm	1441295	1549970	0.000	0.000	55142.10	**0.983**	0.000	7112.92
cage15	5154859	47022346	**0.816**	0.004	25787.80	0.709	0.165	71808.90

Table 3.3: The modularity (Mod), coverage (Cov), mirror coverage (MCov), performance (Perf), average isolated inter-cluster conductance (Aixc), average isolated inter-cluster expansion (Aixe), and minimum intra-cluster density (Mid) found by the basic Algorithm 6 and enhanced Algorithm 7 using k-community.

	Algorithm 6 (Basic)							**Algorithm 7 (Enhanced)**						
Graph	**Mod**	**Cov**	**Mcov**	**Perf**	**Aixc**	**Aixe**	**Mid**	**Mod**	**Cov**	**Mcov**	**Perf**	**Aixc**	**Aixe**	**Mid**
celegans_metabolic	0.33	0.57	0.86	0.85	0.50	3.25	0.05	0.40	0.58	0.88	0.87	0.34	2.17	0.06
email	0.39	0.44	0.96	0.96	0.58	5.26	0.02	0.54	0.62	0.93	0.93	0.38	3.15	0.03
polblogs	0.22	0.39	0.91	0.91	0.09	1.79	0.02	0.43	0.93	0.68	0.68	0.01	0.04	0.04
power	0.85	0.86	0.99	0.99	0.16	0.48	0.02	0.86	0.87	0.99	0.99	0.15	0.44	0.02
PGPgiantcompo	0.73	0.74	1.00	1.00	0.21	0.96	0.01	0.85	0.89	0.95	0.95	0.11	0.52	0.00
astro-ph	0.54	0.54	1.00	1.00	0.39	2.85	0.04	0.65	0.66	1.00	1.00	0.58	1.89	0.01
memplus	0.55	0.63	0.99	0.99	0.24	1.09	0.01	0.54	0.83	0.76	0.76	0.21	1.21	0.00
as-22july06	0.52	0.72	0.86	0.86	0.33	1.17	0.00	0.60	0.73	0.90	0.90	0.32	1.08	0.00
cond-mat-2005	0.51	0.51	1.00	1.00	0.45	2.40	0.01	0.62	0.62	1.00	1.00	0.71	1.92	0.01
kron_g500-simple-logn16	-0.01	0.33	0.73	0.72	0.00	0.17	0.00	0.01	0.67	0.47	0.47	0.00	0.06	0.00
preferentialAttachment	0.15	0.47	0.56	0.56	0.90	24.23	0.00	0.24	0.38	0.88	0.87	0.77	7.30	0.00
G_n_pin_pout	0.14	0.52	0.60	0.60	0.80	8.79	0.00	0.21	0.47	0.74	0.74	0.75	7.72	0.00
smallworld	0.57	0.57	1.00	1.00	0.49	4.91	0.13	0.75	0.75	1.00	1.00	0.28	2.81	0.02
luxembourg.osm	0.96	0.99	0.96	0.96	0.03	0.07	0.00	0.96	0.99	0.96	0.96	0.02	0.06	0.00
rgg_n_2_17_s0	0.61	0.61	1.00	1.00	0.45	4.71	0.20	0.80	0.80	1.00	1.00	0.22	2.50	0.06
caidaRouterLevel	0.61	0.62	0.99	0.99	0.38	1.81	0.00	0.82	0.85	0.97	0.97	0.96	2.13	0.00
coAuthorsCiteseer	0.69	0.69	1.00	1.00	0.31	1.83	0.01	0.82	0.82	1.00	1.00	0.17	1.38	0.00
citationCiteseer	0.43	0.45	0.98	0.98	0.48	3.69	0.00	0.71	0.72	0.99	0.99	0.29	2.49	0.00
coPapersDBLP	0.67	0.67	1.00	1.00	0.44	9.65	0.15	0.72	0.72	1.00	1.00	0.30	8.75	0.10
eu-2005	0.40	0.41	0.99	0.99	0.67	21.32	0.00	0.76	0.81	0.98	0.98	0.23	6.69	0.00
audikw1	0.39	0.51	0.90	0.90	0.83	51.18	0.00	0.64	0.77	0.87	0.87	0.04	2.96	0.00
ldoor	0.39	1.00	0.39	0.39	0.00	0.11	0.00	0.39	1.00	0.39	0.39	0.00	0.11	0.00
kron_g500-simple-logn20	-0.03	0.37	0.79	0.79	0.01	0.53	0.00	0.01	0.67	0.59	0.59	0.00	0.01	0.00
in-2004	0.62	0.63	1.00	1.00	0.40	12.63	0.00	0.93	0.94	0.99	0.99	0.19	2.33	0.00
belgium.osm	0.98	0.98	1.00	1.00	0.04	0.11	0.00	0.98	0.98	1.00	1.00	0.04	0.11	0.00
cage15	0.54	0.55	1.00	1.00	0.67	10.87	0.00	0.71	0.71	1.00	1.00	0.48	9.58	0.00

In addition, Table 3.1 also reports the time taken by the two approaches on each of the graphs. It can be seen that our approach scales well for large graphs, with graphs with up to 5 million vertices solved in reasonable time on a desktop machine.

Table 3.2 presents the modularity and number of clusters found by Algorithm 7 using the k-core and k-community clustering for the same 27 graphs. It can be seen that k-community based clustering outperforms k-core based clustering in 19 of the 27 instances. On an average, the improvement in the modularity was 0.099 for the k-core based clustering and 0.122 for the k-community based clustering. The time required for clustering increases, but is still within reasonable limit. A user can decide for or against using enhancements depending on the trade-off between the extra time required and the increase in modularity.

Table 3.3 presents the modularity, coverage, mirror coverage, performance, average isolated inter-cluster conductance, average isolated inter-cluster expansion, and minimum intra-cluster density for the clusterings found by the basic Algorithm 6 and the enhanced Algorithm 7 using the *k*-community based approach. It can be noted that while the enhanced Algorithm 7 increases the modularity, it does not always have a positive effect on other clustering measures. This is an important observation that suggests that modularity maximization should not be used as the sole measure of good clustering.

3.5 Conclusion

This chapter introduces *k*-community clustering, which can be thought of as something between *k*-core clustering and clique partitioning. The use of polynomially computable *k*-community not only provides a faster approach, but also provides a more effective clustering method by being able to identify cohesive structures that might not be cliques. *k*-Community clustering also provides advantages over *k*-core clustering due to the more cohesive nature of a *k*-community. As our computational results show, both the *k*-core and *k*-communities perform well for certain graphs, but *k*-community approach outperforms the *k*-core approach in general.

4. PRE-POSITIONING DISASTER RESPONSE FACILITIES FOR LARGE SCALE EMERGENCIES

4.1 Introduction

To provide responsive and timely service in the event of natural disasters and terrorist attacks, government agencies are developing large disaster response facilities to pre-position emergency supplies [Balcik and Beamon, 2008]. For example, in the United States, significant research interest has been generated in the location planning of these facilities after the Centers for Disease Control and Prevention (CDC) were entrusted with the task of establishing the Strategic National Stockpile (SNS). According to the CDC web site [CDC, Last accessed: May 2011]:

> Strategic National Stockpile (SNS) has large quantities of medicine and medical supplies to protect the American public if there is a public health emergency (terrorist attack, flu outbreak, earthquake) severe enough to cause local supplies to run out. Once Federal and local authorities agree that the SNS is needed, medicines will be delivered to any state in the U.S. within 12 hours.

This research focuses on the optimal placement of disaster response facilities like the SNS that will be used to pre-position emergency supplies. Emergency supplies can include food, medicine, potable water, but also medical equipment, generators, tents etc. In deciding on suitable locations for pre-positioning warehouses, decision makers need to consider disasters that may affect large geographical areas, with the potential to devastate entire cities. Earthquakes are a typical example, but hurricanes, large scale fires, and even non-natural events such as terrorist attacks are equally applicable.

Many models for locating facilities for pre-positioning emergency supplies have assumed that facilities are robust and will be functioning even in the wake of a natural disaster [Balcik and Beamon, 2008, Duran et al., 2011]. There exist models that consider facilities that might not be always available at their full capacity [Jia et al., 2007, Paul and

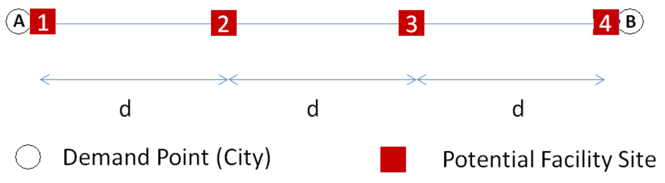


Fig. 4.1.: Illustration to show the shortcomings of the emergency facility location models in literature.

Batta, 2008, Beraldi et al., 2004]. These models assume that a disaster reduces the capacity of a facility by a certain fraction. However, as we will see in the literature review, these models decouple the chances of functioning of a facility from the actual disaster scenario.

To observe how the assumption that damage to disaster response facilities is independent of a given disaster scenario can impact the placement of facilities, consider a simple stylized example: suppose there are two cities A and B where population is concentrated, and four potential facility sites - one at A, one at B, and two between A and B as shown in the Figure 4.1. We will refer to the two cities as the demand points. The distances are as marked on the figure, and the chance of a disaster occurring at any of the two cities is the same. For the purpose of exposition, we assume that A and B are high risk areas where disasters might occur, and that the possibility of a disaster occurring at other locations is small enough to be ignored.

Suppose we want to construct two disaster response facilities. Existing location models developed by Jia et al. [2007], Paul and Batta [2008] and Beraldi et al. [2004] assume that the reduction in capacity of facilities is unrelated to where the disaster occurs, and would therefore suggest locating the facilities at sites 1 and 4. However, if a devastating earthquake occurs at city A, most likely facility 1 will be damaged because of its proximity to the disaster and may not be able to satisfy all demands. Aid would have to come from facility 4 which is far away. Similarly, if an earthquake occurs near city B, facility 4 would not be functioning, and aid would have to come all the way from facility 1. This intuitively poor placement decision occurs because these large scale emergency models assume that facility availability is independent of disaster location, whereas in actuality this is not true.

If one were to condition the functioning of the disaster response facilities on the actual disaster, better solutions might be found. Indeed, the model we present in this chapter suggests locating the facilities at sites 2 and 3. When a disaster happens at A, 2 and 3 being relatively far away from the disaster site will still be functioning at a slightly reduced capacity and can combine to send aid. When a disaster happens at B, the same holds true. Locating facilities at sites 2 and 3 saves transportation cost and also reduces response times.

In addition, this chapter also addresses other important issues such as the stochastic nature of the damage due to disaster and the effect of a disaster on multiple cities. We address these issues by explicitly modeling the damage a disaster causes to the cities and facilities in its vicinity as a random variable that is correlated to the location of the disaster. This modeling approach is based on the intuition that locating a disaster response facility very close to a high risk city or population region may not be optimal as the facility itself might be damaged when needed. This distance-dependence is a reasonable assumption because typically the damage from natural and man-made disasters are highest closest to the primary impact of a disaster such as the epicenter of an earthquake or the track of a hurricane [Schultz et al., 2007].

In this chapter, we formulate the distance-dependent large scale emergency pre-positioning model, and we also provide a solution algorithm. This solution algorithm is an improvement over Benders decomposition. Our algorithm is aimed at solving the pre-positioning models developed in this chapter, but can also be applied to a much larger class of location problems. We use a case study on earthquakes in the state of California to show the performance of our model and to demonstrate the necessity of incorporating the modeling improvements for locating disaster response facilities.

The remainder of this chapter is organized as follows. In Section 4.2, we present an overview of the existing literature. Section 4.3 provides an improved formulation to the problem that considers the effect of a disaster on the facilities and population centers close by, while Section 4.4 provides effective solution algorithms for solving the models. Section 4.5 provides a numerical study of the new model, and Section 2.7 concludes this chapter with a discussion of the contribution of this paper, as well as future research directions.

4.2 Literature Survey

The literature on facility location under uncertainty can be broadly classified into two categories: models where the facilities are constantly in use, like warehouses, and models where facilities come into use after some rare event, such as emergency supply warehouses being used after an earthquake. In the first category, it is reasonable to have models in which the functioning of a facility is independent of externalities like demand. However, the same does not hold true for the second category. Here we have to couple the functioning of a facility with the rare event that causes a demand.

The literature on facility location under uncertainty is fairly advanced for the first category of problems that were discussed in the preceding paragraph, as can be seen from [Berman et al., 2003, 2007, Snyder and Daskin, 2005]. Berman et al. [2003] discuss the location of facilities whose reliability is dependent on the distance between the facility and the demand point. However, their model was developed for a constant demand class of problems. The chance of providing uninterrupted service goes down as the distance increases. They take into account the uncertainty in roads and transportation links being available and functioning, but they do not take into account the functioning of the facility itself. This is a realistic assumption since these models were designed for a firm providing constant service. However, this assumption makes these models less suitable for emergency facility location. The difference to the emergency facility location problem also becomes evident in their assumption that the probability of a facility not being able to provide service is zero at distance zero, and is a monotonically increasing function of distance. This is in complete disagreement with an emergency scenario, where at distance zero, the facility tends to be severely damaged and will not serve.

Berman et al. [2007] describe a p-median model using a different definition to consider uncertainty in the working of facilities. They define r_j to be the probability that the facility is working at any point of time. Using this setup, they show that co-location of facilities (locating facilities next to each other, or over each other) is a phenomenon observed in such cases. As the failure probability grows, facilities become more and more centralized. Snyder and Daskin [2005] define the reliability k-median problem to incorporate uncertainty into classical facility location models. These models have been discussed keeping a supply chain in mind, and do not apply directly to an emergency facility location problem.

We now review the available literature in the second category of problems. Jia et al. [2007] use k-median and k-center models to solve to emergency facility location problem under uncertainty. They account for reduction in service capability through a parameter p_j in their model, which is defined as the fraction of full capacity a facility j is working at in the event of a disaster. However, p_j is taken to be independent of external factors such as the location of disaster itself. Paul and Batta [2008] present a model for hospital location that is similar to [Jia et al., 2007]. They also use a factor f_k to denote the fractional capacity of a hospital located at site k in the wake of a disaster. However, f_k is taken to be independent of the disaster even here. We build on their model and discuss it in detail in the next section.

Murali et al. [2009] extend the model from [Jia et al., 2007] by making the coverage a facility can provide dependent on the distance from the demand point. They argue that the farther a facility is from the demand point, the lesser the chances of providing proper coverage due to damage to the transportation infrastructure. Rawls and Turnquist [2010] discuss a scenario based location and flow model. Each scenario is characterized by changing arc capacities resulting in uncertainties in the flow that can go through the network. Although damage to the transport infrastructure is considered here, damage to the facility itself is not. The difference is important. When damage to the transport infrastructure is considered, service availability decreases as distance of facility from disaster increases. This is because the likelihood of roads between facility and disaster being damaged is higher when facilities are far away. On the other hand, when we model a facility not working, the service availability increases as distance of a facility from disaster increases. This is because damage to facility increases as distance from disaster decreases.

Duran et al. [2011] present a model to place warehouses to minimize the average of weighted response times for different demand scenarios that might occur in a disaster. Balcik and Beamon [2008] provide a deterministic covering model for determining the locations and capacities of facilities for pre-positioning supplies. Both papers, however, assume that the facilities will keep functioning at their full capacity.

Barbarosoglu and Arda [2004] provide a stochastic programming model to plan the transportation for emergency supplies in the event of an earthquake. The model is quite descriptive with scenarios based on the location and impact of the earthquake. However, this is not a facility location model, but a multi commodity flow model, and the authors are concerned with routing the supplies given the positions of the facilities. Mete and Zabinsky [2010] provide another stochastic model for determining quantities of backup supplies hospitals should store on-site and in warehouses to counter emergencies, but their model does not consider damage to facilities and is not suited for large scale disasters.

As far as we are aware, there are no models in the literature on emergency facility location that also consider distance dependent facility functioning. Jia et al. [2007] and Paul and Batta [2008] provide models that we believe come closest to achieving this goal. However, they do not associate the functioning of a facility with the location of the disaster. In addition, most of these models assume that a disaster will only affect one city or population center at a time. This is not necessarily true for a large natural disaster, the effects of which can be very widespread. More recently, the Verma and Gaukler [2011] presented models and preliminary results that overcame these issues. We aim to fill that gap with the models and a more comprehensive analysis described in this research.

4.3 Mathematical Formulation

In this section we present three models that incorporate the assumption of the scenario based damage to disaster response facilities and population centers. The first is a deterministic model that is used to study the effects of our new assumptions on the location of the disaster response facilities. The next model is a 2-stage stochastic programming model that considers the damage caused by a disaster to disaster response facilities and population centers to not only be scenario dependent, but also stochastic. The second model is more realistic, and presents computational issues while solving for very large instances. From this point of view, both models have their own utility providing trade-offs between more realistic representation of reality and computational feasibility. An extension to the stochastic programming model which incorporates risk into the equation is also presented.

4.3.1 Deterministic Model

The model presented in this section is considerably different from related models in literature in several ways. First, we do not assume that a disaster can only occur on population centers. In our formulation, epicenters can be located anywhere. Secondly, the damage to facilities has been explicitly made dependent on the location of disasters. Lastly, in case of a disaster, demand is not constrained to be generated from just one population center, but all of them depending on their distance from the disaster's epicenter. With these changes we are able to more realistically consider disasters that are devastating enough to encompass a large area.

Let I be the set of population centers (demand points) in the region under consideration, F be the set of possible facility locations, and E be the possible epicenters given by a forecasting mechanism [Rundle et al., 2003]. In the model provided below, for simplicity and without loss of generality, the demands D_i at demand point $i \in I$ and capacities c_f of a facility $f \in F$ are represented in the same units. These units could represent, for example, the number of people whose needs have to be fulfilled if there is one emergency supply packet stored per person to be served. The probability of a disaster occurring at and epicenter $e \in E$ is denoted by w_e. The quantity $\bar{d_{if}}$ denotes the travel distance from facility f to demand i, whereas d_{ex} denotes the euclidean distance between e and $x \in I \cup F$, and is used to estimate the damage at x due to a disaster at e. The parameter p_{ex} denotes the damage at $x \in I \cup F$ as a fraction of the total demand/capacity. It should be noted that given d_{ex}, we assume that we can obtain p_{ex}. k denotes the upper limit on number of facilities to be built.

The binary variable x_f denotes the decision to open a facility at location $f \in F$. The variable y_{eif} denotes the amount of service provided by facility f to demand point i when a disaster occurs at e. The formulation is given below is referred to it as the *distance-damage* model.

$$\text{Min} \quad \sum_{e\in E}\sum_{i\in I}\sum_{f\in F} w_e d_{if} y_{eif} \tag{4.3.1}$$

$$\text{Subject to} \quad \sum_{f\in F} x_f \leq K \tag{4.3.2}$$

$$\sum_{i\in I} y_{eif} \leq (1-p_{ef})c_f x_f \qquad \forall e \in E, \forall f \in F \tag{4.3.3}$$

$$\sum_{f\in F} y_{eif} \geq p_{ei} D_i \qquad \forall e \in E, \forall i \in I \tag{4.3.4}$$

$$x_f \in \{0,1\} \qquad \forall f \in F \tag{4.3.5}$$

$$y_{eif} \geq 0 \qquad \forall e \in E, \forall i \in I, \forall f \in F \tag{4.3.6}$$

The objective function (4.3.1) is the expected transportation cost over all disaster scenarios assuming the costs are linear in the distance that has to be traveled and the amount of supplies to be shipped. This linearity assumption is common in literature. Constraint (4.3.2) limits the number of facilities built. Constraints (4.3.3) limit the total supply a facility can provide to all the demand points that need supplies. Constraints (4.3.4) make sure all the demands under a scenario are satisfied. Constraint sets (4.3.5) and (4.3.6) are binary and non-negativity constraints respectively.

To quantify the extent of damage on facilities and population centers alike, we define the notion of *expected conditional damage*, $p_{ex} \in [0,1]$, at a point $x \in I \cup F$ (population center or facility location) in the event of a disaster at point e (e.g., the epicenter of an earthquake). $p_{ex} = 0$ means that no damage is expected at point x when a disaster occurs at e and $p_{ex} = 1$ means that everything is damaged at x. The exact function could be changed depending on the circumstances under which the model is being used. For example, in our computational experiments in Section 4.5, we use a distance-damage function proposed by B.F. Howell and Schultz [1975].

As in the models present in literature, we consider a demand to be generated only when a disaster occurs at one of the possible locations. Hence, a scenario essentially specifies the point where a disaster has occurred and the effects of the disaster on the facilities and population centers nearby. Thus, a scenario e is completely defined by the location of the disaster; the demand at each of the cities under the scenario, $p_{ei}D_i, i \in I$; and the impact of the disaster on the disaster response facilities, $p_{ef}c_f, f \in F$. Note that damage to the facilities and population centers is dependent on each scenario, and more than one population center can be affected in a disaster scenario.

Intuitively, the modifications discussed in the beginning of this section are expected to result in significant changes in the locations of facilities. Recognizing the impact of the disaster on the facilities themselves, we would like facilities to be at a safe distance away from the disaster sites that affect large populations. Moreover, the model would place more facilities around areas with large accumulation of population centers that can generate large demands collectively.

Our formulation does not directly consider potential damage to the transportation infrastructure, e.g. damage to roadways between emergency response facilities and population centers. We feel that the safety of the facilities and the supplies stored there is of higher importance for a location problem than the distribution aspect. Indeed, if the emergency response facilities are substantially damaged (and thus supplies are destroyed), the distribution problem becomes moot. In addition, typically there are multiple transportation modes available for supplies in emergency situations. For example, if roadways are destroyed, following an earthquake, supplies can be transported via air. During hurricane Katrina and the ensuing flooding, the majority of supplies were transported by boat. Thus, if alternate (high-cost) transportation modes are available, we believe that the functioning of facilities and the safety of supplies, rather than the availability of the primary transportation mode, ought to be the main determinant of suitable locations for emergency response facilities.

4.3.2 The Stochastic Distance Dependent Model

To incorporate the stochastic nature of damage caused by a disaster, we now present a two-stage stochastic programming model with binary first stage where the random variable $\tilde{p}_{ex}$, which denotes the damage at $x \in I \cup F$ as a fraction of the total demand/capacity, replaces p_{ex}. It is assumed that the distribution of $\tilde{p}_{ex}$ is known or can be obtained.

In the model presented below, all the variable and parameter definitions remain the same as for the deterministic model (4.3.1)-(4.3.6). The first stage decision is to choose the locations of a given number (k) of facilities. Once a disaster occurs, the demands and reduced capability of the facilities are known, and the second stage decisions are the amounts to be routed from the opened facilities to the demand points for each disaster scenario.

$$\min \qquad E_{\tilde{p}}[f(x,\tilde{p})] \tag{4.3.7}$$

$$s.t \qquad \sum_{f\in F} x_f \leq k, \qquad x \in \{0,1\}^{|F|} \tag{4.3.8}$$

Where for a particular realization p of damage $\tilde{p}$, we have

$$f(x,p) = \sum_{e\in E}\sum_{f\in F}\sum_{i\in I} w_e \bar{d}_{if} y_{eif} \tag{4.3.9}$$

$$s.t. \quad \sum_{i\in I} y_{eif} \leq (1-p_{ef})c_f x_f \qquad \forall e\in E, \forall f\in F \tag{4.3.10}$$

$$\sum_{f\in F} y_{eif} \geq p_{ei} D_i \qquad \forall e\in E, \forall i\in I \tag{4.3.11}$$

$$y_{eif} \geq 0 \qquad \forall e\in E, \forall i\in I, \forall f\in F \tag{4.3.12}$$

Note that an expectation over all the possible disaster scenarios is already implicit in the objective function (4.3.9) of the subproblem. The only random variables we take an expectation over in the objective function (4.3.7) of the master problem are the damages to facilities and cities in each of the disaster scenarios.

4.3.3 The CVaR Distance Dependent Model

The models described till now aim to minimize the expected damage over the possible disaster scenarios. However, for large scale disasters that are rare events, it may be prudent to locate facilities for worst case scenarios. Building models that do this by incorporating some measure of risk that penalizes the worst case scenarios more may prove to be valuable.

A useful concept in this regard is the conditional value at risk (CVaR) [Rockafellar and Uryasev, 2000]. Given a fraction α, CVaR$_\alpha$ is the cost of the worst α scenarios. From our point of view, this is useful perspective since we want to locate facilities such that the relief supplies can be provided well in even the most catastrophic events. [Rockafellar and Uryasev, 2000] show that CVaR is of a convex function is also a convex function of the decision variables, and thus can be optimized in a manner similar to optimizing expectation. The following model, which aims to minimize the CVaR using two-stage stochastic programming is inspired from [Künzi-Bay and Mayer, 2006].

$$\min \quad z + \frac{1}{1-\alpha} E_{\tilde{p}}[f(x,\tilde{p})] \tag{4.3.13}$$

$$s.t \quad \sum_{f \in F} x_f \le k, \quad x \in \{0,1\}^{|F|} \tag{4.3.14}$$

Where for a particular realization p of damage $\tilde{p}$, we have

$$f(x,p) = \sum_{e \in E} w_e v_e \tag{4.3.15}$$

$$s.t. \quad v_e \ge \sum_{f \in F} \sum_{i \in I} \bar{d}_{if} y_{eif} - z \qquad \forall e \in E \tag{4.3.16}$$

$$\sum_{i \in I} y_{eif} \le (1 - p_{ef}) c_f x_f \qquad \forall e \in E, \forall f \in F \tag{4.3.17}$$

$$\sum_{f \in F} y_{eif} \ge p_{ei} D_i \qquad \forall e \in E, \forall i \in I \tag{4.3.18}$$

$$v_e \ge 0 \qquad \forall e \in E \tag{4.3.19}$$

$$y_{eif} \ge 0 \qquad \forall e \in E, \forall i \in I, \forall f \in F \tag{4.3.20}$$

Intuitively, this model locates facilities in safer locations than the deterministic model to safeguard against the worst case scenarios where the damage to facilities and demand points is very high.

4.4 Solution Method

The deterministic formulation (4.3.1)-(4.3.5) is solved by using the CPLEX 12 mixed integer programming solver. However, to solve the stochastic formulation (4.3.7)-(4.3.12), a modified L-shaped method that optimizes the Sampling Average Approximation (SAA) of the stochastic program is used. In SAA, a stochastic program with a large number of scenarios is approximated by sampling the random variables and generating a small representative set of scenarios. Multiple approximations are generated and solved using

Algorithm 8 to obtain a pool of approximate solutions. These solutions are then tested over a larger set of sample scenarios to identify the solution that performs the best on a much closer approximation of the stochastic program. In our computations, three realizations of 500 samples each are first generated for initial approximation, and each of the three is solved using Algorithm 8. The solutions obtained are then tested on an approximation with of 10,000 sample scenarios. A similar methodology based on SAA is used to solve the CVaR minimization problem.

Algorithm 8 is a modification of the usual L-shaped method (which is based on Benders algorithm [Benders, 1962]). In the regular Benders method the master problem is optimized to obtain a lower bound on the overall optimal, and the solution supplied to the subproblems. With each iteration, the master program grows in size due to the Benders cut supplied by the sub problem, and solving the master problem to optimality becomes computationally prohibitive. Cote and Laughton [1984] overcome this problem by using any feasible solution to the master problem. However, their algorithm suffers from slow convergence since even the bad solutions are evaluated as long as they are feasible. Poojari and Beasley [2009] use a variation where they solve the master problem in each iteration, but also use a genetic algorithm to create a pool of good solutions and add optimality cuts to the master problem. In this manner, the number of times the master problem has to be solved is reduced by a constant factor. Some researchers have introduced a variant where the master problem is always solved using a heuristic, completely eliminating the need for an exact method, but this approach cannot guarantee optimality of the overall algorithm. Holmberg [1994] provides a good survey of methods that solve the Benders master program approximately.

Algorithm 8 `Modified L-shaped`: Two-stage Stochastic Programs with binary first stage

```
1: $x^0 \leftarrow$ initial feasible solution
2: $LB \leftarrow -\infty$
3: $UB \leftarrow \infty$
4: while $UB - LB > \varepsilon$ do
5:     Solve subproblems using $x^t$.
6:     Update UB.
7:     Add cut $\eta + \beta_t^T x \geq \alpha_t$ to master problem.
8:     Heuristically solve master problem to obtain $x^{t+1}$.
9:     if $x^t = x^{t+1}$ then
10:        Optimally solve master problem and update $x^{t+1}$.
11:        Update LB.
12:    end if
13: end while
```

$$\min \quad \eta \tag{4.4.1}$$

$$s.t \quad \sum_{f \in F} x_f \leq k \tag{4.4.2}$$

$$\eta + \beta_t^T x \geq \alpha_t \qquad t = 1 \ldots N \tag{4.4.3}$$

$$x \in \{0,1\}^{|F|} \tag{4.4.4}$$

In our approach, the master program (4.4.1)-(4.4.4) is solved using a heuristic without updating the lower bound until the heuristic provides the same solution in two successive iterations. The rationale for doing this is two-fold: 1) if the solution to the heuristic repeats, the same cut will be generated again and the heuristic will keep generating the same solution endlessly; 2) a good heuristic will provide distinct solutions in successive iterations unless the Benders algorithm is close to optimality, thus reducing the number of times an exact algorithm has to be used considerably. Since the heuristic is not guaranteed

Algorithm 9 `Master Problem Heuristic`

```
1: $r_t \leftarrow \alpha_t, t = 1 \ldots N$
2: for $i = 1 \ldots k$ do
3:     Choose $f$ such that $x_f = 0$ and $\max_{t=1 \ldots N}(r_t - \beta_{tf})$ is minimized.
4:     Set $x_f \leftarrow 1$
5:     Set $r_t \leftarrow r_t - \beta_{tf}, t = 1 \ldots N$
6: end for
7: while $g$, $h$, $i$ & $j$ exist such that $x_g = x_h = 0$ and $x_i = x_j = 1$ and
        $\max_{t=1 \ldots N}(r_t - \beta_{tg} - \beta_{th} - \beta_{ti} - \beta_{tj}) < \max_{t=1 \ldots N} r_t$ do
8:     $x_g \leftarrow 1, x_h \leftarrow 1, x_i \leftarrow 0, x_j \leftarrow 0$
9: end while
```

to find the optimal solution to the master problem, the lower bound value is updated only when the master program is solved to optimality using an exact method.

Since the master problem's search space is finite, it can be claimed that the Algorithm 8 will converge to the optimal if it doesn't loop between solutions to the master problem indefinitely. Next, we claim that if the heuristic used to solve the master program is consistent in finding the same solution for the master problem irrespective of the ordering of the constraints, then loops caused by the heuristic returning the same sub-optimal solutions will be avoided by the condition in line 9 of the algorithm. This is because if any solution repeats, the cut generated by the dual coefficients will be the same as the one generated in the previous iteration when the repeating solution was explored. Thus, the new master problem will not have any new constraints added, and the heuristic will return the same solution in two consecutive iterations – a condition that line 9 of Algorithm 8 is designed to break by solving the master problem to optimality and obtaining a completely new solution.

A greedy solution technique, supplemented by local search described in Algorithm 9 is used as the heuristic solution technique for solving the master problem (obtained after N iterations of Benders algorithm) of the stochastic programming model. For the CVaR minimization, the master program is considerably different, and CPLEX is used with a time limit of 3 seconds to obtain a heuristic solution quickly.

We find that using Algorithm 8 in conjunction with Algorithm 9 reduces the time spent in solving the master program by an order of magnitude. Computational experiments show a speed up of at least two times using the modified algorithm over regular Benders approach, with solving the 500 scenario subproblems becoming the bottleneck for further speedup.

Although the algorithmic modifications in this section were designed for the models presented here, the essential underlying principles are fairly general. These modifications can be applied to any general problem where Benders decomposition is applicable and solving the reduced master program in each iteration is the bottleneck. In fact, the modifications only require a good heuristic to be designed for the reduced master problem.

4.5 Case Study: Pre-Positioning for Earthquake Damage

To obtain computational insight into our model formulation, we consider earthquakes as a threat for which we are pre-positioning facilities. Earthquakes are chosen as an example of a large-scale emergency situation because they recur fairly frequently in certain regions, they have the potential to devastate large areas, and, from the perspective of our modeling approach, their distance-damage functions have been measured and are empirically well understood [B.F. Howell and Schultz, 1975]. Furthermore, fairly advanced literature is available in probabilistic earthquake forecasting, which can provide us with possible earthquake epicenters as required by our model [Rundle et al., 2003].

We use a case study on California to illustrate the effectiveness of our model. We use the 20 largest cities (by population) in California as the demand points that might require emergency supplies in case of an earthquake, with demands proportional to their respective populations. The same 20 cities, as well as 38 additional grid points with integer longitude and latitude values spread across California are chosen as potential facility locations.

We place a secondary facility in Utah to serve any remaining demand that is not served by the facilities in California due to unavailability of supplies. In reality, either a single central facility or a network of secondary facilities will have to provide supplies in case

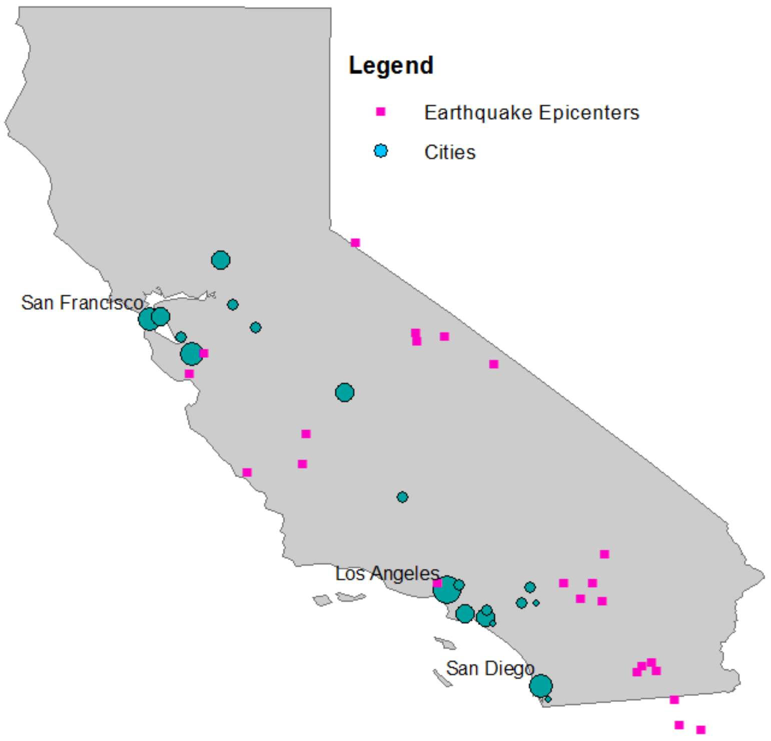


Fig. 4.2.: Parameter setting showing the demand points (20 largest cities in California) and the disaster scenarios (23 earthquakes of magnitude larger than 6 that occurred in California since 1973).

the primary facilities fail to do so. Using this artificial secondary facility allows us to incorporate some notion of a service level objective in our model: we can now compare the percentage of affected population served by primary facilities in different models, rather than observing feasibility of a solution in one model versus infeasibility in another model. For implementation purposes, the variable corresponding to this secondary facility is fixed to one to indicate that it is open, and the costs associated are multiplied by a factor of 100 to penalize unavailability of supplies in the k primary facilities located in California. The rationale behind assigning a large penalty factor to supplies availed from the secondary facility is that ideally we would prefer all the supplies to come from the k facilities within California. Supplies coming from the secondary facilities are not only delayed, but also undermine the purpose of locating facilities in California. In addition to the extra trans-

Table 4.1: Comparison of the expected cost of providing supplies to disaster affected areas in million people-miles using the optimal locations found by the stochastic model (Z_{STO}), the deterministic model (Z_{DET}) and the k-median model without conditional availability (Z_{KM}). The percentage of demand that is met by the secondary facility in Utah is also presented. An estimate of the optimality gap of the sampling average approximation and its standard deviation is also provided. The time in the rightmost column is for the whole sampling average approximation procedure. (P_{STO}, P_{DET} and P_{KM}).

#	Cap	k-Median		Deterministic		Stochastic				
Facility	(mil)	Z_{KM}	P_{KM}	Z_{DET}	P_{DET}	Z_{STO}	P_{STO}	Gap	σ_{Gap}	Time(s)
	2	34.29	50.54	27.17	39.71	27.17	39.71	0.06	0.056	109.89
	2.5	27.34	40.34	19.34	28.04	19.34	28.04	0.05	0.062	111.53
1	3	21.54	31.78	13.14	18.75	13.14	18.75	0.02	0.043	112.72
	3.5	16.93	24.94	8.54	11.85	8.54	11.85	0.03	0.044	127.87
	2	10.62	15.57	5.42	7.24	5.29	7.00	0.02	0.042	123.79
	2.5	5.75	8.34	2.23	2.51	1.98	2.01	0.03	0.017	162.97
2	3	3.23	4.61	2.18	2.86	0.97	0.51	0.01	0.010	634.62
	3.5	1.93	2.69	1.48	2.00	0.65	0.14	0.02	0.003	958.05
	2	2.75	3.97	1.35	1.64	0.94	0.54	0.02	0.009	669.35
	2.5	1.14	1.58	1.07	1.45	0.40	0.25	0.01	0.010	4147.92
3	3	0.55	0.70	0.52	0.66	0.22	0.04	0.02	0.002	11643.92
	3.5	0.29	0.33	0.28	0.31	0.16	0.04	0.01	0.002	19140.37

portation cost, the factor penalizes the additional response time that will be required to obtain supplies from the secondary facility.

The earthquake scenarios are constructed by using historic data of the 23 earthquakes of magnitude larger than 6 on the Richter scale that occurred in and around California since 1973 [see USGS, Last accessed: May 2011][1]. An expectation over these 23 scenarios (assumed equiprobable) is taken in the second stage of the stochastic programming formulation. The earthquake epicenters and 20 demand points are shown in Figure 4.2.

We assume that the probability of occurrence of an earthquake at any of the potential epicenters is the same. Since the candidate epicenters are taken directly from historical epicenter locations of high-impact earthquakes, this assumption effectively models the fu-

[1]The data used is available at http://people.tamu.edu/~anuragverma/SEFLdata

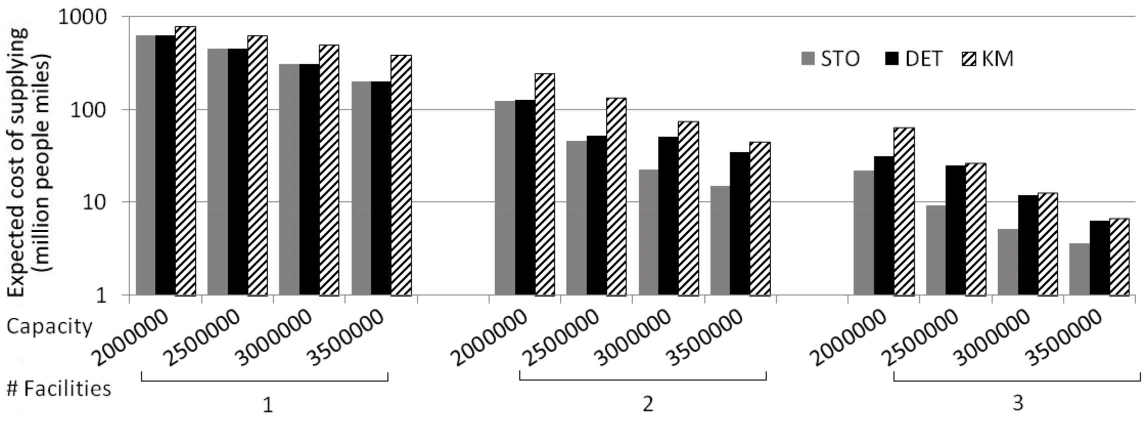


Fig. 4.3.: Expected cost of providing supplies to disaster affected areas in million people-miles using the optimal locations found by the stochastic model (Z_{STO}), the deterministic model (Z_{DET}) and the k-median model without conditional availability (Z_{KM}) plotted on the logarithmic scale.

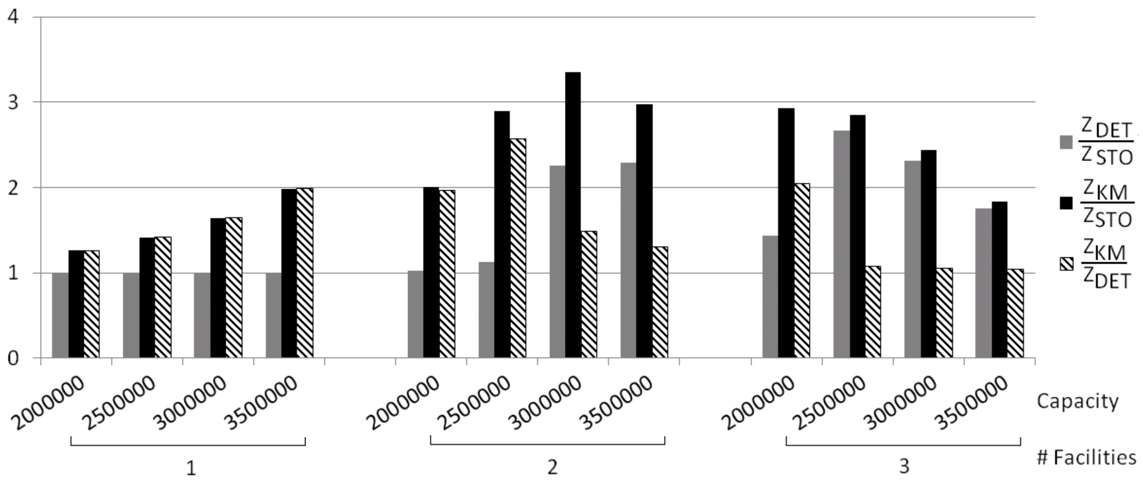


Fig. 4.4.: Ratios of the cost of providing supplies to disaster affected areas as obtained by the 2-stage stochastic model, the deterministic model, and the k-median model.

ture earthquake location distribution. The number of facilities is varied from 1 to 3. The capacity of each facility is varied between 2, 2.5, 3 & 3.5 million units. The capacity of a facility can be thought of in the same unit dimension as the demand points. For exam-

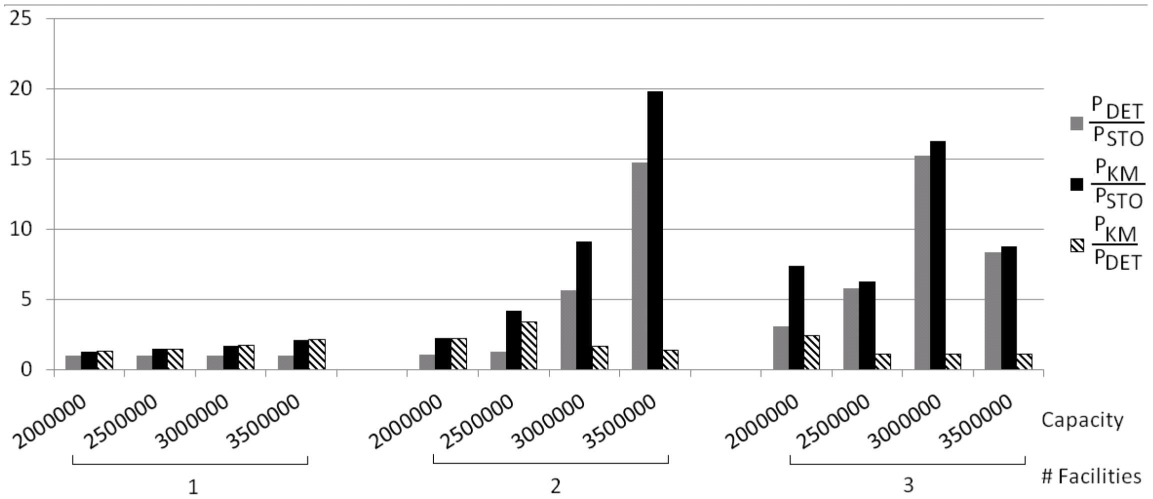


Fig. 4.5.: Ratios of the demand that had to be satisfied by the secondary facility when the primary facilities were located using the 2-stage stochastic model, the deterministic model, and the *k*-median model.

ple, the capacity could be measured by the amount of ready-to-deploy disaster response packages containing food, water, first aid supplies etc., that are stored at the facility.

The damage due to an earthquake is modeled using the intensity-distance function provided in [B.F. Howell and Schultz, 1975]. For the purpose of our computational study, we assume the damage to be proportional to the intensity, and be given as a function of the distance from the epicenter as $p_{ex}(d_{ex}) = 0.69e^{(0.364-0.130\ln(d_{ex})-0.0019d_{ex})}$ where d_{ex} is measured in kilometers [B.F. Howell and Schultz, 1975]. To incorporate stochasticity of earthquake damages, $\tilde{p}_{ex}$ is modeled as a random variable that can take on the values p_{ex}, 0 and $2p_{ex}$ with equal probability. Note that p_{ex} is the corresponding damage in the deterministic model, and $E[\tilde{p}_{ex}] = p_{ex}$.

Table 4.1 provides the expected cost of serving the affected population in case of a disaster using the locations found by the stochastic and deterministic models (Z_{STO} and Z_{DET}) developed in this chapter, and also a pure *k*-median model on which many models in literature are based (Z_{KM}). We observe that as the total built capacity increases (either because the number of facilities or the capacity increases), the expected costs go down.

This is apparent in Figure 4.3 that plots Z_{STO}, Z_{DET} and Z_{KM} against various configurations of capacity and the number of facilities built. The same trend can be observed in the percentage of the population that has to be served by the external facility in Utah on an average across all disaster scenarios (P_{STO}, P_{DET} and P_{KM}). Each of P_{STO}, P_{DET} and P_{KM} decrease as the built capacity of the facilities is increased. Both the trends are fairly intuitive, since increasing overall built capacity should reduce the costs and the need to avail the services of the secondary facility.

Some more insightful observations can be made when the costs and percentage demand met by secondary facility are compared between locations found by the stochastic, deterministic and k-median models. To this end, in Figures 4.4 and 4.5 we provide some measures that are more demonstrative of the effects of using the different models. For example, Figure 4.4 provides plots of Z_{DET}/Z_{STO}, which measures the effect of using the stochastic model against the use of the deterministic model, Z_{KM}/Z_{STO}, which draws a similar comparison with the k-median model instead, and Z_{KM}/Z_{DET}, which compares the k-median model to the deterministic model. Although in general the stochastic solution does better overall in these metrics, we observe that there exists a range of built capacity where the stochastic solution offers particularly significant improvement over the deterministic and k-median solutions. In this range of capacity, the stochastic solution is three to four times better than the k-median solution and 2 times better than the deterministic solution when the objective values are compared.

The same trend is also apparent in Figure 4.5, which plots P_{DET}/P_{STO}, P_{DET}/P_{STO}, and P_{DET}/P_{STO} for different numbers and capacities of the facilities. When the build capacity is on the lower end or the higher end, both Z_{DET}/Z_{STO} and P_{DET}/P_{STO} are close to 1, indicating that both deterministic and stochastic models are providing similar results. This is because when the total built capacity is very low when compared to the demands, there is not much difference between locations found by the deterministic and stochastic models because both models try to place facilities as far away from the disasters to preserve supplies, and obtain more or less the same solutions. On the other hand, when the total built capacity is very high, both the models locate facilities at similar locations since the stochasticity of the damage is absorbed by the high built capacity. Thus, in both cases, the stochasticity is not the primary factor determining the locations. The stochasticity comes into play when the built capacity is such that its utilization by the potential demand is medium to high. In such a case, the stochastic solution is almost two times better than the deterministic solution.

Figure 4.4 and 4.5 also provide Z_{KM}/Z_{DET} and P_{KM}/P_{DET} which show that while in general the deterministic solution is better than the k-median solution, there is a particular range of total built capacity in which the deterministic model is almost two times better than the k-median solution in terms of the objective function. This can again be attributed to the fact that when the built capacity is too low, none of the models is able to provide good locations, while when the capacity is too high, the locations found by either model are similar, because damage to facilities (which results in a reduction of capacity), becomes less relevant.

Finally, the quantities Z_{KM}/Z_{STO} and P_{KM}/P_{STO}, which provide the overall benefit over using the stochastic model over the k-median, model are in essence the combination of benefits caused by using the stochastic model over deterministic model, and the deterministic model over the k-median model. As a result, the similar trends in Z_{KM}/Z_{DET} and Z_{DET}/Z_{STO} are accentuated in the metric Z_{KM}/Z_{STO}, resulting in the stochastic solution being three to four times better in terms of the objective function value in the range where the built capacity is neither too high nor too low. The reason for this can be attributed to a combination of the reasons outlined in the previous two paragraphs - in this range, both stochasticity of the damage and the damage to facilities come into play.

It should be noted that a disaster management agency will need to build disaster response facilities with capacities that are both economical and effective. This becomes especially true in the current scenario with federal and state governments restricted in their spending, and occurrences of natural and man-made disasters on a rise. The results from Table 4.1, as well as the trends observed in Figure 4.4 & 4.5, suggest that our stochastic solution turns out to be much better than the deterministic solution in the range of capacities that is of primary interest to decision makers. Thus, the stochastic model provides much value over the deterministic models developed in literature in terms of finding good locations under realistic budget constrained conditions.

4.5.1 Effect of the CVaR Model

For the CVaR model, the same data was used with the value of α set to 0.9, indicating that we want to minimize the expected cost of the worst 10% of the scenarios. The results are presented in Figure 4.6 in the form of a comparison against the the stochastic model. The quantities compared in the chart are the overall expected cost of transportation (denoted by Z_{CVaR} and Z_{STO}, for the two models respectively), which the stochastic model minimizes, and the expected cost of transportation in the worst 10% of the scenarios ($Z^{0.9}_{CVaR}$ and $Z^{0.9}_{STO}$, respectively), which the CVaR model minimizes. The charts shown in the diagram plot the ratios Z_{CVaR}/Z_{STO} and $Z^{0.9}_{STO}/Z^{0.9}_{CVaR}$ to gauge how much of an effect optimizing one measure has on the other. It is evident that for low built capacities, there is no difference in the two models. However, as the built capacity increases, the solutions suggested by the two models start to differ considerably. With larger capacities, the stochastic model places the facilities closer to the disaster affected areas. However, the CVaR model places facilities keeping in mind the worst 10% scenarios, and places the facilities in different locations accordingly. As a result, the overall expected cost of supplying goes up, but the service in the really catastrophic disasters is much improved.

This can be further seen from the Figure 4.7, which plots the expected cost of supplying using the stochastic solution and the CVaR solution for different capacity settings in 10 different brackets - the best 10% scenarios, $10\% - 20\%$, $20\% - 30\%$, ... , and the worst 10% scenarios. This diagram makes the difference between the stochastic and CVaR models fairly clear. In Figure 4.7(b) and Figure 4.7(c) the stochastic solution does better in the best 90% of the scenarios, and thus overall, while the CVaR solution does far better in the worst 10% scenarios. This is because in essence, the CVaR model goes finds much

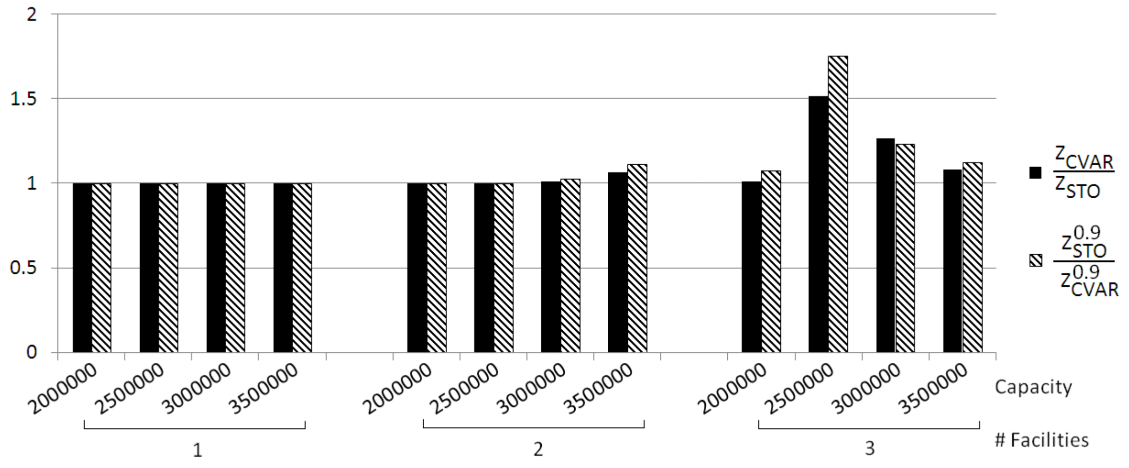


Fig. 4.6.: Ratios of the expected cost of providing supplies to disaster affected areas across all scenarios (Z_{CVaR}/Z_{STO}) and for the worst 10% scenarios ($Z^{0.9}_{STO}/Z^{0.9}_{CVaR}$) as obtained by the 2-stage stochastic model and the CVaR model.

safer locations aimed at the really catastrophic scenarios, even if it means performing a worse on the scenarios that are not very bad.

4.5.2 Analysis of the Locations Suggested by Different Models

We now analyze the locations of the facilities obtained by the three models considered by us when the capacities are in the critical range discussed in the previous paragraph. Figure 4.8 shows the locations of three facilities with capacities set to two million as found by the deterministic model and the *k*-median model. It can be observed that the deterministic model developed in this research places two facilities in similar locations, but the third facility is moved from a very earthquake prone location to a much safer location. The cost of supplying facilities from this safer location is higher, but the availability of supplies is higher than the closer but risk-prone facility. With the third facility at the location suggested by the *k*-median model, a small part of the supplies comes from a close location, but a large penalty is incurred for the remaining demand that is fulfilled from the

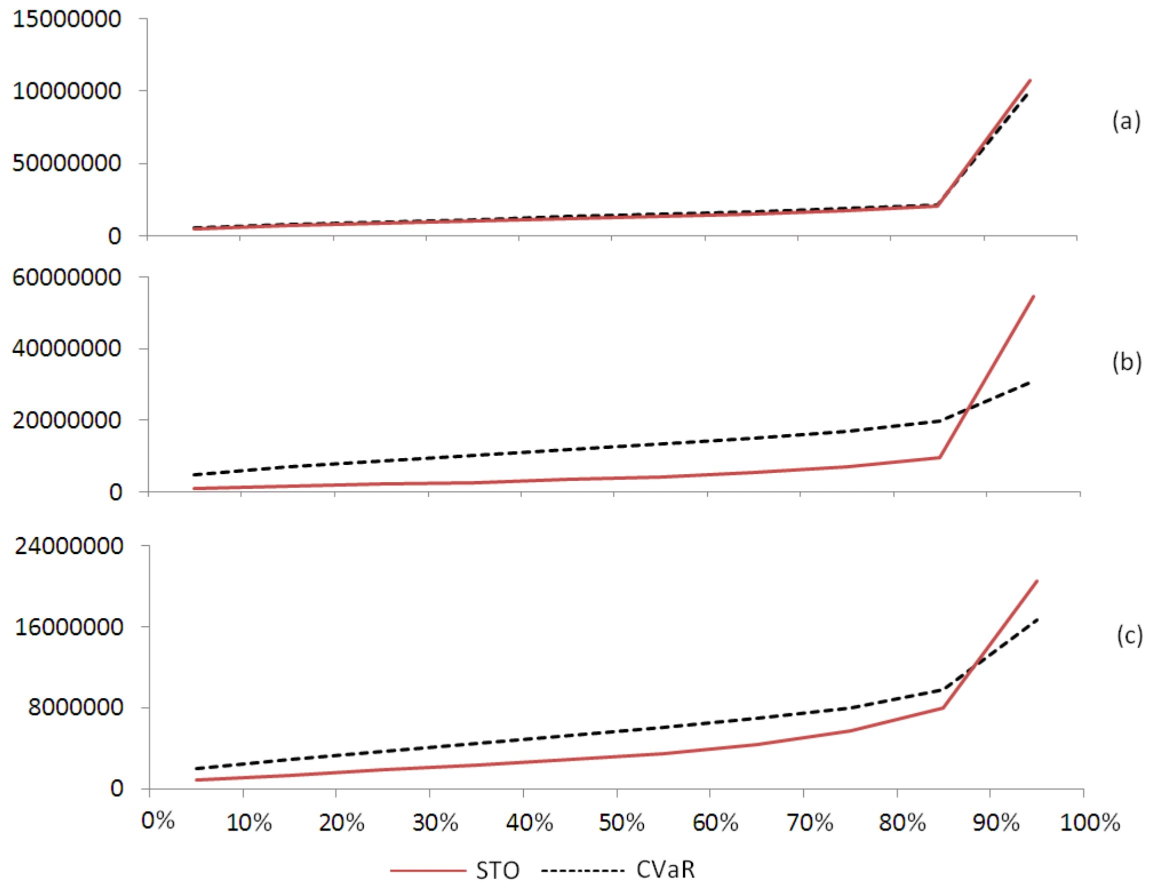
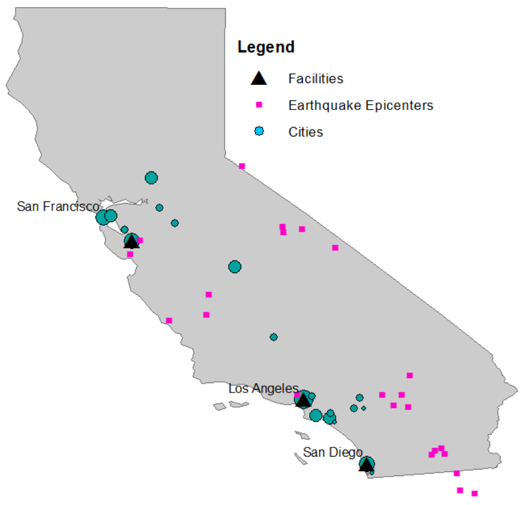


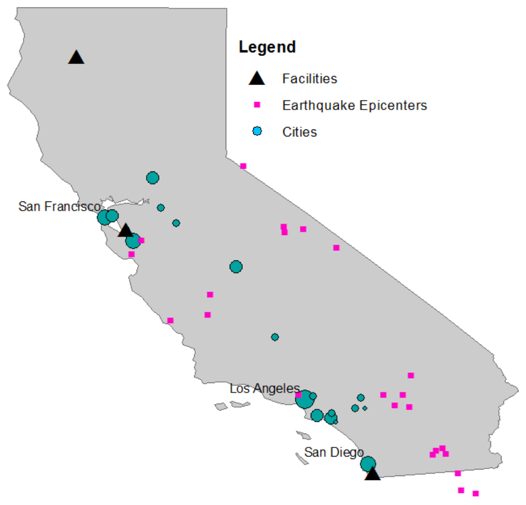
Fig. 4.7.: The the expected cost of providing supplies to disaster affected areas shown for the best 10% scenarios, 10% – 20%, 20% – 30%, ... , and the worst 10% scenarios, when three facilities with capacities (a) 2,000,000, (a) 2,500,000, and (a) 3,000,000 are placed.

secondary facility. As a result, the deterministic model finds it beneficial to pay a higher transportation cost to obtain a large part of the required supplies from the safer location.

Figure 4.9 shows the locations of facilities when the capacities are set to 3.5 million as found by the deterministic model and the stochastic model. A noticeable change in Figure 4.9(a) from Figure 4.8(b) is that as the capacities are increased, the deterministic model places facilities in areas of high demand even if they are risk prone. This is because the higher capacities result in little or no penalty even when the facilities are damaged. However, in Figure 4.9(b) the stochastic model still places the one facility far away from

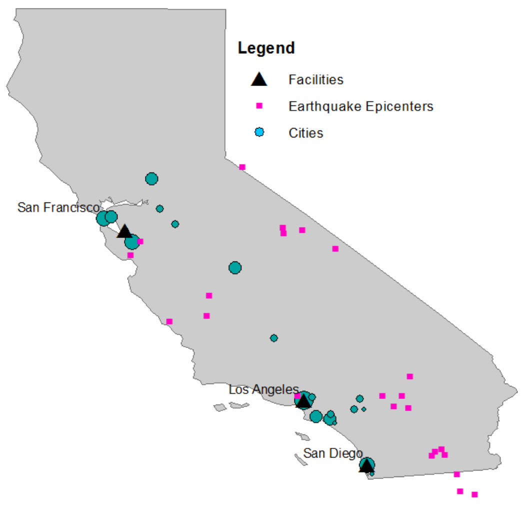


(a) *k*-Median model

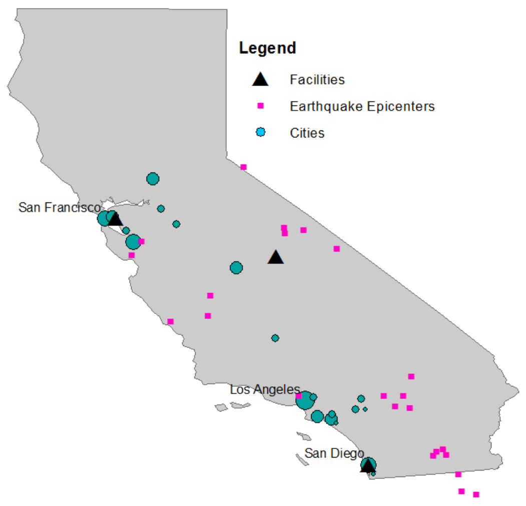


(b) Deterministic model

Fig. 4.8.: Comparison of locations of three facilities of capacities 2 million as found by the Deterministic and *k*-Median models.



(a) Deterministic model



(b) Stochastic model

Fig. 4.9.: Comparison of locations of three facilities of capacities 3.5 million as found by the Stochastic and Deterministic models.

the risky Los Angeles area. This can be attributed to the fact that the stochastic program is trying to avoid large penalties resulting from scenarios where the damage from an earthquake is high by placing facilities in relatively safer locations than the deterministic model would.

4.5.3 Effects of Varying the Penalty

Table 4.2 depicts the expected service cost as the penalty of availing supplies from the external facility from Utah is changed between 1 (no penalty), 10 (low penalty) and 100 (high penalty). Penalties act as a handle in our models using which we can ascertain the time-sensitive nature of service in the aftermath of a large scale emergency. The choice of the penalty needs to be determined by the decision maker, and will take into consideration various factors that reflect the 'cost' of not providing timely service and having to bring in supplies from an external warehouse. We believe that larger penalties are a more realistic choice in most cases since the service here is critical to a large population.

We can draw the following conclusions from the results: First, as the penalty is increased, for the same number and capacity combination, more demand is satisfied by the model that uses a high penalty. This is intuitive and expected, since a higher penalty forces the model to choose a solution that will require less services from the external facilities. Second, when there is a low penalty, or no penalty at all, the stochastic solution does not improve on the deterministic solution by much when compared to the high penalty case. This can be attributed to the fact that when the penalty is low, the cost of availing supplies from the external facility in some extremely bad scenarios is not large enough to change a good location that serves very well during "average" damage scenarios. Thus, the stochastic model also chooses locations similar to the deterministic equivalent model.

4.6 Conclusion

In the existing literature, failure of large scale emergency response facilities and occurrence of a disaster have been modeled to be independent of each other. This paper takes a

Table 4.2: Ratio of the expected cost of serving the disaster affected population by facilities located using the deterministic model and by the stochastic model as the penalty of serving using the external facility is increased.

Penalty	Capacity	Num. Facilities		
		1	2	3
	2000000	1.00	1.02	1.02
1	2500000	1.00	1.00	1.00
	3000000	1.00	1.00	1.01
	3500000	1.00	1.02	1.00
	2000000	1.00	1.05	1.09
10	2500000	1.00	1.18	1.05
	3000000	1.00	1.06	1.00
	3500000	1.02	1.05	0.99
	2000000	1.00	1.03	1.43
100	2500000	1.00	1.13	2.66
	3000000	1.00	2.26	2.31
	3500000	1.00	2.29	1.75

different approach by acknowledging the fact that facility failures will often be caused by the very disasters they are supposed to provide relief from.

The deterministic and stochastic models proposed in this paper enhance the emergency facility location literature in the following ways: they 1) provide a handle on making the availability of a facility directly dependent on a disaster, 2) allow us to consider the location of the epicenter of a disaster to be separate from the location of the demand points, and 3) allow us to model the demands as clusters of population acknowledging the possibility of more than one demand point being affected by a disaster. Furthermore, the stochastic model developed allows us to study a more realistic scenario where predicting the damage caused by a disaster is not straightforward, and thus highlights the shortcomings of the deterministic models. The paper also presents a variant of Benders Algorithm that has been found to be very effective in solving the models presented and could be extended to other facility location models.

Using a case study on pre-positioning for earthquakes in California, we demonstrate that existing models tend to produce location results that can turn out to be highly undesirable once the potential effects of the disaster on the response facilities themselves are taken into account. Furthermore, our numerical results indicate that incorporating the stochastic nature of disasters is extremely important for a cost effective relief effort.

5. A DISTRIBUTED APPROXIMATION ALGORITHM FOR THE BOTTLENECK CONNECTED DOMINATING SET PROBLEM

5.1 Introduction*

A popular and efficient method of routing and communication in wireless sensor networks (WSN) is creating a virtual backbone. *Connected dominating sets* (CDS) have been often used to describe a virtual backbone in an ad-hoc wireless network, since any node in the network is at most one hop away from the nodes in the CDS, and the nodes in a CDS are all connected and can communicate easily within themselves. Another desirable feature in the backbone is that it should be as small as possible – this helps to reduce the communication overhead in maintaining the backbone and passing information through it. Therefore, much of the literature dealing with construction of a virtual backbone in WSN is focussed on the *minimum connected dominating set* (MCDS) problem [Blum et al., 2005, Wu and Li, 1999, Das and Bharghavan, 1997, Guha and Khuller, 1996, Thai et al., 2007, Min et al., 2006, Shin et al., 2010, Wu et al., 2010]. The MCDS problem is known to be NP-hard [Garey and Johnson, 1979b]. Blum et al. Blum et al. [2005] provide an excellent survey of centralized and distributed algorithms and results regarding the use of MCDS in wireless sensor networks.

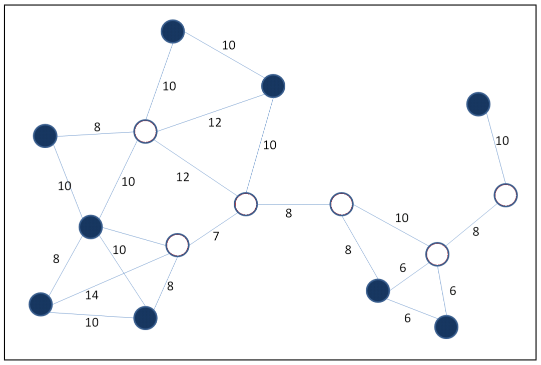
This research deals with an alternative approach to forming the virtual backbone proposed in [Butenko et al., 2011]. Instead of fixing the transmission range of the nodes and minimizing the number of nodes in a CDS, we now fix the number of nodes and try to minimize the transmission range that guarantees a CDS of the given size. This problem is of interest due to the extreme importance of energy considerations in sensor networks. The power required for supporting a communication link between nodes u and v at a distance of d_{uv} is proportional to d_{uv}^{β}, where β varies between 2 and 5 [Li and Stojmenovic,

Reprinted with kind permission from Springer Science+Business Media: Optimization Letters, A distributed approximation algorithm for the bottleneck connected dominating set problem, Online First^{TM}, 6 April 2011, P. 1-13, A. Verma and S. Butenko, DOI: 10.1007/s11590-011-0314-2.

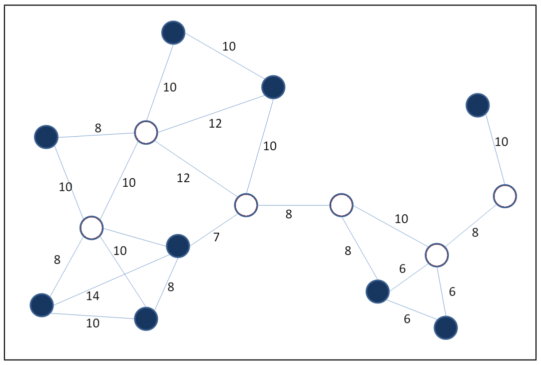
2005]. Thus, the choice of the transmission range of the nodes directly impacts the network's lifespan. Power saving schemes in WSN have been studied previously [Wu and Wu., 2003], though they consider a different approach of alternating the nodes in the CDS. Again, [Blum et al., 2005] provides an excellent survey of papers in this area.

Next, we formally define the problem studied in this chapter. Consider an edge weighted graph $G = (V, E, w)$ with non-negative edge weights given by w_{uv} for each edge $(u, v) \in E$. Weighted graphs are a natural way of representing a wireless sensor network, where the edge weights represent pairwise distances between the nodes. Given the weighted graph $G = (V, E, w)$, the unweighted graph $G(\bar{w}) = (V, E_{\bar{w}})$ with $E_{\bar{w}} = \{(u, v) : (u, v) \in E, w_{uv} \leq \bar{w}\}$ is called the bottleneck graph of G with the bottleneck $\bar{w}$. That is, in a bottleneck graph of G two vertices are connected if and only if the weight of the edge connecting them in G is no greater than $\bar{w}$. The *k-bottleneck connected dominating set* (k-BCDS) problem is to find the minimum $\bar{w}$ such that it is possible to find a CDS of size k in the bottleneck graph $G(\bar{w})$. In other words, the k-BCDS problem seeks a minimum edge weight in the graph such that the corresponding bottleneck graph has a connected dominating set of the predefined size k. By solving this problem we would be increasing the lifetime of the network by setting the transmission range to the weight of the bottleneck edge. Figure 5.1 provides an illustration of using MCDS vs k-BCDS as a virtual backbone. Observe how the maximum range a node has to transmit in this virtual backbone is reduced from 14 to 12 by using k-BCDS instead of the MCDS. This may lead to an increase in lifetime of the network by $(14/12)^{\beta}$, which is considerable given that β varies between 2 and 5 [Li and Stojmenovic, 2005].

Approximating the k-BCDS problem within a factor $2 - \varepsilon$ is shown to be NP-hard in [Butenko et al., 2011], which also provides a centralized 3-approximation algorithm for the k-BCDS problem. However, in order to apply the k-BCDS model to ad-hoc wireless sensor networks, it is desirable to develop a *distributed* algorithm such that all the nodes need only local neighborhood information. This research provides a 6-approximate algorithm that



(a) MCDS in a wireless sensor network with distances given. The white nodes mark the nodes in the CDS. Note that the maximum range a node has to transmit in this virtual backbone is 14.



(b) The k-BCDS for the given graph, with k set to 6. The white nodes mark the nodes in the CDS. Note that the maximum range a node has to transmit in this virtual backbone is 12.

Fig. 5.1.: Difference between MCDS and k-BCDS of the same size.

can be implemented in a distributed setting. The approximation result holds for networks where the edge weights satisfy the triangle inequality.

The remaining sections of this chapter are organized as follows. The proposed distributed algorithm and its approximation analysis are presented in Sect. 5.2 and 5.3, respectively. A mixed-integer programming formulation of the problem of interest is given in Sect. 5.4. Section 5.5 describes the results of sample numerical experiments, and Sect. 5.6 concludes the chapter.

5.2 Distributed Algorithm

In this section we provide a 6-approximate distributed algorithm, implementation of which involves a series of operations by each node. The algorithm uses the concept of a k-center on a weighted graph $G = (V, E, w)$. Given a graph $G = (V, E, w)$, the k-center problem is to find a set S of k vertices such that $\max_{v \in V} \min_{u \in S} \{w_{vu}\}$ is minimized. In addition, the algorithm requires each node to have a knowledge of distances from its neighbors. We note that distance estimation in ad-hoc networks has been addressed in literature [Girod and Estrin, 2001, Savvides et al., 2001, Biaz and Ji, 2005].

Before describing the main algorithm, we outline the schemes *FindMin* (and *FindMax*) for finding the minimum (or the maximum) of an attribute stored in the nodes of the network. The procedure *FindMin* (*FindMax*) starts off with each node holding an attribute $a(v)$, and terminates with each node obtaining the information about the minimum (maximum) attribute value across the network, and the knowledge of the node that had that minimum (maximum) attribute value to begin with. The basic idea of the algorithm is as follows: each node in the graph maintains the least attribute value that it has seen yet, and the identification (*id*) of the node that has it. Periodically, every node propagates this information to its neighbors. If a node receives a value from its neighbor that is smaller (larger) than the one stored by it, then the value and corresponding *id* is updated. In case of ties in attribute value, nodes with smaller *id* are favored. After at most $|V|$ such rounds of information exchange by neighbors, the minimum (maximum) attribute value would have percolated through all the nodes in the network. These procedures are modified versions of *flooding*, which is a well studied protocol for data exchange in ad-hoc wireless sensor networks [Akkaya and Younis, 2005]. The time and message complexity of these procedures is $O(|V|^2)$ assuming that a node can broadcast a message to all its neighbor at once, and that only one node can be transmitting at any time.

Our main algorithm consists of two major steps, summarized in Algorithms 10 & 11 respectively: finding a $\lfloor k/3 \rfloor$-center and connecting the $\lfloor k/3 \rfloor$-centers. The main idea of the algorithm is as follows: we first approximate the $\lfloor k/3 \rfloor$-center of the network using a distributed version of Gonzalez's algorithm [Gonzalez, 1985]. This results in formation of a spanning forest with $\lfloor k/3 \rfloor$ star-shaped trees. The centers of these stars are chosen to be in the dominating set. Next we use a distributed minimum spanning tree algorithm to connect the $\lfloor k/3 \rfloor$ centers. As we describe later in this section, at most $2(\lfloor k/3 \rfloor - 1)$ nodes are added into the dominating set to achieve this. At the end of the second step, we obtain a connected dominating set with at most $(k-2)$ nodes. We prove in the next section that the weight of the largest edge in the spanning tree formed by the CDS obtained is at most 6 times the optimal k-BCDS value.

Algorithm 10 6-approximation Algorithm for *k*-BCDS (Step 1).

```
1. Find the $\lfloor k/3 \rfloor$-Center of the network.

    1.1 **Choose the node with minimum *id* to be the first center.**
        Use *FindMin* to find the node $z$ with minimum *id*.
        Node $z$ chooses itself to be the first center, sets $d_z = 0$.
        All other nodes $v \in V \setminus \{z\}$ set $d_v = \infty$.
        All the nodes $v \in V$ set $l_v = id_z$.
    1.2 **Update the distance of all the nodes from their respective nearest center.**
        This can be achieved by repeating $|V|$ rounds of the following:
        Each node $v$ broadcasts $[d_v, l_v]$ to its neighbors.
        At receiving $[d_u, l_u]$ from a neighbor $u$, $v$ updates $d_v = \min(d_v, d_u + w_{uv})$.
        If $d_v$ is updated, $l_v$ is updated to $l_u$.
    1.3 **The node farthest away from its nearest center chooses itself to be the next center.**
        Use *FindMax* to find the node $y$ with maximum $d_y$.
        Node $y$ chooses itself to be in the $\lfloor k/3 \rfloor$ center, sets $d_y = 0$.
    1.4 **Repeat steps 1.2 and 1.3** $(\lfloor k/3 \rfloor - 1)$ **times.**
        Each node knows when to stop as it can maintain a count of times these steps were
        executed.
    1.5 **Form** $\lfloor k/3 \rfloor$ **spanning components of the network.**
        At the end of the steps 1.1-1.4, each node $v$ has its label $l_v$ set to the node *id* of its
        nearest center.
        Each node not in the $\lfloor k/3 \rfloor$-center sets its *range* to $d_v$. The nodes also broadcast $d_v$ to
        their neighbors.
        Each node in the $\lfloor k/3 \rfloor$-center receives $d_v$ from the nodes in its periphery, and sets its
        *range* to the maximum $d_v$ received.
        In this manner, we obtain $\lfloor k/3 \rfloor$ components each with a unique label in its nodes.
```

Step 1, presented in Algorithm 10 describes a distributed version of the 2-approximate algorithm to find a $\lfloor k/3 \rfloor$-center. Throughout the algorithm, d_v denotes the distance of a vertex v to the nearest center, and the label l_v denotes the *id* of the nearest center. We first use the *FindMin* algorithm to choose the node with minimum *id* to be the first center. Next we update the distance d_v of each node from its nearest center. We then use *FindMax* to find the node with maximum d_v (the node farthest from its nearest center), and choose it to be the next center. This process is repeated until $\lfloor k/3 \rfloor$ centers have been chosen. After

Algorithm 11 6-approximation Algorithm for k-BCDS (Step 2).

2. Connect the $\lfloor k/3 \rfloor$-Centers using distributed minimum spanning tree algorithm.

   2.1 **Each node finds the least cost edge connecting it to a node from a different component.**
   Each node v probes its neighborhood $N(v)$ for nodes with a different label. $T_v := \{u : u \in N(v), l_u \neq l_v\}$.
   Each node v sets $t_v := arg\min_{u \in T_v}(w_{uv})$. Ties broken by minimum node *id*.

   2.2 **For each component C, find the least cost edge between a vertex in C and another component.**
   Use *FindMin* within each component to find the minimum weight edge $(s_{min}(C), t_{min}(C))$ emanating from C. Ties are broken as described in procedure *FindMin*.
   If none of the nodes in C are able to find an edge across to a different component, STOP (we have only one component left).
   Else, for exactly one node v in each component C, (v, t_v) is the same as $(s_{min}(C), t_{min}(C))$.

   2.3 **Add the nodes associated with this least cost edge into the dominating set.**
   For every component C, the nodes $s_{min}(C)$ add themselves to the dominating set and send a message to $t_{min}(C)$ to do the same.
   The nodes $s_{min}(C)$ and $t_{min}(C)$ set their *range* to a value higher than the edge weight of $(s_{min}(C), t_{min}(C))$.
   Components joined in this manner agree upon a common label for all their nodes. Each node v updates its l_v to this value. This can be done using a scheme like *flooding* to propagate the label of one of the components to the ones that got joined to it.

   2.4 **Repeat steps 2.1-2.3.**

the algorithm finishes, each node knows whether it is in the $\lfloor k/3 \rfloor$-center or not, and if not, the nearest center (l_v) and its distance (d_v) to it. For correctness of the next step in the algorithm, we require the nodes in the $\lfloor k/3 \rfloor$-center to form a dominating set of the graph. Throughout this chapter we assume that k is sufficiently high and the initial transmission range of the nodes is set high enough in the beginning to guarantee this property. At the end of step 1, $\lfloor k/3 \rfloor$ center nodes have chosen themselves to be in the dominating set. The bottleneck edge until this step is one of the edges connecting a node to its nearest center.

Step 2, presented in Algorithm 11 is similar to a distributed minimum spanning tree algorithm [Attiya and Welch, 2004], and starts with $\lfloor k/3 \rfloor$ star-shaped components spanning all the nodes in the network. To begin with, each component has a connected dominating set (the one node at the center of the star). We join the $\lfloor k/3 \rfloor$ components by adding $(\lfloor k/3 \rfloor - 1)$ edges across these components. Both the end points of these edges are then included in the dominating set. Since all these nodes are connected to their dominating node in the original component, we get a connected dominating set of the whole graph. In the process, we add at most $2(\lfloor k/3 \rfloor - 1)$ nodes to the dominating set.

At the end of the algorithm's execution, each node has the knowledge of the neighboring nodes it has to directly communicate with. As a result, each node can set its own transmission range accordingly (using the variable *range* that was obtained in the course of the algorithm). The maximum of these transmission ranges, which may not be explicitly known to all the nodes, is the bottleneck found by our algorithm.

The stopping criteria of the loops of the algorithm (steps 1.4 & 2.2) should be carefully examined here. Each node is involved in executing the sub-steps of the algorithms synchronously, and knows when it is time to move to the next step. This is because each sub-step involves a constant number of communication rounds to take place before we can move to the next one. As a result, for step 1, each node has the knowledge of when steps 1.2 & 1.3 have been executed $(\lfloor k/3 \rfloor - 1)$ times. For step 2, the algorithm stops if no node in a component is able to find a node from a neighboring component. Since our network is connected, this implies that all the nodes form one single component, and we do not need to add any more nodes.

The choice of edges to be added is done by finding the least cost edge emanating from each component. Ties in choosing the least cost edge emanating from a component are broken using the following scheme: if two edges (p_1, q_1) and (p_2, q_2) have the same cost, then the edge with lower $\min(p_i, q_i), i = 1, 2$ is chosen. If there is still a tie, then the edge with lower $\max(p_i, q_i), i = 1, 2$ is chosen. Breaking ties in this manner avoids formation of cycles while connecting the components. The steps involved in choosing the nodes to be added to the dominating set in a distributed fashion are outlined in the algorithm.

The two steps in the algorithm add $\lfloor k/3 \rfloor$ and at most $2(\lfloor k/3 \rfloor - 1)$ nodes to the dominating set, respectively. This gives us a total of $3\lfloor k/3 \rfloor - 2$ nodes, which is less than k. The only edges added in the process are the ones for the $\lfloor k/3 \rfloor$-center, and the ones forming a connection across these components. Hence, the bottleneck edge is either one of the $\lfloor k/3 \rfloor$-center edges, or an edge connecting two components. This will be used in proving the approximation bounds of the algorithm.

To evaluate the time and message complexity, we adopt a standard synchronous model of computation [Linial, 1992]. We find a theoretical bound on the number of communication rounds our algorithm needs to perform. It should be noted that for a synchronous model, each node can exchange a message with each of its neighbors in a communication step [Jia et al., 2002]. Accordingly, we find that our algorithm has a time complexity of $O(k|V|)$. This is because steps 1.2 and 1.3 of Algorithm 10 are executed $\lfloor k/3 \rfloor$ times, and each one requires $|V|$ rounds of complete information exchange between all neighbors. Similarly, step 2.2 of Algorithm 11 is executed at most $O(\log k)$ times, and has the same time complexity of $O(|V|)$. It should be noted that if all the nodes knew the topology of G, the time complexity would have been $O(kd_G)$, where d_G is the diameter of the graph. Since this is not the case in an ad-hoc network, we assume the worst case ($d_G = O(|V|)$). For computing the message complexity, we assume that each node can send information to its neighbors in $O(1)$ messages [Das and Bharghavan, 1997]. With that in mind, we find the message complexity of our algorithm to be $O(k|V|^2)$.

5.3 Approximation Analysis

Let d_{kBCDS} be the weight of the bottleneck edge our algorithm finds, and d^*_{kBCDS} be the optimal weight. We prove that the algorithm presented above has an approximation ratio of 6 (i.e., $d_{kBCDS} \leq 6d^*_{kBCDS}$). All the results in this section are valid when triangle inequality holds in the graph.

Lemma 5.3.1. *Let the $\lfloor k/3 \rfloor$-center value found by Algorithm 10 be $d_{\lfloor k/3 \rfloor}$, and the optimal be $d^*_{\lfloor k/3 \rfloor}$. Then, $d_{\lfloor k/3 \rfloor} \leq 2d^*_{\lfloor k/3 \rfloor}$.*

Proof. Step 1 is a distributed implementation of Gonzalez's 2-approximate algorithm. The proof follows from the definition of a 2-approximation algorithm. $\square$

Lemma 5.3.2. *Under the assumption of triangle inequality, the optimal $\lfloor k/3 \rfloor$-center value is at most 3 times the optimal k-BCDS value: $d^*_{\lfloor k/3 \rfloor} \leq 3d^*_{kBCDS}$.*

Proof. Let D be the set of nodes in the optimal k-BCDS, with bottleneck value d^*_{kBCDS}. We construct a breadth-first-search (BFS) tree for the spanning tree induced by D rooted at any leaf node. Let there be $l_i, i = 1, 2, \ldots, L$ nodes in level i of the tree, where L is the total number of levels. Since $(l_1 + l_4 + l_7 + \ldots) + (l_2 + l_5 + l_8 + \ldots) + (l_3 + l_6 + l_9 + \ldots) = k$, at least one of the following three inequalities is true:

1. $l_1 + l_4 + l_7 + \ldots \leq \lfloor k/3 \rfloor$
2. $l_2 + l_5 + l_8 + \ldots \leq \lfloor k/3 \rfloor$
3. $l_3 + l_6 + l_9 + \ldots \leq \lfloor k/3 \rfloor$

If none of the three inequalities are satisfied, then by summing them up we can show that the given equality cannot be satisfied. Next, we choose the nodes in the levels in one of the satisfied inequalities to construct a $\lfloor k/3 \rfloor$-center with the bottleneck value $\bar{d}_{\lfloor k/3 \rfloor}$. Since every third layer in the BFS is chosen, any node in D is at most 2 hops away from a node in the $\lfloor k/3 \rfloor$-center thus formed. Thus, any node in V is at most 3 hops away from a node in the $\lfloor k/3 \rfloor$-center. Hence, by triangle inequality, $\bar{d}_{\lfloor k/3 \rfloor} \leq 3d^*_{kBCDS} \Rightarrow d^*_{\lfloor k/3 \rfloor} \leq 3d^*_{kBCDS}$. $\square$

Lemma 5.3.3. *If d_{kBCDS} is the weight of an edge added in Algorithm 10, then $d_{kBCDS} \leq 6d^*_{kBCDS}$.*

Proof. In step 1, only edges connecting a $\lfloor k/3 \rfloor$-center to a node on its periphery are added. Hence, from lemmas 5.3.1 and 5.3.2, it follows that $d_{kBCDS} = d_{\lfloor k/3 \rfloor} \leq 2d^*_{\lfloor k/3 \rfloor} \leq 6d^*_{kBCDS}$. $\square$

Lemma 5.3.4. *If d_{kBCDS} is the weight of an edge added in Algorithm 11, then $d_{kBCDS} = d^*_{kBCDS}$.*

Proof. For an edge with weight d_{kBCDS} to be added to k-BCDS in step 2, it must have been the least weight edge going from a component C to any other node not in C. Thus, to maintain connectivity of the graph, the inclusion of this edge (or another edge with the same weight connecting C to the remaining nodes) is necessary in the optimal solution. Hence, $d_{kBCDS} \leq d^*_{kBCDS} \Rightarrow d_{kBCDS} = d^*_{kBCDS}$. □

Theorem 5.3.1. *Algorithm 10–11 has an approximation ratio of 6, i.e., $d_{kBCDS} \leq 6d^*_{kBCDS}$ when triangle inequality holds.*

Proof. Since all edges in the k-BCDS formed are added either in steps 1 or 2, from lemmas 5.3.3 and 5.3.4 it follows that $d_{kBCDS} \leq max(d^*_{kBCDS}, 6d^*_{kBCDS}) = 6d^*_{kBCDS}$. □

5.4 Mixed-Integer Programming Formulation

To evaluate the quality of the solution obtained by the proposed algorithm in practice, it is desirable to know the optimal objective value for the considered instances of the problem. To solve the k-BCDS problem to optimality, we used a mixed-integer programming (MIP) formulation that is based on the observation that a graph $G = (V, E)$ has a CDS of size at most k if and only if it has a spanning tree with at least $|V| - k$ leaves. Given a connected dominating set D in G, a spanning tree T of G will be called *corresponding* to D if its set of leaves includes all vertices in $V \setminus D$. We will build a directed-out spanning tree rooted in vertex 1.

Let $x_i, i \in V$, be a binary decision variable indicating whether i belongs to k-BCDS, and let $y_{ij}, i, j \in V, i \neq j$, be a binary decision variable indicating whether the edge (i, j) is in a spanning tree corresponding to the k-BCDS sought. We introduce a real variable z to represent the bottleneck cost of the k-BCDS. In the the MIP formulation (5.4.1)-(5.4.16) given below, the objective, (5.4.1), is to minimize the bottleneck cost. Constraint (5.4.2) ensures that the CDS is of size at most k. The constraints (5.4.3) set the bottleneck cost to be the maximum weight of any edge in the corresponding spanning tree. Constraints (5.4.4) are present because only the vertices in k-BCDS can have outgoing edges, as the rest of them must be leaves in the corresponding spanning tree. Constraint (5.4.5) are present to ensure that the first node can have one edge going out even if it is a leaf not in the k-BCDS, and at most n edges if it is in the k-BCDS. The first node should have at least one edge going out even if it is a leaf (constraint (5.4.6)). The constraints (5.4.7) and (5.4.8) ensure that all vertices except the first one should have exactly one edge coming into it (one parent) and the first node does not have a parent. The spanning tree will have exactly $(n-1)$ edges due to constraint (5.4.9). The constraints (5.4.10)-(5.4.13) used to avoid cycles are formulated in the spirit of the classical Miller-Tucker-Zemlin (MTZ) formulation originally introduced for the traveling salesman problem in Miller et al. [1960]. The remaining are the non-zero and integrality constraints.

5.5 Computational Results

Randomly generated graphs were used to gauge the performance of the distributed algorithm empirically. The nodes were placed in 2-dimensional coordinates (x,y), where $x \in [0,100], y \in [0,100]$ were chosen to be uniform random variables. Five instances of graphs with 50 nodes (G1_50, G2_50, .. G5_50) and five instances with 100 nodes (G1_100, G2_100, .. G5_100) were generated. A maximum transmission range was specified that determined which nodes could communicate between themselves. For the examples in this chapter, the range was made high enough for full connectivity. The distributed algorithm was then run on the graphs generated to obtain a k-BCDS for different values of k. All experiments described below were performed on a machine with Intel Core 2 Duo 2.10 GHz CPU, and 2 GB of RAM.

Formulation (5.4.1)-(5.4.16) was solved using CPLEX 12.1 in order to compare the bottleneck value found by the distributed algorithm presented to the optimal. Although CPLEX was able to give optimal solutions for graphs with 50 nodes, solving instances with 100 vertices took prohibitive amount of time. For direct comparison, Table 5.1 pro-

$$\begin{aligned}
\text{min} \quad & z && && (5.4.1)\\
\text{subject to} \quad & \sum_{i\in V} x_i \le k && && (5.4.2)\\
& z \ge w_{ij}y_{ij} && i,j \in V && (5.4.3)\\
& \sum_{j\in V} y_{ij} \le nx_i && i \in V\setminus\{1\} && (5.4.4)\\
& \sum_{j\in V} y_{1j} \le 1+(n-1)x_1 && && (5.4.5)\\
& \sum_{j\in V} y_{1j} \ge 1 && && (5.4.6)\\
& \sum_{i\in V} y_{ij} = 1 && j \in V\setminus\{1\} && (5.4.7)\\
& y_{i1} = 0, && i \in V && (5.4.8)\\
& \sum_{i,j\in V, i\neq j} y_{ij} = n-1 && && (5.4.9)\\
& u_1 = 1 && && (5.4.10)\\
& u_i \le n && i \in V\setminus\{1\} && (5.4.11)\\
& u_i \ge 2 && i \in V\setminus\{1\} && (5.4.12)\\
& u_i - u_j + 1 \le (n-1)(1-y_{ij}), && i,j \in V\setminus\{1\} && (5.4.13)\\
& x_i, y_{ij} \in \{0,1\}, && i,j \in V, i\neq j && (5.4.14)\\
& u_i \in \Re && i \in V && (5.4.15)\\
& z \in \Re && && (5.4.16)
\end{aligned}$$

vides the optimal and approximate solution values in one of the randomly generated graph. The same data is visualized in Fig. 5.2. The approximation ratios for 4 more randomly generated graphs can be seen from Fig. 5.3. One can observe that the approximation ratio reduces and goes towards 1 as k increases in all the cases. Table 5.1 also provides the time taken by CPLEX and our algorithm to get to the solution provided.

Figure 5.2 plots the approximate and optimal solution values together. Observe that for large k the approximation algorithm provides optimal solutions. Also, the optimal d^*_{kBCDS} converges to the value of the optimal bottleneck spanning tree as $k \to n$. This is expected, since as k becomes sufficiently large, adding extra vertices simply results in more vertices coming into the CDS without the bottleneck edge being affected at all.

Computational studies of the algorithm on 5 randomly generated graphs with 100 nodes were also done, though CPLEX was not able to provide the optimal solution in a reasonable amount of time (1 hour). Table 5.2 provides the algorithm solution and CPLEX upper and lower bounds found for one of the randomly generated graphs. It should be noted that the CPLEX upper bound is the value of a feasible solution found, while the quality of the lower bound as compared to the optimal value cannot be judged based on the available information. Considering this, the ratio of the value of the solution found by the approximation algorithm to the lower bound would provide a theoretically safe approximation ratio, while a more practical performance measure should use the ratio to the upper bound. Ratios calculated using both of these bounds are presented in Fig. 5.4. It

Table 5.1: The bottleneck edges for the k-BCDS found on random graph G1_50 with varying k. The solutions found and the time taken to find by the distributed algorithm and CPLEX are presented. The approximation ratio is also provided. The approximation ratios for other random graphs are presented in Fig. 5.3.

Graph	k	Algo Sol	Algo	Optimal	CPLEX	Approx Ratio
		d_{kBCDS}	Time	d^*_{kBCDS}	Time	d_{kBCDS}/d^*_{kBCDS}
$G1\_50$	4	73.99	0.14s	43.96	227.5s	1.68
$G1\_50$	7	68.14	0.14s	32.29	277.1s	2.11
$G1\_50$	10	35.45	0.16s	25.83	587.1s	1.37
$G1\_50$	13	35.45	0.22s	23.53	3600.0s	1.51
$G1\_50$	16	34.10	0.25s	21.05	3600.0s	1.60
$G1\_50$	19	34.10	0.28s	21.05	652.6s	1.62
$G1\_50$	22	23.63	0.31s	21.05	1717.7s	1.12
$G1\_50$	25	23.63	0.39s	21.05	107.2s	1.12
$G1\_50$	28	23.69	0.39s	21.05	524.9s	1.13
$G1\_50$	31	21.05	0.44s	21.05	67.7s	1.00
$G1\_50$	50	21.05	0.66s	21.05	346.2s	1.00

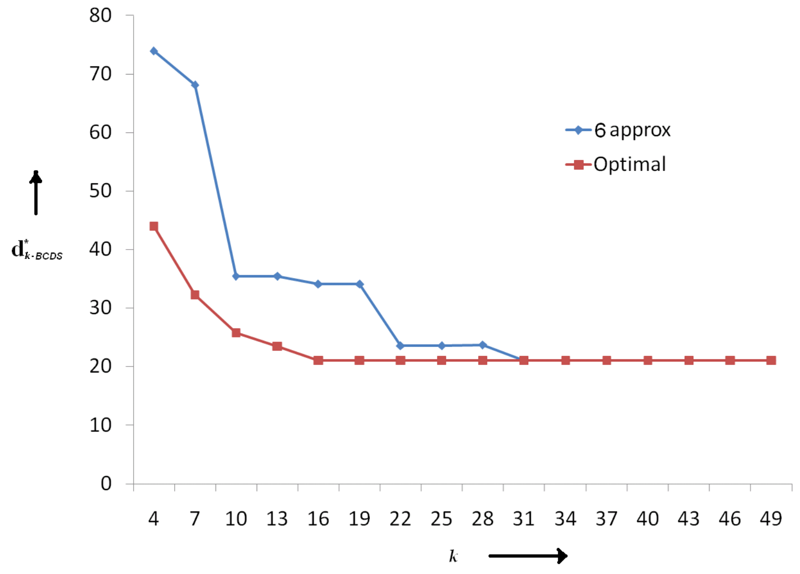


Fig. 5.2.: Comparison of the bottleneck edge value found by the distributed algorithm against the optimal found by CPLEX 12.1 for the *k*-BCDS with varying *k* on the random graph G1_50

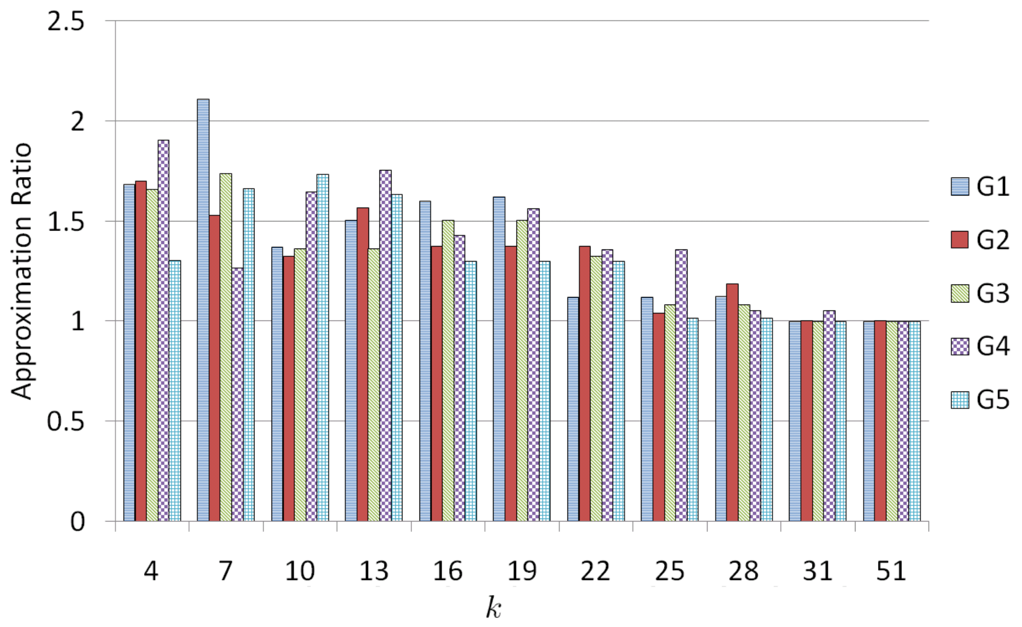
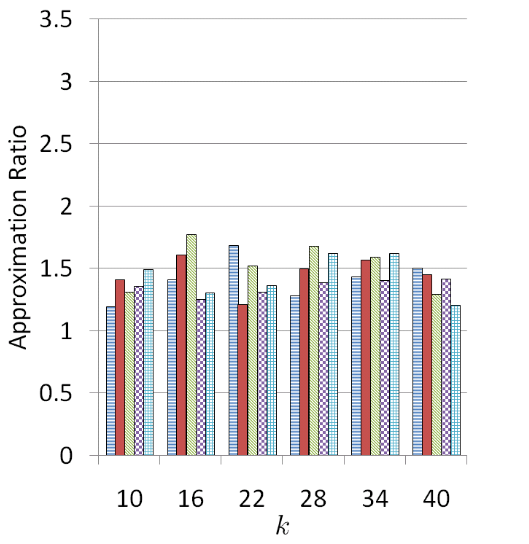


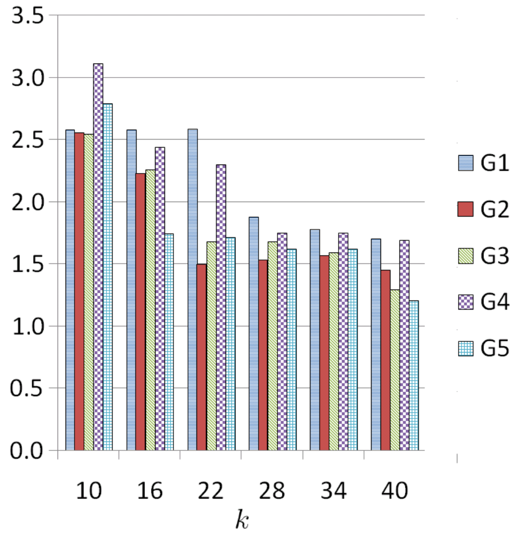
Fig. 5.3.: Approximation ratio of the solution found using distributed algorithm against the optimal found by CPLEX 12.1 for the *k*-BCDS with varying *k* on 5 random graphs G1_50,G2_50,..,G5_50 with 50 nodes

Table 5.2: The bottleneck edges for the k-BCDS found on random graph G1_100 having 100 nodes with varying k. The solutions found and the time taken to find by the distributed algorithm and CPLEX are presented. The approximation ratio is also provided. The approximation ratios for other random graphs are presented in Fig. 5.4a and 5.4b.

Graph	k	d_{kBCDS}	Algo Time	CPLEX LB	CPLEX UB	CPLEX Time	Approx Ratio(LB)	Approx Ratio(UB)
G1_100	10	38.23	0.94s	14.82	31.99	>3600s	2.58	1.19
G1_100	16	32.22	1.34s	12.50	22.79	>3600s	2.58	1.41
G1_100	22	33.70	1.75s	13.03	19.98	>3600s	2.59	1.69
G1_100	28	22.24	2.20s	11.83	17.35	>3600s	1.88	1.28
G1_100	34	22.24	2.59s	12.50	15.48	>3600s	1.78	1.44
G1_100	40	22.24	3.02s	13.03	14.73	>3600s	1.71	1.51



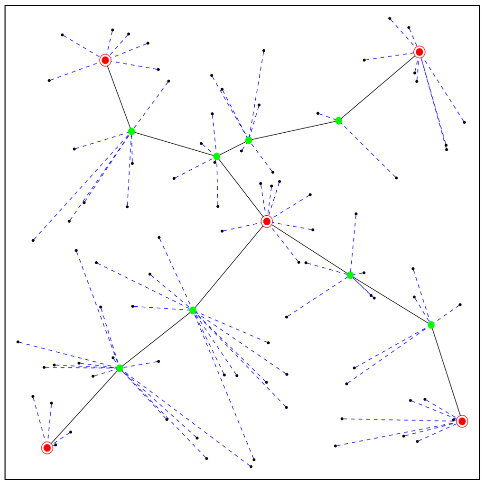
(a) Approximation ratio using CPLEX upper bound



(b) Approximation ratio using CPLEX lower bound

Fig. 5.4.: k-BCDS on 5 random graphs G1_100,G2_100,..,G5_100 with 100 nodes for varying k. The left chart shows the ratio of the algorithm solution to the upper bound found be CPLEX. The right chart shows the ratio of the algorithm solution to the lower bound found be CPLEX.

should be noted that none of the values exceed 6, even with the ratios using the CPLEX lower bounds, which could potentially be improved.


(a) k-BCDS with $k = 13$ on a network with 100 nodes

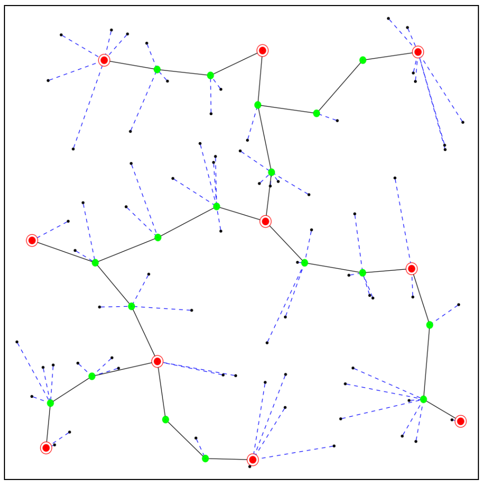

(b) k-CDS with $k = 28$ on a network with 100 nodes

Fig. 5.5.: k-BCDS on networks with 100 nodes for varying k. The double circle nodes were chosen by the algorithm to be in the $\lfloor k/3 \rfloor$-center in step 1. The nodes that connect them together, marked by a bold dot are added in step 2. The solid lines represent the connections in the virtual backbone. The dotted lines represent the communications from an external node to the backbone.

Figure 5.5 shows the CDS found on a graph with 100 nodes for different values of k. Notice that the nodes chosen as $\lfloor k/3 \rfloor$-centers are connected to another $\lfloor k/3 \rfloor$-center node by a path with 2 other nodes that are also included in the CDS. The pictorial representation of the graphs and the bottleneck CDS provides further insight into the algorithm and the structure of the CDS.

5.6 Conclusion

This chapter proposes a 6-approximate distributed algorithm for the k-BCDS problem on wireless sensor networks, which is motivated by the objective of increasing the lifetime of the network by reducing the maximum transmission range used by any of the nodes. Furthermore, computational results show that the approximation ratio is much less than 6 in practice. Some interesting observations worthy of further investigation came up from the computational results. For example, what is the minimum value of k at which the optimal *k*-BCDS value reaches its limit and stops decreasing? What is the minimum *k* for which the proposed algorithm finds the same bottleneck as the limiting bottleneck? A quantitative study on these parameters will provide useful insight into the problem. These questions might be important in deciding what values of *k* are appropriate to use in a sensor networks. Investigation into improved formulations for the *k*-BCDS problem might also be fruitful.

6. CONCLUSION

Given the immense potential graphs and networks have in providing a succinct representation of a system, it is no surprise that network science has become a popular area of research. The availability of a wide variety of analytical tools and algorithms in graph theory have further led to the application of these techniques across many different domains. While some applications can use previously developed theory and generalized algorithms rather easily – either a direct application or with slight modification – others require the development of dedicated tools to achieve meaningful results.

The underlying theme in all the topics studied in this dissertation is network models. The problems addressed in the first two topics, namely the maximum clique problem and the graph clustering problem, are very well known in literature and have been studied extensively. The approach presented in this dissertation takes a fresh look at the solution strategies for these problems, particularly focussing on large scale power law networks that arise in many applications. Power law or scale free networks provide a valuable avenue for future work especially because of their ubiquitous nature – many commonly studied networks such as social, internet, and market graphs show structural properties similar to power law graphs. Advances in information technology have resulted in the availability of graphs that are massive in size, and traditional techniques are no longer applicable at this larger scale. The void created by the lack of algorithms dedicated for such networks that provides avenues for future research. In particular, the knowledge of the special structure of these graphs can be used to develop algorithms that are particularly effective even on very large scale graphs.

Scale reduction, which was used in conjunction with *k*-community for solving the maximum clique problem, can take different forms and be used to solve other problems such as the maximum independent set and vertex coloring problems, amongst others. This requires studying the properties of these problems on such graphs in greater detail, concentrating not just an optimization approach but also on a much deeper graph theoretic analysis. Apart from scale reduction, studying structural properties of graphs with respect to these hard problems can also help in developing valid inequalities that can then be used in an integer programming framework.

On the other hand, clustering algorithms available today use a variety of techniques such as modularity optimization, spectral analysis and information flow models (dynamic processes). However, for many applications where the clusters need to exhibit certain structural characteristics, such generic clustering algorithms do not always suffice. The usefulness of the k-community in this regard was particularly apparent from the examples provided. However, k-communities suffer from some structural limitations similar to k-cores, such as linear rate of growth of the diameter in the size of the graph. These limitations are overcome to an extent by clique relaxations such as k-plexs, k-clubs, γ-quasi-cliques, etc., but their computational complexity makes them difficult to use on large scale graphs. For clustering applications, a future direction of research would be the development of new clique relaxations that are polynomially computable, yet exhibit cohesive properties that are much stronger than those of k-communities and k-cores.

The main focus of the third topic of location of emergency response facilities is in modeling the problem to a greater detail. With the advancements in computational resources, a much closer look at the problem and its modeling aspects in terms of detail is warranted. One future direction of study is in developing models that have more intricate requirements from the facilities, namely that of the service levels to be provided to the population of a city in case of a disaster. For example, one service level requirement might be that at least half the demand be met within 4 hours of the disaster, while the rest be met within 6 hours. This would require the modeling to be done at a higher level of detail, but would serve well for the decision makers who often have to keep such constraints in mind.

Although the solution techniques used for solving the 2-stage stochastic programming model were adapted from the well known Benders algorithms for our specific model, they could be of value in improving solution techniques for the classical p-center and p-median problems which have been solved by Benders algorithm in literature, especially since they are similar in structure to our problem. As illustrated by this hybrid approach, the usefulness of fast heuristics and approximation algorithms is not limited to problems where a polynomial computation time is required, but also aiding in the development of faster exact approaches for difficult problems.

Lastly, the fourth topic takes a fresh look at the problem of finding a virtual backbone for communication in wireless sensor networks by providing a distributed approximation algorithm that tries to minimize the distance any sensor has to transmit. The field of distributed algorithms, where decisions are not made by a central authority – as in conventional optimization – but by a collection of actors in the system, is fast growing and presents a viable area of research. Although our problem requires a distributed algorithm to minimize computational and communication overheads which are expensive for sensors that have limited resources, many other problems require a distributed approach because the actors in a system might have their own objectives in addition to the system-wide objective.

In wireless sensor networks, further study into distributed algorithms, in particular an analysis of the benefits of an improved performance against an increase in the communication/computational overhead that might occur is required. For examples, one approach could be to provide a distributed version of the algorithm provided by [Butenko et al., 2011], which would need edge weight information to be exchanged throughout the network, but has a much better approximation bound. It would be insightful to see which algorithm will actually performs better over the lifetime of the sensor network in practice. Furthermore, the present algorithms use a value of k that is pre-determined. However, unless the range of the sensor is very large and the corresponding network is completely connected, it is not always possible to find a connected dominating set of size k. A hybrid strategy of using the minimum connected dominating set to determine an appropriate value of k in conjunction with the bottleneck connected domination will be a more practical approach to the problem as k will be found dynamically rather than programmed prior to deployment. Another approach could be to employ a binary search scheme to find the appropriate size of the connected dominating set. A comparative analysis of these different approaches with the algorithm provided in this paper would be insightful in determining the value of k to be used for practical purposes.

Network science, network modeling and algorithms – like many other fields – will continue to benefit greatly from the improved computational infrastructure and availability of huge data sets due to the outreach of the internet and the development of tools for data sharing and creation. For example, tools such as Geographic Information Systems (GIS) make access to comprehensive demographic information extremely easy for researchers. Parallel and distributed computing provide opportunities for improving algorithms to utilize the full potential of these systems. Collectively, such advances provide an opportunity to collect relevant data at a much finer level of detail and to analyze it in a much more sophisticated manner, something that was not possible a decade ago.

REFERENCES

J. Abello, P.M. Pardalos, and M.G.C. Resende. On maximum clique problems in very large graphs. In J. Abello and J. Vitter, editors, *External memory algorithms and visualization*, volume 50 of *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, pages 119–130. American Mathematical Society, 1999.

J. Abello, M.G.C. Resende, and S. Sudarsky. Massive quasi-clique detection. In S. Rajsbaum, editor, *LATIN 2002: Theoretical Informatics*, pages 598–612, London, 2002. Springer-Verlag.

K. Akkaya and M. Younis. A survey on routing protocols for wireless sensor networks. *Ad Hoc Networks*, 3(3):325–349, 2005.

R.D. Alba. A graph-theoretic definition of a sociometric clique. *Journal of Mathematical Sociology*, 3:113–126, 1973.

M. Altaf-Ul-Amin, K. Nishikata, T. Koma, T. Miyasato, Y. Shinbo, M. Arifuzzaman, C. Wada, and M. Maeda et al. Prediction of protein functions based on *k*-cores of protein-protein interaction networks and amino acid sequences. *Genome Informatics*, 14:498––499, 2003.

J.I. Alvarez-Hamelin, L. Dall'Asta, A. Barrat, and A. Vespignani. *k*-core decomposition: a tool for the visualization of large scale networks. *Computing Research Repository*, abs/cs/0504107, 2005.

H. Attiya and J. Welch. *Distributed Computing: Fundamentals, Simulations and Advanced Topics*. John Wiley & Sons, Hoboken, NJ, 2004.

G. D. Bader and C. W. V. Hogue. An automated method for finding molecular complexes in large protein interaction networks. *BMC Bioinformatics*, 4(2), 2003.

E. Balas and J. Xue. Weighted and unweighted maximum clique algorithms with upper bounds from fractional coloring. *Algorithmica*, 15:397–412, 1996. 10.1007/BF01955041.

B. Balasundaram, S. Butenko, and I.V. Hicks. Clique relaxations in social network analysis: The maximum k-plex problem. *Operations Research*, 59:133–142, 2011.

B. Balcik and B. M. Beamon. Facility location in humanitarian relief. *International Journal of Logistics: Research and Applications*, 11(2):101–121, 2008.

G. Barbarosoglu and Y. Arda. A two-stage stochastic programming framework for transportation planning in disaster response. *The Journal of the Operational Research Society*, 55(1):43–53, 2004.

J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4:238–252, 1962. 10.1007/BF01386316.

P. Beraldi, M.E. Bruni, and D. Conforti. Designing robust emergency medical service via stochastic programming. *European Journal of Operational Research*, 158:183–193, 2004.

O. Berman, Z. Drezner, and G.O. Wesolowsky. Locating service facilities whose reliability is distance dependent. *Computers & Operations Research*, 30(11):1683 – 1695, 2003.

O. Berman, D. Krass, and M.B.C. Menezes. Facility reliability issues in network p-median problems: Strategic centralization and co-location effects. *Operations Research*, 55(2):332–350, 2007.

Jr. B.F. Howell and T.R. Schultz. Attenuation of modified mercalli intensity with distance from the epicenter. *Bulletin of the Seismological Society of America*, 65:651 – 665, 1975.

G. Bianconi and M. Marsili. Emergence of large cliques in random scale-free networks. *Europhysics Letters*, 74:740–746, May 2006.

S. Biaz and Y. Ji. A survey and comparison on localisation algorithms for wireless ad hoc networks. *International Journal of Mobile Communications*, 3(4):374–410, 2005.

J. Blum, M. Ding, A. Thaeler, and X. Cheng. Connected dominating set in sensor networks and MANETs. In D.-Z. Du and P.M. Pardalos, editors, *Handbook of Combinatorial Optimization*, pages 329–369. Springer Science+Business Media, Inc., New York, NY, 2005.

V. Boginski, S. Butenko, and P.M. Pardalos. Statistical analysis of financial networks. *Computational Statistics & Data Analysis*, 48(2):431 – 443, 2005.

S. Butenko and S. Trukhanov. Using critical sets to solve the maximum independent set problem. *Operations Research Letters*, 35(4):519 – 524, 2007.

S. Butenko and W.E. Wilhelm. Clique-detection models in computational biochemistry and genomics. *European Journal of Operational Research*, 173(1):1 – 17, 2006.

S. Butenko, S. Kahruman-Anderoglu, and O. Ursulenko. On connected domination in unit ball graphs. *Optimization Letters*, 5:195–205, 2011.

R. Carraghan and P.M. Pardalos. An exact algorithm for the maximum clique problem. *Operations Research Letters*, 9(6):375 – 382, 1990.

L. Cavique. A scalable algorithm for the market basket analysis. *Journal of Retailing and Consumer Services*, 14(6):400 – 407, 2007.

CDC. Centers for disease control and prevention. http://www.bt.cdc.gov/stockpile/, Last accessed: May 2011.

F. Corno, P. Prinetto, and M. Sonza Reorda. Using symbolic techniques to find the maximum clique in very large sparse graphs. In *Proceedings of the 1995 European conference on Design and Test*, EDTC '95, pages 320–, Washington, DC, USA, 1995. IEEE Computer Society.

G. Cote and M.A. Laughton. Large-scale mixed integer programming: Benders-type heuristics. *European Journal of Operational Research*, 16(3):327 – 333, 1984.

B. Das and V. Bharghavan. Routing in ad-hoc networks using minimum connected dominating sets. In *IEEE International Conference on Communications, 'Towards the Knowledge Millennium'*, volume 1, pages 376–380, Montreal, 1997.

DIMACS10. 10th DIMACS Implementation Challenge: Graph Partitioning and Graph Clustering. http://www.cc.gatech.edu/dimacs10/, Last accessed: May 2012.

DIMACS2. DIMACS Clique Benchmark Instances. ftp://dimacs.rutgers.edu/pub/challenge/, Last accessed: November 2011.

S. Duran, M.A. Gutierrez, and P. Keskinocak. Pre-positioning of emergency items for care international. *Interfaces*, 41:223–237, May 2011.

E. J. Gardiner, P. J. Artymiuk, and P. Willett. Clique-detection algorithms for matching tree-dimensional molecular structures. *Journal of Molecular Graphics and Modeling*, 15:245–253, 1998.

M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W.H. Freeman and Company, New York, NY, 1979a.

M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York, NY, 1979b.

M. Gendreau, P. Soriano, and L. Salvail. Solving the maximum clique problem using a tabu search approach. *Annals of Operations Research*, 41:385–403, May 1993.

L. Girod and D. Estrin. Robust range estimation using acoustic and multimodal sensing. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2001)*, Maui, HI, October 2001.

T.F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985.

S. Guha and S. Khuller. Approximation algorithms for connected dominating sets. *Algorithmica*, 20:374–387, 1996.

K. Holmberg. On using approximations of the benders master problem. *European Journal of Operational Research*, 77(1):111 – 125, 1994.

J. Idicula. Highly interconnected subsystems of the stock market. NET Institute Working Paper No. 04-17, 2004. Available at SSRN: http://ssrn.com/abstract=634681.

H. Jia, F. Ordonez, and M. Dessouky. A modeling framework for facility location of medical services for large-scale emergencies. *IIE Transactions*, 39(1):41–55, 2007.

L. Jia, R. Rajaraman, and T. Suel. An efficient distributed algorithm for constructing small dominating sets. *Distributed Computing*, 15:193–205, 2002.

K. Katayama, A. Hamamoto, and H. Narihisa. An effective local search for the maximum clique problem. *Information Processing Letters*, 95(5):503 – 511, 2005.

A. Künzi-Bay and J. Mayer. Computational aspects of minimizing conditional value-at-risk. *Computational Management Science*, 3:3–27, 2006.

X.-Y. Li and I. Stojmenovic. Broadcasting and topology control in wireless ad hoc networks. *Handbook of Algorithms for Mobile and Wireless Networking and Computing*, pages 239–261, 2005.

N. Linial. Locality in distributed graph algorithms. *SIAM Journal on Computing*, 21(1): 193–201, 1992.

T. Matsunaga, C. Yonemori, E. Tomita, and M. Muramatsu. Clique-based data mining for related genes in a biomedical database. *BMC Bioinformatics*, 10(1):205, 2009.

H.O. Mete and Z.B. Zabinsky. Stochastic optimization of medical supply location and distribution in disaster management. *International Journal of Production Economics*, 126(1):76 – 84, 2010.

C.E. Miller, A.W. Tucker, and R.A. Zemlin. Integer programming formulations and traveling salesman problems. *Journal of the ACM*, 7:326–329, 1960.

M. Min, H. Du, X. Jia, C. X. Huang, S. C.-H. Huang, and W. Wu. Improving construction for connected dominating set with Steiner tree in wireless sensor networks. *Journal of Global Optimization*, 35:111–119, 2006.

N. Modani, K. Dey, S. Mukherjea, and A.A. Nanavati. Discovery and analysis of tightly knit communities in telecom social networks. *IBM Journal of Research and Development*, 54(6):7:1 –7:13, nov.-dec. 2010.

P. Murali, F. Ordonez, and M.M. Dessouky. Capacitated facility location with distance-dependent coverage under demand uncertainty. *Non-published Research Reports*, Paper 130, 2009. URL `http://research.create.usc.edu/nonpublished_reports/130`.

L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999.

J.A. Paul and R. Batta. Models for hospital location and capacity allocation for an area prone to natural disasters. *International Journal of Operational Research*, 3(5):473–496, 2008.

C.A. Poojari and J.E. Beasley. Improving benders decomposition using a genetic algorithm. *European Journal of Operational Research*, 199(1):89 – 97, 2009.

B. Protasi, R. Battiti, and M. Protasi. Reactive local search for the maximum clique problem. Technical report, Algorithmica, 1995.

C.G. Rawls and M.A. Turnquist. Pre-positioning of emergency supplies for disaster response. *Transportation Research Part B: Methodological*, 44(4):521 – 534, 2010.

J.W. Raymond and P. Willett. Maximum common subgraph isomorphism algorithms for the matching of chemical structures. *Journal of Computer-Aided Molecular Design*, 16: 521–533, 2002.

R.T. Rockafellar and S. Uryasev. Optimization of conditional value-at-risk. *The Journal of Risk*, 2(3):21–41, 2000.

J.B. Rundle, D.L. Turcotte, R. Shcherbakov, W. Klein, and C. Sammis. Statistical physics approach to understanding the multiscale dynamics of earthquake fault systems. *Reviews of Geophysics*, 41, Feb 2003.

A. Savvides, C.-C. Han, and M.B. Srivastava. Dynamic fine-grained localization in ad-hoc networks of sensors. In *MobiCom '01: Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, pages 166–179, New York, NY, 2001. ACM.

C.H. Schultz, K.L. Koenig, and R.J. Lewis. Decisionmaking in hospital earthquake evacuation: Does distance from the epicenter matter? *Annals of Emergency Medicine*, 50(3): 320–326, 2007.

S.B. Seidman. Network structure and minimum degree. *Social Networks*, 5:269–287, 1983.

S.B. Seidman and B.L. Foster. A graph theoretic generalization of the clique concept. *Journal of Mathematical Sociology*, 6:139–154, 1978.

Y. Shen, D.T. Nguyen, Y. Xuan, and M.T. Thai. New techniques for approximating optimal substructure problems in power-law graphs. *Theoretical Computer Science*, In Press, 2011.

I. Shin, Y. Shen, and M.T. Thai. On approximation of dominating tree in wireless sensor networks. *Optimization Letters*, 4:393–403, 2010.

SNAP. Stanford Large Network Dataset Collection. http://snap.stanford.edu/data/, Last accessed: November 2011.

L.V. Snyder and M.S. Daskin. Reliability models for facility location: The expected failure cost case. *Transportation Science*, 39(3):400–416, 2005.

D.M. Strickland, E. Barnes, and J.S. Sokol. Optimal protein structure alignment using maximum cliques. *Operations Research*, 53:389–402, May 2005.

M.T. Thai, F. Wang, D. Liu, S. Zhu, and D.-Z. Du. Connected dominating sets in wireless networks with different transmission ranges. *IEEE Transactions on Mobile Computing*, 6:721–730, 2007.

E. Tomita and T. Seki. An efficient branch-and-bound algorithm for finding a maximum clique. In C. Calude, M. Dinneen, and V. Vajnovszki, editors, *Discrete Mathematics and Theoretical Computer Science*, volume 2731 of *Lecture Notes in Computer Science*, pages 278–289. Springer Berlin / Heidelberg, 2003. 10.1007/3-540-45066-1-22.

USGS. U.S. Geological Survey. http://earthquake.usgs.gov/earthquakes/, Last accessed: May 2011.

A. Verma and G.M. Gaukler. A stochastic optimization model for positioning disaster response facilities for large scale emergencies. In Julia Pahl, Torsten Reiners, and Stefan Voß, editors, *Network Optimization*, volume 6701 of *Lecture Notes in Computer Science*, pages 547–552. Springer Berlin / Heidelberg, 2011. 10.1007/978-3-642-21527-8-60.

D.R. Wood. An algorithm for finding a maximum clique in a graph. *Operations Research Letters*, 21(5):211 – 217, 1997.

J. Wu and H. Li. On calculating connected dominating set for efficient routing in ad hoc wireless networks. In *DIALM '99: Proceedings of the 3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pages 7–14, New York, NY, 1999. ACM.

J. Wu and B. Wu. A transmission range reduction scheme for power-aware broadcasting in ad hoc networks using connected dominating sets. *IEEE-Vehicular Technology Conference*, 5:2906–2909, 2003.

W. Wu, X. Gao, P. M. Pardalos, and D.-Z. Du. Wireless networking, dominating and packing. *Optimization Letters*, 4:347–358, 2010.