SIMULATING EPIDEMICS IN RURAL AREAS AND OPTIMIZING PREPLANNED QUARANTINE AREAS USING A CLUSTERING HEURISTIC

by

JOSEPH EDWARD ANDERSON

B.S., United Stated Military Academy, 1999

A THESIS

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Industrial and Manufacturing Systems Engineering
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2009

Approved by:

Major Professor
Todd Easton, PhD

Abstract

With the present threat of bioterrorist attacks and new natural disease strains developing, efficient and effective countermeasures must be in place in case of an epidemic outbreak. The best strategy is to stop the attack or natural phenomenon before it happens, but governments and individual citizens must have measures in place to limit the spread of a biological threat or infectious disease if it is ever introduced into society.

The objective of this research is to know, before an outbreak, the best quarantine areas. Quarantines force similar individuals together and can be mathematically modeled as clustering people into distinct groups.

In order to effectively determine the clustering solution to use as a quarantine plan, this research developed a simulation core that is highly adaptable to different disease types and different contact networks. The input needed for the simulation core is the characteristics of the disease as well as the contact network of the area to be modeled.

Clustering is a mathematical problem that groups entities based on their similarities while keeping dissimilar entities in separate groups. Clustering has been widely used by civilian and military researchers to provide quality solutions to numerous problems. This research builds a mathematical model to find clusters from a community's contact network. These clusters are then the preplanned quarantine areas.

To find quality clusters a Clustering Heuristic using Integer Programming (CHIP) is developed. CHIP is a large neighborhood, hill-climbing heuristic and some computational results verify that it quickly generates good clustering solutions. CHIP is an effective heuristic to group people into clusters to be used as quarantine areas prior to the development of a disease or biological attack. Through a small computational study, CHIP is shown to produce clustering solutions that are about 25% better than the commonly used *K*-means clustering heuristic.

CHIP provides an effective tool to combat the spread of an infectious disease or a biological terroristic attack and serves as a potential deterrent to possible terrorist attacks due to

the fact that it would limit their destructive power. CHIP leads to the next level of preparation that could save countless lives in the event of an epidemic.

Table of Contents

List of Figures

List of Tables

Acknowledgements

There are a couple of people I would like to acknowledge. My wife, Sandy Anderson, was a huge help in letting me talk to her and ask her questions about this thesis. Also, Kyle Carlyle was another excellent listener and person to bounce ideas off of. Lastly, I would like to thank the faculty of my committee: Dr. Caterina Scoglio, Dr. Todd Easton, and Dr. John Wu.

Dedication

This work is dedicated to my wife, Sandy, and our family.

CHAPTER 1 - Introduction

"The greatest threat before humanity today is the possibility of a secret and sudden attack with chemical, or biological, or nuclear weapons."

President George W. Bush,

National Defense University Address, February 2004

With the present threat of bioterrorist attacks and new natural disease strains developing, efficient and effective countermeasures must be in place in case of an epidemic outbreak. The best strategy is to stop the attack or natural phenomenon before it happens, but governments and individual citizens must have measures in place to limit the spread of a biological threat or infectious disease if it is ever introduced into society.

To date, the most effective ways to stop the spread of an epidemic has been quarantining infectious individuals or developing vaccines. Vaccines take years to produce and administer and are unlikely to be effective in the case of a bioterrorist attack or the natural introduction of a new virus.

The objective of this research is to know, before an outbreak, the best quarantine areas. For the purposes of this paper, quarantine is defined as a strict isolation to prevent the spread of disease. Quarantines force similar individuals together and can be mathematically modeled as clustering people into distinct groups.

Clustering is a mathematical problem that groups entities based on their similarities while keeping dissimilar entities in separate groups. Clustering has been widely used by civilian and military researchers to provide quality solutions to numerous problems.

One prominent clustering example in military applications is in wireless sensor networks. Wireless sensor networks (WSNs) have been used in a number of different applications such as combat field reconnaissance, border protection, and security surveillance. WSNs are defined as a "large scale ad hoc, multi-hop, unpartitioned network of largely homogeneous, tiny, resource

constrained, mostly immobile sensor nodes that would be deployed in the area of interest" [1].[1] The military deploys WSNs in remote areas to sense the movement of people or vehicles by seismic vibrations.

Recently there have been a number of clustering algorithms specially designed for WSNs [2, 3, 4, 5]. Clustering WSNs offers many advantages. Clustering establishes a cluster-head (CH) that acts as a leader for that grouping and has more resources than the rest of the sensors. Clustering can conserve bandwidth since only the CH conducts inter-cluster communications. Clustering can stabilize network topology at the sensor level and thus reduce topology maintenance costs. Sensors would only need to communicate with the CH and in turn the CH would reduce redundancy in transmissions. The CH can act to save power by alternating which sensors are active and the number of messages they are transmitting. Lastly, the CH can also ensure that there is not an overlap in coverage areas by disabling some sensors or engaging the sensors at different times around the cluster [6].

There are numerous different applications that use clustering to find better solutions and these numerous applications best describe the importance of the concept. Several such applications include: data mining, where data is sorted into useable and logical groups [7, 8]; social networks, where users are grouped based on their likes, dislikes, or interests [9, 10]; and market research, where consumers are grouped based on their possible interests, recommendations, and the products that they have purchased [11, 12].

From a computational stand point, finding optimal clustering solutions is one of the most computationally challenging problems, and no fast algorithms exist to generate the optimal clustering solution, unless $\mathcal{P} = \mathcal{NP}$. Thus, the goal of clustering heuristics is to find good clustering solutions in a reasonable amount of time.

1.1 Research Motivation

The motivating factor for this research is to stop the spread of an infectious disease or bioterrorist attack by utilizing clusters to develop quarantine areas prior to an outbreak. The

[1] The numbers in brackets correspond to those of the bibliography.

clusters represent the groups formed by quarantine areas to stop the disease from destroying an entire population. These quarantine areas are based upon the movement data and interactions between people.

In the mid-1300s, the Black Death spread throughout Europe and parts of Central Asia and many believe that the Black Death was actually the bubonic plague [13]. Although science and pharmaceuticals have advanced enough to stop the spread of this pandemic, the bacteria thought to have caused the Black Death still exist today. In fact, fleas infected with the bubonic plague were used as a bacteriological weapon as recently as the 1940s [14].

It is reasonable to believe that, just as medicine has advanced, so to has the ability to manufacture new and improved bacteria and viruses that are resistant to advanced medicine. The concept of quarantines began soon after the Black Death pandemic. Today, the Centers for Disease Control and Prevention (CDC) have twenty Quarantine Stations covering the United States and some of its territories. The CDC has many measures in place to stop an infectious disease from entering the country; however, if the disease does not show itself until it is in the heartland of America, what measures are in place to stop the disease from spreading to the rest of the nation?

The motivation behind this research is the special nature of rural Americans traveling great distances across land on a regular basis. Due to the fact that these Americans come into contact with many different people who in turn travel long distances and have additional contacts, a disease could spread across a very large rural space in a relatively short amount of time. In this manner, knowing where to quarantine an area and being able to pre-position supplies and resources before an outbreak is extremely beneficial.

1.2 Research Contributions

The research began by constructing a simulation model to show the spread of an infectious disease and how it would move through a small community approximately the size of Clay Center, Kansas (approximate population: 4,600). The simulation core is the result of combined efforts by three other students, Kyle Carlyle, Matthew James, David Willis, and myself along with our primary instructor, Dr. Todd Easton. Each interaction of an infected entity with an uninfected person posed a risk of spreading the disease. This is modeled by assigning a

probability that an infectious individual contaminates a particular susceptible person. The probability is asymmetric, varies between individuals, and is modeled as a contact network. For the purposes of this paper, a contact network is a mathematical representation of peoples' interactions. Using the contact network of Clay Center, in less than 30 seconds, the model simulated the spread of an infection over 10 days, achieving a computationally tractable result. The simulation shows that whole rural towns must be quarantined instead quarantining certain sections.

Based on these simulated contact networks, this research develops an integer program that solves for the optimal clustering solution. This clustering integer program can only be solved on very small contact networks. A contact network with over 30 entities ran for several days without finding a provably optimal clustering solution.

To overcome these computational inadequacies, this research develops a heuristic to achieve quality clustering solutions in a reasonable amount of time. The Clustering Heuristic using Integer Programming (CHIP) is a large neighborhood heuristic and some computational results verify that it quickly generates good clustering solutions. CHIP develops an effective heuristic to group people into quarantine areas prior to the development of a disease or biological attack. Through a small computational study, CHIP is shown to produce clustering solutions that are about 25% better than the commonly used *K*-means clustering heuristic.

CHIP provides an effective tool to combat the spread of an infectious disease or a biological terroristic attack and serves as a potential deterrent to possible terrorist attacks due to the fact that it would limit their destructive power. CHIP leads to the next level of preparation that could save many lives. Governments should use this heuristic to develop preparation plans to save many lives that would otherwise be lost.

1.3 Thesis Outline

The second chapter presents background information and the general ideas of graph theory as well as clustering, heuristics, and applications to infectious diseases. The chapter gives many of the basic concepts of graph theory and contact networks and is essential to understanding the rest of the work.

Chapter 3 introduces and discusses the epidemic simulation. Specifically, this chapter shows the establishment of a contact graph used for the simulation and how the probabilities are calculated. Additionally, this chapter presents some computational results.

The fourth chapter introduces the research and the resulting heuristic. This chapter contains a progression from the clustering integer program to the clustering heuristic that is the central idea of this thesis. Some computational results demonstrate that this clustering heuristic finds excellent solutions.

Chapter 5 summarizes the quantifiable results from this research. The fifth chapter also recommends future research that could continue from this work. Some extensions on how governments can apply this research is also discussed.

CHAPTER 2 - Background Information

A basic level of understanding is needed before going into greater detail into this research. In this section, many of the basic concepts about graph theory are explained. First, it is necessary to understand the applications and fundamentals of graph theory. Next this chapter discusses some of the common problems associated with graphs. Then the chapter explains some of the concepts of clustering, specifically some of the distance measures and related concepts. At this point it is also necessary to discuss heuristics and how they can be applied to this research. This chapter finishes with a brief discussion of graph theory's relation to epidemic modeling and infectious diseases.

2.1 Graph Theory

Graph theory has been used extensively to model and solve complex problems. Graph theory provides a structured framework to represent objects and their relationships to other objects. The idea is to take something that is extremely complex in the real world and model it as a graph and then to develop methodologies to solve the problem on the graph. The solution is then translated into real-world policies.

This work only covers a few of the basics concerning graph theory and then discusses some of the more complex models. A more complete description of graph theory can be found in Diestel [15] and Ahuja [16].

2.1.1 Applications of Graph Theory

This section highlights a few of the thousands of graph theory applications that include applications in the military, communications, network analysis, and job assignments. The military has extensively used graph theory in a number of different applications. Many in the military have used graph theory to model communications networks, both to optimize connectivity [17] as well as to maximize the disruption of enemy communications [18]. Graph theory is also applied to military operations using resource graphs and process graphs to improve efficiency in scheduling [19]. Some have used graph theory to assist the military decision

makers by solving the shortest path problem and a game to theoretically model and test different strategies [20]. These are just a few of the applications of graph theory on military planning and operations. With graph theory's many uses to the military, it obviously offers much to the rest of civilization as well.

One group from the University of Arizona used network analysis to look at Al Qaeda networks and help the Department of Homeland Security focus on terrorist networks [21]. Using data mining techniques, this method can connect collected data from several different federal agencies and form one cohesive piece of information. Graph theory has also been used in network analysis for knowledge mapping in organizations [22]. The idea is to use a knowledge map, or graph, to better link ideas and techniques around an organization.

Graph theory is also a very common tool to use when developing, designing, and incorporating communications networks. Kilic [23] uses graph theory and methods to model cyclic sequential circuits. Graph theory was also used to develop a nearest neighbor embracing graph for communications networks [24]. These are just two examples where graphs were used as models for communications networks.

Task scheduling and determining job assignments are also useful applications of graph theory. Harvey [25] writes about a method for determining how to fairly match people to jobs using an optimal semi-matching method. Using this method, Harvey presents two new algorithms centered on a particular graph type that is extremely useful for job assignments. Karaata [26] shows a method for finding the maximal matching application in stable marriages and presents a new algorithm that is useful in matching the best two pairs in a graph.

Graph theory can also be applied to the analysis of road networks and their efficiency. Some look at the complex job of maintaining accurate road network data that is always changing by incorporating graph theory techniques [27]. Chen examines the relationship of a road network to graph theory and utilizes some special techniques to improve road network construction. Others look at the relationship of a two-way road network as one type of graph and a one-way road network as another type [28]. Lew uses the relationship between these two graphs to improve a previously developed algorithm.

These examples show some of the many different ways that graph theory can be implemented to generate solutions to complex problems. It is not surprising that graph theory

has also found its way into modeling the spread of infectious diseases. This is the fundamental topic of this research and applications in this area are reserved for later in this thesis. First some fundamental concepts of graph theory are explained.

2.1.2 Fundamentals of Graph Theory

Formally, an undirected graph $G = (V, E)$ is defined as a set of sets such that $E \subseteq [V]^2$; thus, the elements of E are 2-element subsets of V. The vertices are typically denoted by circles and represent elements in V. Lines connecting two vertices denote the edges in the graph. An undirected graph is depicted in Figure 2.1 with $V = \{m, n, o, p, q, r, s, t, u, v, w, x, y, z\}$ and $E = \{\{m, n\}, \{m, o\}, \{o, s\}, \{n, p\}, \{n, t\}, \{p, r\}, \{r, s\}, \{r, t\}, \{r, v\}, \{t, u\}, \{t, w\}, \{u, w\}, \{w, y\}, \{v, x\}, \{v, z\}, \{v, y\}, \{x, z\}, \{x, y\}, \{y, z\}\}$.

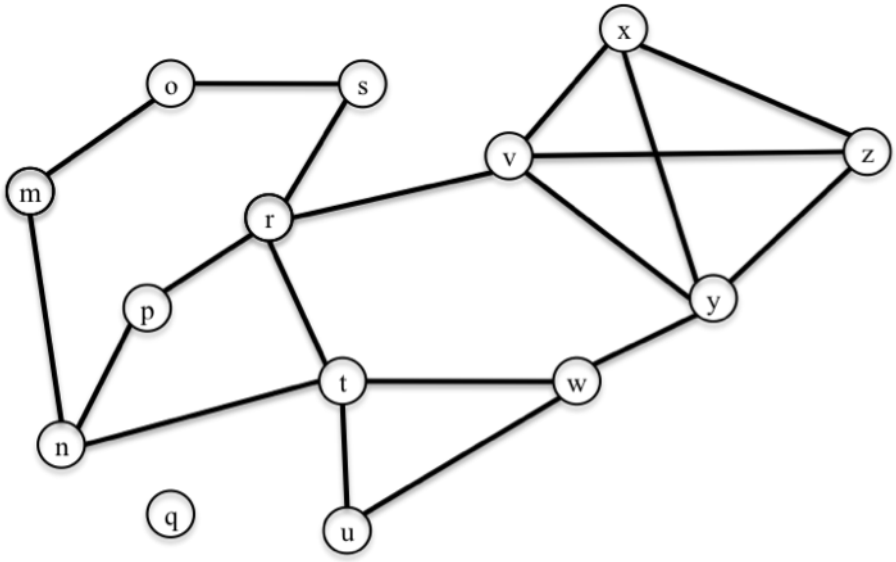

Figure 2.1 An undirected graph

Graph theory has numerous important definitions that are critical to this work. Two nodes u and v are said to be adjacent if $\{u,v\} \in E$ and the node u is said to be incident to edge $\{u,v\}$. The degree of a node is the number of incident edges. As an example, in Figure 2.1 vertex r is adjacent to vertices p, t, s and v and the obvious edges are incident. Therefore, the degree of vertex r is four, $\deg(r) = 4$.

A subgraph $H = (V', E')$ is a subset of G where V' is a subset of V and E' is a subset of E. A subgraph of Figure 2.1 can be seen in the left of Figure 2.2. An induced subgraph is given by

a set of nodes in a graph and includes all edges that are incident to any two nodes in the set. The right graph in Figure 2.2 is an example of the induced subgraph of $\{r, t, v, w, x, y, z\}$.

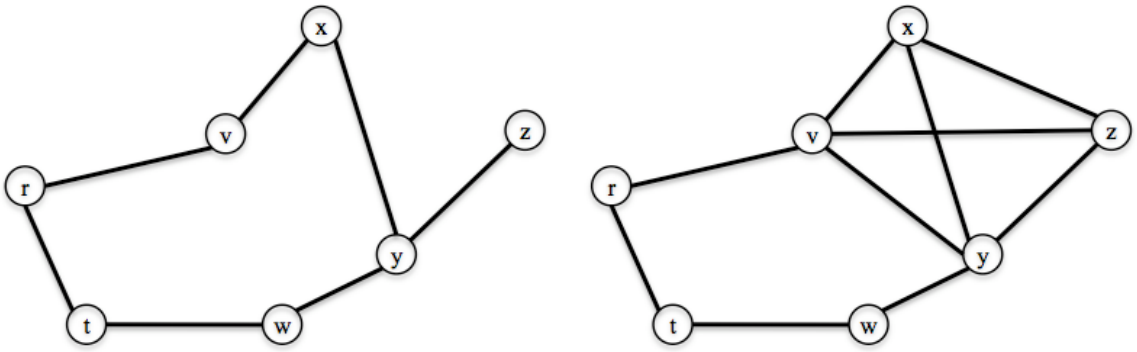

Figure 2.2 Subgraph and induced subgraph

An s to t path in a graph is a non-empty graph where nodes s and t that are linked by a particular set of nodes and edges. Figure 2.3 shows a path from s to t indicated in bold along edges ($\{s, r\}$, $\{r, p\}$, $\{p, n\}$, $\{n, t\}$).

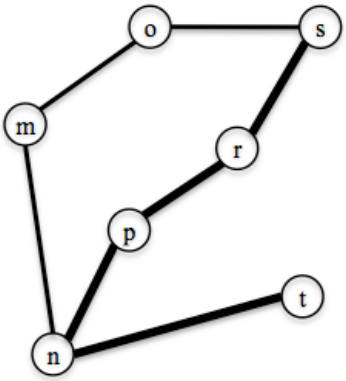

Figure 2.3 Path from s to t

A cycle is easily defined as a path that starts and ends at the same node. Clearly a cycle has no official beginning or end. An example of a cycle in the graph from Figure 2.1 is (v, y, w, u, t, n, p, r) and is shown in Figure 2.4.

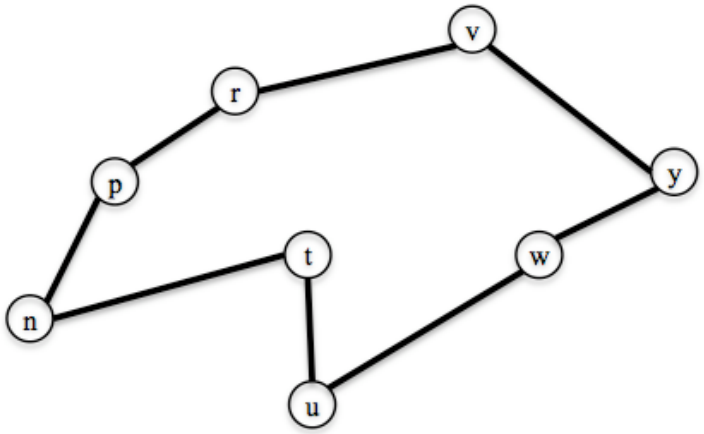

Figure 2.4 Example of a cycle

A complete graph is a graph where all vertices are adjacent. Induced subgraphs that are complete are commonly referred to as cliques. Cliques play an important role in this research because if edges represent similarities, then a clique is a perfect cluster. A clique with n vertices is denoted as K_n and has $n(n\text{-}1)/2$ edges. Two examples of cliques are given in Figure 2.5.

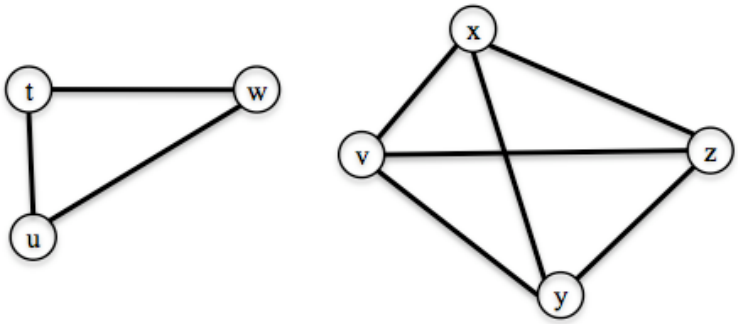

Figure 2.5 Examples of a K_3 and a K_4

Much of the applications of graph theory involve directed networks. A directed graph $D = (V, A)$ has a vertex set V and an arc set A. The arc set consists of ordered pairs of vertices from V. The order of these vertices denotes the direction along that arc. Figure 2.6 shows a directed graph with $V = \{m, n, o, p, r, s, t\}$ and $A = \{(m,o), (o,s), (r,s), (s,r), (r,t), (r,p), (p,n), (n,t), (n,m)\}$.

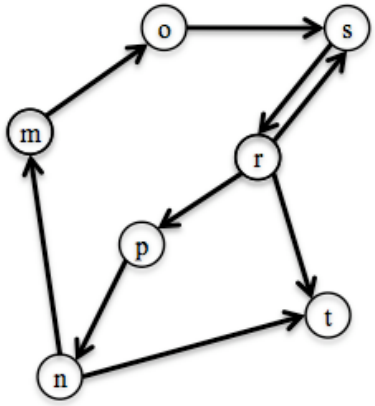

Figure 2.6 Directed graph

Networks are apparent in everyday life. A network flow problem is a directed graph where each node and edge has a set of numbers assigned to it. Some frequent assignments of numbers represent the capacity, distance, cost, profit, demand, supply, etc. Due to the ability of networks to accurately model real world problems, many standard network problems exist and some of these problems along with methods used to solve them are the subject of the next section.

2.2 Problems on Graphs

Consider the following simple network flow problem that involves material shipment from multiple warehouses to multiple locations. Assume that there are many different items at three different warehouses located in Los Angeles, Seattle, and Denver. These items need to be moved from the warehouses to the manufacturing plants in San Francisco, Sacramento, and Salt Lake City. Each warehouse has only a portion of what the manufacturing plants need to operate. Therefore, items from each warehouse must be delivered to each manufacturing plant. This problem could be modeled using network flow to determine the minimum cost of transporting the items from warehouse to manufacturing plant. The numerous papers written on this subject show the importance of this topic. A small subset of these papers include [29, 30, 31, 32, 33, 34].

Besides these problems, other graph problems have been used to improve chip design [35, 36]; vehicle routing [37, 38]; decreasing road traffic noise [39, 40]; improving supply

logistics and management [41, 42]; and increasing connectivity in wireless telecommunications [43, 44].

Problems on graphs and networks are largely divided into two classes, $\mathcal{P}$ and $\mathcal{NP}$-hard. Some problems are still open. Problems in $\mathcal{P}$ have polynomial time algorithms that can solve the problem. Problems that are $\mathcal{NP}$-hard have instances that require an exponential amount of time to solve, unless $\mathcal{P} = \mathcal{NP}$. Just because a problem is $\mathcal{NP}$-hard, does not imply that the particular instances cannot be solved in polynomial time, but rather that there exists some class of instances that are too computationally challenging to solve to optimality.

Due to the numerous important applications of network flows, researchers consistently seek polynomial time algorithms to solve these problems. Some of the fundamental work with polynomial time algorithms on networks involves shortest path [45, 46], maximum flow [47, 48], minimum cost network flows [49], assignment and matching [50], and minimum spanning tree [51, 52].

Solutions to $\mathcal{NP}$-complete problems still need to be obtained. Since researchers cannot guarantee optimality in a reasonable amount of time, they have turned to heuristics. Heuristics sacrifice the optimality of the solution for a reasonable running time and are the focus of Section 2.5. Before discussing heuristics in greater detail, it is necessary to discuss a key tenant of this research: clustering.

2.3 Clustering

The idea of clustering is to group like items and keep dissimilar items in separate groups. Clusters can be found in almost everything we do. We group our friends based on different characteristics. For example, a married couple might have groups like “his friends from work” or “her friends from high school.” Unconsciously, people categorize items, people, and information into clusters as a better way to understand it.

Even something as simple as looking up a phone number in the local phonebook is a great example of clusters. First, if you are looking for a person’s phone number, you check the white pages; whereas, if you were to look for a business, you would check in the yellow pages. Obviously the yellow pages are organized by specialty, but you did not have to sort through the

information for the whole country. The phone numbers had already been clustered and packaged separately so that it made the information easier to find.

Information and data on the internet is extremely clustered. Some examples of clustering information are the social networking sites like Facebook, Myspace, LinkedIn, iLike, or Safari. On these websites a person is categorized by their previous and/or current schools, their marriage status, or the music and books they like.

Some other examples are the shopping websites like eBay, Zappos, or Eddie Bauer. When searching for an AM/FM receiver on eBay, it is easier to start in the electronics section, and then further reduce the search size by looking at only audio equipment. Or when looking for bowling or golf shoes on Zappos, it is easier to look in the Men's: Athletic section instead of just searching through the thousands of shoes on the website. These are just a few examples of the many different ways clustering is used on the internet.

A search for clustering or cluster algorithms on INSPEC results in over 225,000 articles, which indicates clustering's importance as a problem and its interest to researchers. Some examples of recent research are: a new overlapping clustering algorithm that provides non-exhaustive clustering [53]; database mining using cluster analysis to determine the clustering structure [54]; and a technique for recognizing fingerprints using a clustering algorithm [55].

2.3.1 Clustering Definitions

Finding good clusters is a complex topic. In fact, researchers cannot even agree upon what defines a good cluster or even what constitutes a feasible clustering solution. What defines feasible clustering varies, but typically can be defined as partitional, fuzzy or partial clusters. This thesis focuses on partitional clustering and only a short discussion of the other types of clustering solutions is provided here.

All clustering problems have a set of elements, possibly infinite, that need to be grouped into various groups. Here only finite elements are considered, but it is simple to extend these definitions to infinite sets. The input to a clustering problem is a set $N = \{1,\ldots, n\}$. A solution with k clusters to a clustering problem is $\mathcal{C} = \{C_1,\ldots, C_k\}$ where each $C_i \subseteq N$. Variations of the type of sets in $\mathcal{C}$ determine what type of clustering is desired.

For a partitional clustering solution $C_1 \cup C_2 \cup \ldots \cup C_k = N$ and $C_i \cap C_j = \varnothing$ for all $i \neq j$ and $i, j \in \{1,\ldots, k\}$. Thus, partitional clustering requires that each element of N is in exactly one cluster.

For a fuzzy clustering solution $C_1 \cup C_2 \cup \ldots \cup C_k = N$. Thus, fuzzy clustering allows elements of N to be in multiple clusters. Frequently in a fuzzy clustering some value is associated with each item in a cluster such that the sum of these values over all clusters equals one. Thus, an item can be in cluster C_1 30% of the time and cluster C_2 the other 70%.

Partial clustering solutions can be both partitional and fuzzy. The basic idea is that some items of N need not be clustered. Thus, some items are not in any clusters. It is fairly straightforward to see that a partial clustering solution could be modeled by simply allowing each such non-clustered item to be its own cluster.

Even if researchers can agree upon what type of clustering best models a problem, researchers may disagree on which clustering solution is the best. That is, given two distinct clustering solutions, determining which one is better is up for debate and is the focus of the next section.

2.3.2 Clustering Measures

Cluster validation determines the quality of a clustering solution. A good clustering solution would have a high similarity between entities in the same cluster and a low similarity between entities in other clusters. The level of dissimilarity within a cluster is expressed as the within cluster dispersion (δ) and the intercluster dispersion (Δ) is the measure of how different the clusters are from each other. In some way or anther, the objective of all clustering algorithms is to minimize δ and maximize Δ.

Clustering solutions are validated by essentially three criterion that include external criteria, internal criteria, and relative criteria. A detailed description of the various clustering validation methods can be found in Halkidi [56].

External criteria validation has some prespecified solution to compare to the clustering solution found. Normally, the prespecified solution is based on intuition or design. The primary

weakness of the external criteria technique is that the prespecified solution must be known in advance.

Validation based on internal criteria designates a real number, by way of an objective value, to the clustering solution. The objective values can be based on such parameters as distance, correlation coefficients, standard deviations, etc. just to name a few. Two solution's objective values are then compared to determine which solution is better.

Relative criteria validation evaluates a clustering solution by altering the input data. In this technique, some of the data might be removed and the new solution is compared to the actual solution. After repeating this process for a number of iterations, the technique may also include changing the number of desired clusters or adding artificial or real data.

One of the most important decisions in developing or using clustering algorithms is choosing a distance measure. Choosing different distance measures changes the similarity or dissimilarity between two items, which can greatly change the clustering solution.

One of the basic distance measures is Euclidean, or straight line, distance. The Euclidean distance from a, where $a = (a_1, a_2, a_3, \ldots, a_n) \in R^n$, to b, where $b = (b_1, b_2, b_3, \ldots, b_n) \in R^n$, is $\sqrt{\sum_{i=1}^{n} (a_i - b_i)^2}$.

The Euclidean distance just a particular instance of what is known as a p-norm. Formally, the p-norm distance between points a and b is defined as $\| a\text{-}b \|_p$ and is given by the formula $\sqrt[p]{\sum_{i=1}^{n} (a_i - b_i)^p}$, for $p \geq 1$.

Besides when $p = 2$, Euclidean distance, the other two most commonly used distance measures are when $p = 1$ and $p = \infty$. When $p = 1$ the distance is referred to as the Manhattan distance and is the distance between two points measured at right angles. In a plane, the Manhattan distance between two points, $p^1 = (x_1, y_1)$ and $p^2 = (x_2, y_2)$, it is: $|x_1 - x_2| + |y_1 - y_2|$. This represents the number of blocks you would have to walk, say north and east. When $p = \infty$ the distance is referred the maximum distance between any two coordinates, $\max_{i \in \{1,\ldots,n\}} \{| a_i\text{-}b_i |\}$.

Since researchers cannot agree on the best measures to use in forming clusters, it is easy to assume that they disagree on what makes a good clustering solution. In Figure 2.7, it is apparent that one clustering solution is about as good as the other. The top solution has vertex *t* with the cluster of vertices {*u, v, w, x*} and the other solution puts the *t* vertex in the {*m, n, o*} cluster.

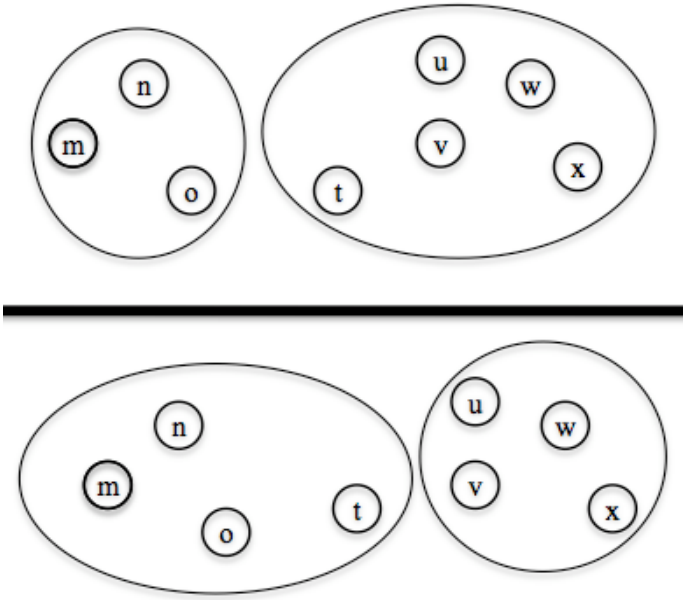

Figure 2.7 Two clustering solutions

Given that there are a number of different distance measures and choosing different measures can greatly affect the outcome of a clustering solution, researchers developed several clustering internal criteria validation measures. Some of these measures or objective functions include the silhouette index, Dunn's index, Daves-Bouldin Index, and *C*-index [57]. This research only discusses the silhouette index in greater detail.

In a given cluster C_n, where $n = 1, 2, \ldots k$, the silhouette index assigns each node x_i a quality measure s_i known as a silhouette width. The silhouette width is a confidence indicator of the inclusion of x_i in cluster C_j. The value of s_i for x_i in C_j is defined as: $s_i = \frac{b(x_i) - a(x_i)}{\max\{a(x_i), b(x_i)\}}$, where $a(x_i)$ is the average distance between the i^{th} node and all nodes included in cluster C_j and $b(x_i)$ is the minimum average distance between the i^{th} node and all nodes clustered in C_l, for $l = 1, 2, \ldots, k$, and $l \neq j$.

For any given cluster, the cluster silhouette index S_i is given as: $S_j = \frac{1}{|s_j|}\sum_{i=1}^{|s_j|} s_i$. For a given cluster solution $\mathcal{C}$ = {C_1, C_2, … C_k}, the global silhouette index, GS, is defined as: $GS=\frac{1}{k}\sum_{j=1}^{k} S_j = \frac{1}{k}\sum_{j=1}^{k}(\frac{1}{|s_j|}\sum_{i=1}^{|s_j|}\frac{b(x_i)-a(x_i)}{\max\{a(x_i),b(x_i)\}})$. The values of GS are always between -1 and 1. When comparing two clustering solutions, the solution with the higher silhouette index is the better solution.

2.3.3 Clustering Heuristics

Given all of the confusion regarding the definition of a feasible cluster and an optimal clustering solution, it should come as no surprise that there are numerous heuristics that attempt to generate quality clustering solutions. This section discusses only a few of these.

Clustering heuristics are generally said to be either hierarchical or partitional. Hierarchical clustering builds from one cluster to the next in a successive process over a number of iterations. It creates nested clusters which can best be seen in a tree structure called a dendogram. Partitional clustering attempts to directly decompose the data set into a set of disjoint clusters.

Hierarchical clustering heuristics either start with all elements in one cluster and then break this cluster into smaller clusters, called divisive clustering, or start with each element in its own cluster and combine similar clusters into fewer clusters, called agglomerative clustering.

Dendograms are tree diagrams used to illustrate the arrangement of clusters (Figure 2.8). Notice that a divisive heuristic starts at the top and works down; whereas, an agglomerative method starts at the pendants and works up. In either case, the method stops at some level and reports the clustering solution.

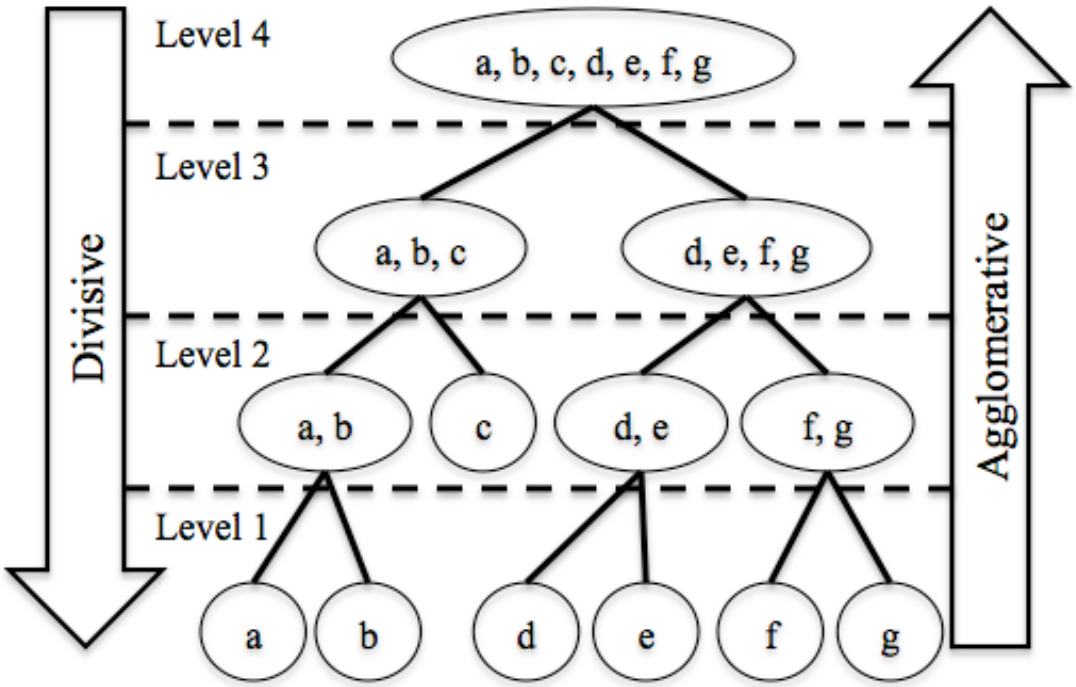

Figure 2.8 Dendogram of hierarchical clusters

Several common disadvantages of hierarchical clustering are the tendency to break large clusters, the non-convex shape, and a lack of robustness [58]. Additionally, the dendograms are typically broken into clusters by some ad hoc methods. The nature of this method means that the clustering decision cannot be undone and may adversely affect the final clustering solution.

From a data set, partitional clustering attempts to determine how many clusters should be in the solution and also what these clusters should be. Many researchers have used partitional strategies to find good clustering solutions [59, 60, 61]. The most common partitional clustering method used is *K*-means clustering [62].

The computational results in this thesis are compared to *K*-means and so this method is explained in detail here. The input to *K*-means is a data set of real numbers and an integer k representing the number of clusters. The data is divided into k clusters, with the clusters usually being randomly assigned. The centroid or average point of each cluster is determined. The next iteration compares the distance from every data point to every centroid and if a data point is closer to another centroid, then that point moves to that cluster. New centroids are calculated and the process repeats for a set number of iterations or until no data points change clusters.

A major disadvantage of most partitional clustering heuristics and *K*-means is that the number of clusters must be known before running the algorithm. Obtaining the optimal number of clusters for a given data set is an $\mathcal{NP}$-hard problem [63]. Making an algorithm that must determine the optimal number of clusters only increases the computational difficulty. As a way of avoiding this difficulty, most of the current clustering heuristics require the user to input the number of clusters desired. Given a large data set, the user does not know the number of clusters in most practical applications. Ideally the heuristic should get reasonably good clustering solutions without providing the number of clusters.

Some of the other disadvantages of most clustering heuristics are that they get stuck at local optima, their inability to detect non-spherical clusters, and the formation of clusters with minimum similarity [58].

As mentioned earlier, solving for the optimal clustering solution is an $\mathcal{NP}$-hard problem. Due to the difficulty of clustering problems, researchers use heuristic methods that are solve rather quickly. The major disadvantage to heuristics is that they cannot prove that the given solution is the optimal solution to the problem. This thesis explains heuristics in greater detail in the next section.

2.4 Heuristics

Since heuristics cannot prove optimality, they attempt to generate excellent solutions in a reasonable amount of time. A search of heuristics on the INSPEC database results in over 54,000 papers and this area is a major area of research. The reason that so much research is performed on heuristics is that the problems need to be solved and finding the optimal is just too challenging for many problems.

2.4.1 Neighborhoods

A neighborhood is a fundamental concept of a heuristic. Given a solution of a problem, a solution's neighborhood consists of all other solutions that are sufficiently "close" to this solution, where "close" is well-defined in some sense. Formally, let Π be some problem. Let I be an instance of Π and let X represent all feasible solutions to problem Π on instance I. Then

the neighborhood of $x' \in X$, $N_{x'}(\varepsilon)$ is the set of all $x'' \in X$ such that the $|x'-x''| \leq \varepsilon$ where || is some well defined distance measure and ε is some threshold, thus, $N_{x'}(\varepsilon)=\{x'' \in X: |x'-x''| \leq \varepsilon\}$.

Neighborhoods are frequently classified as large or small. Small neighborhoods only have a few solutions in each neighborhood, while large neighborhoods typically have exponentially many different solutions in each neighborhood. Thus, large neighborhood heuristics tend to produce better results, but typically require substantially more time per iteration [16].

For instance, assume that a heuristic has a clustering solution given in Figure 2.9. Furthermore assume that "close" is defined as one vertex changing clusters. Then a neighboring solution is given in Figure 2.10. Notice that node *t* has changed from one cluster to the next. In this case, it is easy to count the size of the neighborhood. Each node can move to one of two different clusters, so there are 2 * 12 = 24 solutions to this first solution's neighborhood. Thus, this well-defined measure creates a small neighborhood.

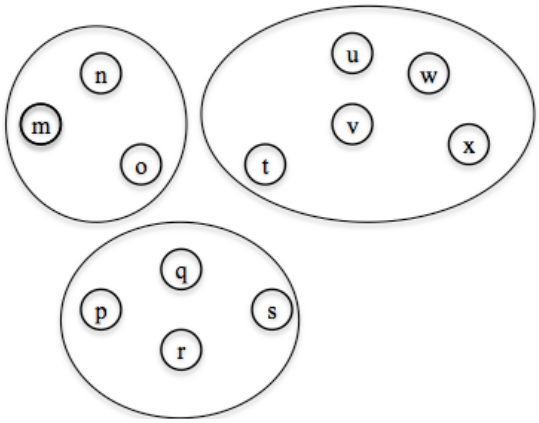

Figure 2.9 Clustering solution with 12 nodes

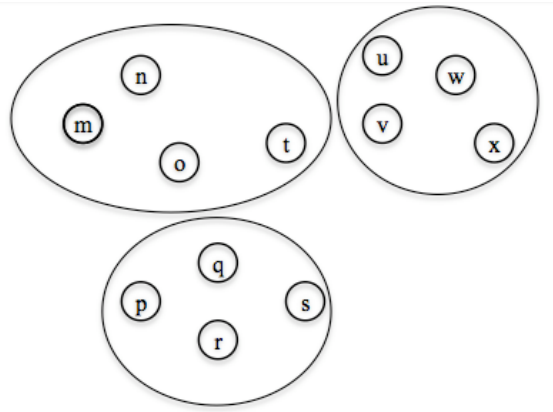

Figure 2.10 Neighboring clustering solution to Figure 2.9

In contrast, consider the well-defined measure to consist of all clustering solutions consisting of three vertices changing clusters. In this case any three nodes could move to any of the other clusters or stay in its current cluster. Any three nodes can be selected to move and so there are $\binom{n}{3} = n(n\text{-}1)(n\text{-}2)/3!$ different sets of 3 nodes that could be selected. Any of these three nodes could move to any of the k clusters. Thus, there are $\binom{n}{3} k^3$ different solutions in a solution's neighborhood including the original solution. In this small case with this is $\binom{12}{3}3^3 =$ 5,940 different neighboring solutions. As the number of clusters or nodes increases, the size of the neighborhood grows rapidly. Consequently, "close" in this case defines a large neighborhood.

The bulk of heuristics start with a feasible solution and move to a neighboring solution. This process continues through many iterations. Some heuristics force a move to the best neighboring solution, while others may move to any neighboring solution including worse solutions. Many of these heuristics have been applied to numerous problems and such heuristics are called metaheuristics.

2.4.2 Metaheuristics

Metaheuristics are defined as "a class of approximate methods, that are designed to attack hard combinatorial optimization problems where classical heuristics have failed to be effective

and efficient" [64]. Some common metaheuristics are greedy algorithms, simulated annealing, and tabu search. Numerous papers have been written on this topic and Aarts and Lenstra [65] provide an excellent review of this material.

The goal of a metaheuristic is to search for a feasible solution and move such a solution toward a good solution in an acceptable amount of time. Typically metaheuristics are iterative processes that move from a neighboring solution to another neighboring solution according to some well-defined properties. Metaheuristics may find the optimal solution, but they cannot prove such a solution is an optimal solution.

The hill climbing heuristic is one of the most commonly used metaheuristics. The objective of a hill climbing heuristic is to start at any feasible solution, x^o, and set i to 0. The main step finds the neighborhood of x^i and chooses x^{i+1} to be any solution of x^i's neighborhood that has a better objective value than x^i. The heuristic terminates after no such move can be made and x^i is now a local optimal solution.

Another metaheuristic that researchers commonly use is called tabu search. Tabu search is a local search technique that uses memory structures to mark potential solutions as "taboo" or untouchable. Begin with a feasible solution x' and an empty Tabu list, T and a threshold k. The next step is to find a neighbor of x', if the new solution is not in T, then set x' = new feasible solution. Add x' to T and if T is too long, $|T| > k$, remove the oldest entry in the list. This process repeats for a set number of iterations and reports the best found solution.

The purpose of the taboo list is to ensure that the algorithm does not repeatedly return to solutions that have already been found. In this manner tabu search sometimes moves from a better solution to one that is worse in an attempt to avoid local optima.

Various other metaheuristics exist. Simulated annealing [66, 67], ant colonization [68, 69], and particle swarm [70, 71] are a few of the other metaheuristics previously developed. These metaheuristics use different methods to move towards an optimal solution and attempt to avoid staying at a local optimal solution.

2.5 Epidemic Modeling

The best way to know how a government is to respond to an epidemic is to model its spread and know how best to stop the epidemic from spreading. Modeling epidemics is the only

feasible method of war-gaming the government's response should an outbreak occur. With different models, a government can establish policies and put measures in place to deal with a number of possible scenarios.

One of the basic methods for modeling an epidemic is to categorize each entity into a particular state at any given moment of time based upon their current health. In this manner, individuals progress from one state to the next. Different models assume different classes of states to accurately represent the progression of a disease on the host. Some of the more commonly used models are presented here.

The most basic model for the spread of infectious diseases follows a susceptible-infectious-recovered (*SIR*) pattern. If an individual is classified as *S*, then that individual is in the susceptible state and could possibly contract the disease. Once the subject contracts the disease, the next state modeled is the infectious state *I*, when that subject could possibly infect other subjects. The last state of the *SIR* model is the removed or recovered state *R*. In this final state the subject no longer has the disease and cannot contract the disease again. Observe that if the subject dies it is classified as recovered, which seems mean. Fundamental to this model is that an individual cannot move from *R* to *S*. For more information on the *SIR* model see Anderson [72] or Brauer [73].

The *SIR* model is rather simplistic, however, researchers are constantly working to improve to the *SIR* model. Some models may move an entity from the *R* state back into the *S* state after a period of time signifying that the person was again susceptible to contract the disease. Some recent work in this area includes Iacoviello and Liuzzi [74] and Neal [75].

While *SIR* provides an adequate model, many researchers have augmented this model with other states. The *SEIR* model adds an exposed state following the susceptible state. This exposed state effectively models the subject coming in contact with an infectious subject, but not being immediately infectious. Thus the individual contracts the disease and becomes infectious, but in the exposed state the individual cannot spread the disease. Recent work using the *SEIR* model can be seen in Zhao [76].

Another model is the *SIS* model, where subjects belong to one of two groups. This method can effectively model the common cold or other diseases that do not increase a subject's

immunity to it. Zhang and Zhao [77] recently used the *SIS* model to study the speed at which a disease might spread and the pattern that it travelled in.

Many of these time state models have provided interesting theoretical results involving whether or not a disease is naturally eliminated. However, these examples frequently have unrealistic underlying assumptions, such as every individual contacts every individual and spreads the disease to every other person with equal weight. Once these basic assumptions are removed, the mathematics becomes extremely difficult and so graph theory and simulation are used to study more realistic cases.

2.6 Graph Theory, Simulation, and Infectious Diseases

A great deal has been done in the area of modeling infectious diseases with graphs. The primary graph theory method is to create a contact network to represent the spread of a disease. Formally, a contact network $G = (V, E)$ has the vertex set V represent the individuals of the study and their interactions are symbolized by the edge set E. Weights, p_{ij}, are assigned to edges and indicate the probability that an infectious disease is transmitted from person i to person j, under the assumption that person i can spread the disease and person j is susceptible to the disease.

Several researchers have used a contact network to model the spread of a disease. Rvachev [78] modeled an influenza outbreak on a system of 128 Russian cities. In this model, the researchers took into account the transportation network and other parameters to model the influenza outbreak as close to reality as they could. Fefferman [33] uses graph theory in epidemic modeling to show how disease models on static networks is different from disease models on dynamic networks.

Recently many of the research papers have combined both theory and simulation of a contact network. Guimaraes [79] models the effects of an infectious disease on the social network of killer whales. In another instance of epidemic models using graph theory, Britton [34] models simple epidemics and some local vaccination strategies. Chung [80] successfully modeled SARS (severe acute respiratory syndrome) and helped to inform public policy makers on epidemic dynamics.

To date, EpiSims, developed by Los Alamos National Laboratory, is the largest simulation of the spread of an epidemic. EpiSims is an individual based epidemic model that

simulates the interactions of millions of people using clusters of high-performance supercomputers. With the EpiSims model, researchers are able to model the spreading of a virtual pathogen and test different intervention methods on a city the size of Portland, Oregon [81].

CHAPTER 3 - Epidemic Simulation

As stated earlier, this research began by creating a simulation to model the spread of an infectious disease in rural Kansas. The motivation behind focusing on rural Kansas is that people's movements and the number of law enforcement personnel become serious issues if it is necessary to quarantine a rural area. In contrast to a large city, people in rural areas travel much further to do something as simple as getting groceries or household goods. Similarly, there are fewer police officers and a great deal more land to manage if a quarantine is required.

The objective in developing the simulation core is to have a simulation package that is easily adaptable to any disease. Thus, a person studying an infectious disease can provide Dr. Easton's research team with information regarding the disease and within a few hours the simulation core can be adjusted to model a disease and test mitigation strategies.

The simulation accuracy is directly dependent on the accuracy of the input given to the research team. An individual desiring to use the core simulation should have knowledge about how the subjects interact, how the disease is spread from one individual to another, and how different individuals advance through the disease characteristics. Detailed information about the disease is also important to adjust the simulation to better reflect the disease type.

The remainder of this chapter is organized as follows. The next section describes the core simulation and how it functions. Building the core simulation was a joint effort by fellow graduate student Kyle Carlyle and myself. The final section describes how the core simulation is used to develop a sample model for the small rural town of Clay Center, Kansas. Undergraduate students Matthew James and David Willis did substantial work on modeling the topology of Clay Center.

3.1 Epidemic Simulation Core

On a basic level, the simulation needs two sets of input to accurately model a disease and specific area type. The accuracy of the information put into the simulation directly impacts the

accuracy of the disease and area modeled. While it is feasible to work with very vague information, it is not desirable.

First, some basic concept of a contact network of the simulated area must be known. This may be something as general as saying people in this area travel a long ways and come into contact with only a few people. It is better to have more information in order to better model the area, but a rough estimate can be built from basic information.

Second, the researcher must know the basic characteristics of the disease to be modeled such as how the disease spreads between individuals (i.e. skin contact, air, etc.) and an estimate probability to model the spread of the disease. An idea of the infectiveness of the disease will also help determine probabilities that the disease is spread from one virtual person to the next.

3.1.1 Contact Network

As stated earlier, given an approximation of the people to be modeled, the simulation core can adapt to different situations. Mostly the information needed concerns how far the modeled individuals travel on a normal basis and how many other people they come into contact with when they travel. With this information, the parameters that build the contact network can be adjusted to better match the individuals modeled.

Researchers can customize the simulation to model the qualities of the disease based on input about the disease. In this manner, the simulation core can adjust from simulating the spread of a SARS epidemic to the outbreak of an influenza epidemic to the common cold with minimal changes involved in adapting the simulation code.

In order to develop the contact network, each individual in the simulation is represented as a node. When the simulation develops a graph, it assigns x and y locations to each node to symbolize each person's location within the model. With the basic graph of nodes, an edge between two nodes represents contact between those two people. These edges are then assigned a probability that, if person A is infectious and person B is susceptible, then the number represents the probability that person B would contract the disease from person A. Section 3.2 describes how to build such a contact network utilizing the proximity of individuals to each other in the Clay Center example.

The final portion of input to the simulation core is the health state of each individual. The simulation core is a discrete state simulation, that is, this model is similar to the SIR and SEIR models discussed in Section 2.5. Consequently, the simulation core assigns each entity in the model to a health state. Consequently, some portion of nodes must be known to be infectious, susceptible, etc. For this research, the disease is assumed to start with a single infectious individual.

From this modeled contact network, the simulation core can determine how the disease spreads through the population. To help explain how the simulation core functions, consider the contact network given in Figure 3.1. Assume person *A* is infectious and persons *B* and *C* are susceptible.

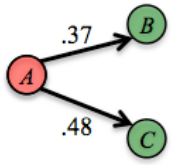

Figure 3.1 Initial contact graph example

3.1.2 Spreading the Disease

When the simulation is running and the requirements are met (i.e. person *A* is infectious and person *B* is susceptible), then the simulation generates a uniform random number that is compared against the probability that person *B* contracts the disease. If the random number is higher than person *B*'s probability, then person *B* does not contract the disease and the simulation moves to the next instance of contact between an infectious person and a susceptible one to determine if the next person contracts the disease. If, however, the random number is lower than the probability on the edge between nodes *A* and *B*, then person *B* contracts the disease.

In the small example from Figure 3.1, person *A* has a 37 percent chance of passing the disease to person *B* and there is a 48 percent chance that person *C* contracts the disease from *A*. The different colors of the nodes represent that person *A* is contagious and people *B* and *C* are susceptible. Given this situation, the simulation would generate one random number to determine if *A* infects *B* and one random number for the same type of contact between *A* and *C*. Assume the random number for the *A* to *B* contact is .29 ≤ .37, so *B* contracts the disease. Also

assume that the random number for the *A* to *C* contact is .61 > .48, then *C* does not contract the disease from *A*. Since there are no more similar situations in this small graph, this time period has been evaluated and the simulation advances to the next time period.

Due to the fact that not every person has the same probability of contracting a disease, the simulation allows the researcher to adjust the probability parameters. With the simulation core, researchers can model a disease that is more infectious by increasing the probability that a simulated disease is contagious. Clearly, researchers can decrease the probability as well.

3.1.3 States, Disease Tracks and Times in States

Once an individual contracts the disease, the simulation core allows the disease to affect different individuals in different manners by aligning the individuals with different "disease tracks." In this way, the simulation can be adjusted to model both an older population that is more susceptible to disease and has a more difficult time fighting it and a younger population that is in better shape and does not exhibit the same set of symptoms.

An historical example of someone that had a different reaction to a disease would be Mary Mallon, or Typhoid Mary. She was completely healthy, but was a carrier of typhoid fever. She did not show any symptoms of the disease and continued to cook at restaurants and is believed to have eventually infected around 50 people, three of whom died.

The final input of the simulation core is a set of disease tracks with probabilities that an individual that contracts the disease follows a certain disease tracks. This is easily explained by returning to the example from Figure 3.1.

Assume that this disease has two disease tracks, SIR and SEIR. Furthermore, assume that there is a .40 probability that an individual that contracts the disease follows an SIR model and a .60 probability that the individual follows an SEIR model.

Since *B* has contracted the disease, the simulation core generates a uniform 0 to 1 random number and compares this to the disease track probabilities to determine, which disease track or progression the individual follows. In this case assume the random number is .59. Since .59 > .4 and .59 < .4+.6, person *B* follows the SEIR disease track.

Once a disease track is determined, then the length of time in the next health state must be determined. Now that person *B* has transitioned from state *S* to state *E*, the simulation core determines that *B* is in the exposed health state for three days before becoming infectious. As *B* finishes the third day of exposed time, the simulation immediately moves *B* to the infectious state and determines that *B* is infectious for two days before moving to the recovered health state.

Obviously, the distribution of this time can be set to various random times and the simulation core can easily be adjusted to accurately model the aggressiveness of the disease as well by reducing the time before a person is infectious after contracting the disease. If the disease modeled is similar to the common cold, where a person starts exhibiting the symptoms of the disease usually within 24 hours, +/- 2 hours, after exposure, then the time a node is marked as exposed can be adjusted to be from 22 to 26 hours, as a uniform or normal distribution with time periods set to hours. In similar fashion, if the disease requires a longer period before symptoms appear, the simulation can be adjusted to the longer time frame. Each of these time periods must be in the same unit for the simulation to accurately portray the disease.

Figure 3.2 depicts the state of the contact network from Figure 3.1 on the second time period where yellow represents exposed. The simulation continues this process for a set number of periods and reports the health states of the individuals upon termination of the model.

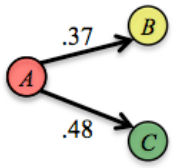

Figure 3.2 Adjusted contact graph example

3.2 Example of the Simulation Core on Clay Center, Kansas

A sample simulation is developed for the small town of Clay Center, located in northeast Kansas. Clay Center and the surrounding rural areas where people would travel to a central population hub in order to meet with friends for dinner, get groceries and supplies, or go to a religious service; these types of interactions are all normal under most circumstances. The spread of infectious disease on this town of 4,600 people can be simulated rapidly (in less than

30 seconds). David Willis and Matthew James, the undergraduate students on this research team, primarily focused on creating the contact network used to replicate Clay Center, Kansas.

The reason for using Clay Center, Kansas as a model to base this simulation on is that Matthew James is from Clay Center and has first hand knowledge of the normal interactions that take place in a town that size. Also, the proximity to Kansas State University makes it easily available to visit and the townspeople are extremely helpful in completing the research team's surveys.

The simulated, and completely fictional, disease was dubbed WAJEC, based on the last names of the creators that developed the simulation. The initial parameters used for this simulation are listed in Table 3.1. In order to better visualize the spread of WAJEC, a graphical user interface (GUI) was developed by the researchers. The GUI rather accurately displays the town proper and the surrounding rural areas in Figure 3.3.

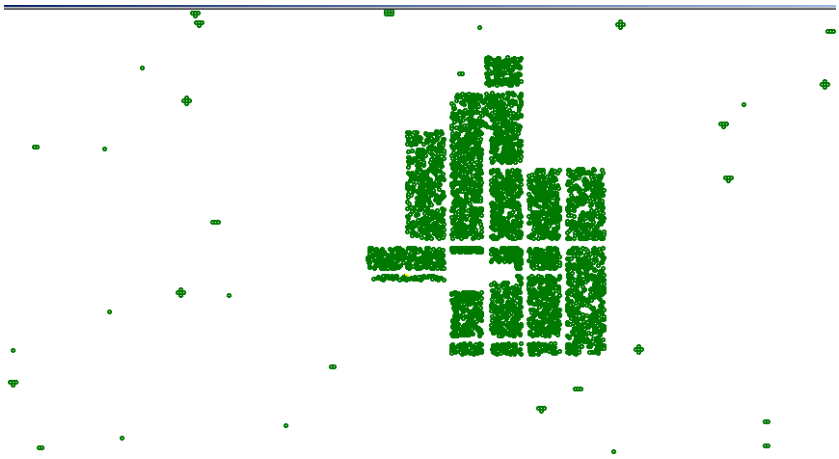

Figure 3.3 Healthy Clay Center

Figure 3.3 shows the population as healthy green dots. Notice also that there are small, wide-spread, bunches of nodes outside the main town area that represent the farmers and people living in the surrounding rural areas that use Clay Center as their place to do business. In the figure it is also apparent where the streets in Clay Center are by the empty white lines in an approximate grid pattern and the blank area representing the town center, which is not a residential area. It is now necessary to look at some of the parameters that setup the WAJEC simulation on the town of Clay Center, Kansas.

3.2.1 Parameters

There are a number of parameters that researchers can adjust to fit the disease and location they are simulating. Some of these parameters control the contact network, while others affect the proliferation of the disease and how it impacts the various individuals differently.

The parameters involved with developing the contact network are listed in Table 3.1. Notice that the number of nodes is equal to the number of individuals to be modeled in the simulation. Families were grouped in the simulation core and can be seen in Figure 3.3 by the fact that the nodes in rural areas are bunched in groups of one to four.

Maximum nodes	4,600
Maximum edges	300
Short distance	10
Medium distance	30
Long distance	240
Short edge probability	20%
Medium edge probability	5%
Long edge probability	1%
Short probability maximum	25%
Medium probability maximum	25%
Long probability maximum	10%

Table 3.1 Contact network parameters

In the simulation, the short distance of 10 is roughly equivalent to one city block. The medium distance is then three blocks and the long distance is two miles. The other parameters are more easily explained if you consider their relationship to one node, *A*. The short edge probability of 20 percent translates to mean that 20 percent of the nodes within the short distance (one city block) are considered to actually have contact with node *A*. The short probability maximum number of 25 percent means that of the nodes that have a short distance contact with *A*, 25 percent have a probability of contracting a disease when the infectious and susceptible criteria are met. This logic follows for each of the distance categories.

3.2.2 Disease Tracks

In the epidemic model used for this research, the researchers added three possible tracks for the different possible reactions by different people. In the simulation, these tracks take an individual through a variety of different states similar to the SIR and SEIR models. Table 3.2 shows the three disease tracks along with probabilities that WAJEC could take. Also important to note is that for the purposes of this simulation, each health state was set to a constant three days.

State	Track 1 (probability .45)	Track 2 (probability .50)	Track 3 (probability .05)
1	Susceptible	Susceptible	Susceptible
2	Exposed	Exposed	Dormant
3	Contagious no symptoms	Symptoms not contagious	Contagious no symptoms
4	Contagious w/ symptoms	Contagious w/ symptoms	
5	Symptoms not contagious	Recovered	
6	Immune		
7	Recovered		

Table 3.2 Disease Tracks

Notice that in Track 1, the entity moves through a gambit of being contagious and having symptoms, and then is immune to the disease and cannot contract it any longer. If the simulation is used for a disease like the common cold, this track could be adapted so that individuals were never immune to the disease.

3.2.3 Simulations

At this point it is advantageous to see more of the graphical representations of the simulation. First, it is necessary to explain what the different colors represent in the graphic. Table 3.3 is a legend to the colored nodes representing the people of Clay Center as WAJEC spreads through the area.

Green	Susceptible
Yellow	Exposed
Red	Carrier
Purple	Contagious
Blue	Symptoms, not contagious
Teal	Immune
Black	Dormant
Olive	Recovered

Table 3.3 Graphic representation legend

The top depiction in Figure 3.4 shows the spread of WAJEC after one week's time and notice the lower picture of Figure 3.4 how WAJEC has spread after ten days. Notice the rapid progression of WAJEC after an additional three days.

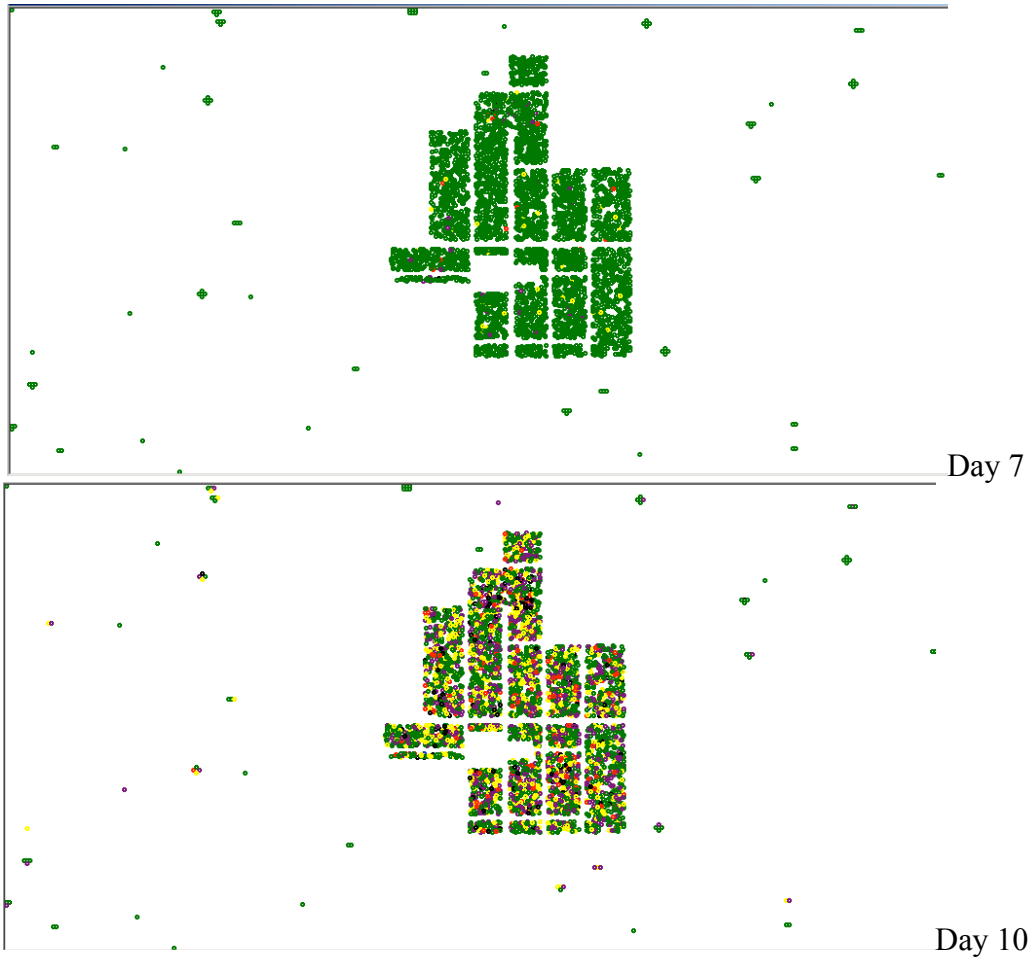

Figure 3.4 WAJEC base case

In order to see the adaptability of the simulation core, the researchers adjusted WAJEC so that the short, medium, and long distances were twice as long as in the base case (now 20, 60, and 480 respectively). The two parts of Figure 3.5 show the resulting contact network after doubling the distance parameters. It is apparent that the epidemic travels further by the fact that there are more individuals sick and that they cover the entire region. Notice also that in this instance, at day 10, there are many more individuals that have progressed further along in their disease track.

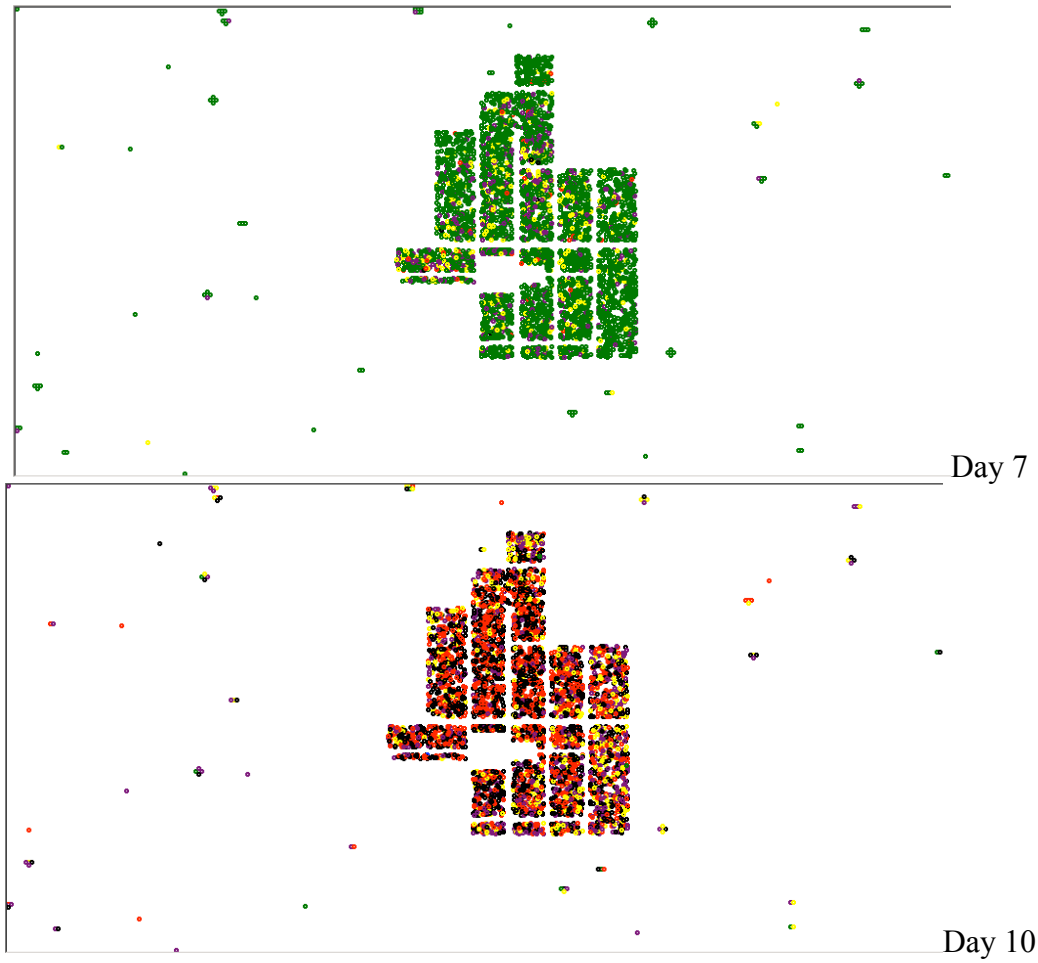

Figure 3.5 WAJEC with distance parameters doubled

Now, reduce distances back to the base case (10, 30, and 240) and double the maximum probabilities to .5, .5, and .2, respectively. Thus, the disease is twice as likely to spread from one node to the next. The resulting GUI output can be seen in Figure 3.6.

Notice that in this case, the disease is not as widespread through the community, but that the individuals that have the disease are further along in their respective disease track. This indicates that they contracted the disease earlier due to the higher probability that it spreads. Also notice that at day 10, many more individuals are infectious and exposed when compared to day 10 of the base case.

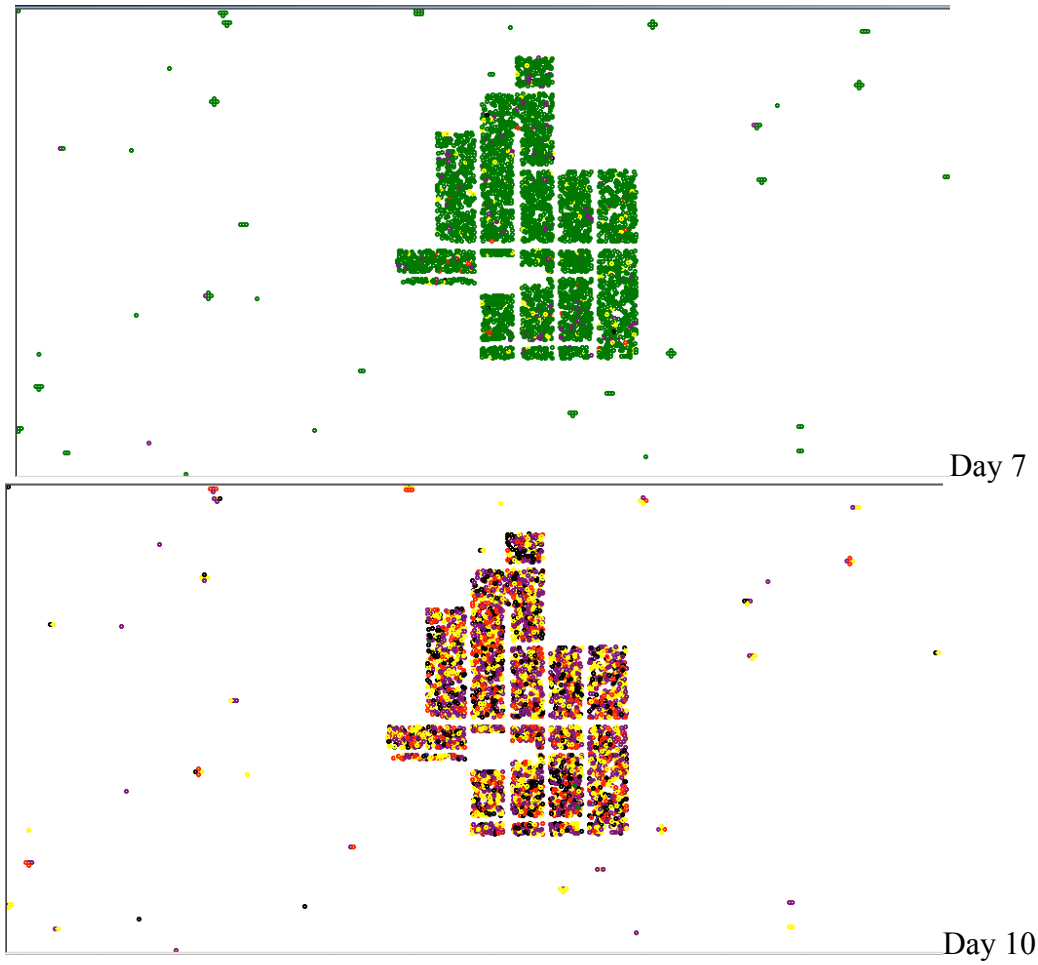

Figure 3.6 WAJEC with probability parameters doubled

By only comparing graphics of in Figures 3.4, 3.5, and 3.6, it is apparent that having a contact network where people travel twice as far can be devastating to the population of Clay Center. When the distance parameters are doubled, the infection spreads throughout the city by day seven, but when the probability parameters are doubled WAJEC does not infect as many of the populace.

Solely considering the GUI figures, it is obvious that WAJEC needs to be contained in less than ten days. Ideally, health officials would recognize the epidemic sooner than seven days and be able to impose a quarantine plan. Announcements of a possible epidemic should be made, by radio, TV, posters, and all other means available, informing the local population to reduce their contacts and effectively shrink the contact network available to the disease. By reducing the distances travelled and interactions between infectious and susceptible people, WAJEC could be contained, possibly to only the town of Clay Center.

Of course, when considering quarantines, it is best to know how stop the disease before it starts to spread. Being able to simulate an epidemic and then war-game possible mitigation techniques could save countless lives.

This section demonstrates a few of the many adjustments in which the simulation core can be changed to model different types of diseases or differences in the way that the modeled population travels and interacts. As mentioned before, other ways in which to adjust the simulation include the disease tracks and parameters, time units used in the simulation, and health state time distributions. This tool is ready to be applied to a particular disease in a specific area.

CHAPTER 4 - Clustering Heuristic using Integer Programming

Once the simulation core was developed, this research continued with the goal of building a clustering integer program (CIP). The idea was that a CIP could solve a clustering solution and be used to develop a quarantine plan in the event of an epidemic outbreak. After determining that the CIP could only solve extremely small graphs, the research developed the Clustering Heuristic using Integer Programming (CHIP). CHIP is a large neighborhood heuristic that provides quality clustering solutions on reasonably sized contact networks. Some computational results are presented to show that CHIP provides clustering solutions that are substantially better than the *K*-means method; the most commonly used clustering heuristic.

4.1 Clustering Integer Programming Formulation

There are numerous different types of integer programs that can be used to formulate a clustering problem. This is the presentation of one such formulation. First there are numerous different objective functions and distance measures that could be used. This objective function must be linear in order to apply integer programming methods.

Developing a good linear objective function for the clustering problem is not easy. This objective function must encourage adjacent nodes with high probabilities to be in the same clusters, while also penalizing for the number of clusters.

For this thesis, the objective function minimizes two penalties. The first is the absence of edges within the cluster; while the second is a penalty for having too many clusters. Thus, the two extreme cases (one cluster or *n* clusters) can be avoided. Therefore, the objective function for is thesis is to minimize one minus the probability that nodes *i* and *j* are in the same cluster plus α times the number of clusters. Consequently, edges with a large weight (p_{ij}) would try to be included in the same cluster, while edges with small weights would be encouraged to have the nodes in separate clusters. Higher values of α encourage fewer clusters.

An example of how to calculate a cluster solution can be seen in Figure 4.1. Assume α is 10, and the clusters are labeled A, B, and C. Notice that within cluster B all probabilities are .5 or higher except for one edge with a value of 0. Thus, B should be a reasonable cluster. Also

notice that all three clusters are each only one edge short of being a clique.

The objective function relative to cluster A is calculated as follows:

$(1\text{-}p_{mn}) + (1\text{-}p_{mo}) + (1\text{-}p_{no}) + (1\text{-}p_{nt}) + (1\text{-}p_{ot}) + (1\text{-}p_{mt}) = .2+.3+.1+.2+.4+1 = 2.2$

Similarly, clusters B and C increase the objective function by 2.5 and 2.7, respectively. Therefore, the objective function for this clustering solution is 2.2 + 2.5 + 2.7 + 3(10) = 37.4.

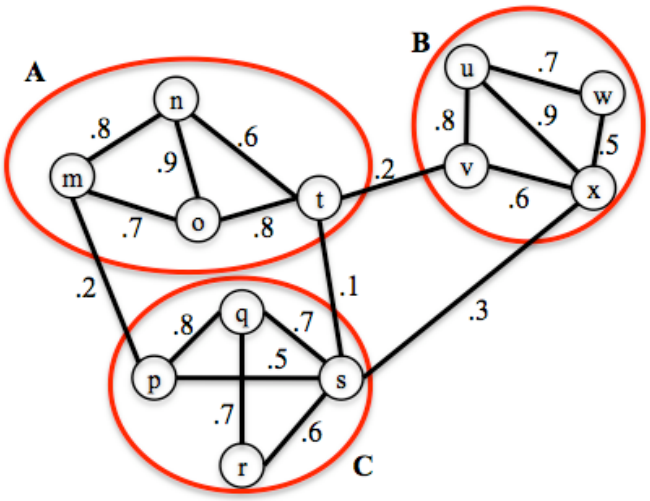

Figure 4.1 Clustering example

For contrast consider the clustering solution where each element is its own cluster, then the objective value is 10(12) = 120. Alternately, if there were only a single cluster, then the objective value is 64.6. Thus, the A, B, and C cluster solution is better than these two extreme cluster solution cases.

Clearly, a higher value of α encourages fewer clusters and if α is above 25, then the single cluster solution is better than this three cluster solution. Similarly, if α is below .8, then the solution with all nodes in their own clusters is better than this three cluster solution. The value of α has an inverse impact on the number of clusters in the optimal solution.

The input to the integer program is a contact network $G = (V, E)$ with the weight of the edge given as p_{ij} being the probability that the disease is spread from individual i to individual j. The value of α is input by the user to adjust the number of clusters.

Clustering Integer Program (CIP)

Sets:

$N = \{1,\ldots, n\}$ the nodes of G.

$K = \{1,\ldots, k\}$ the clusters in G.

Parameters:

p_{ij} = the probability assigned to the edge $\{i, j\}$ in G.

α = the penalty of having an additional cluster.

Decision Variables:

$full_k = 1$ if cluster k contains at least one node; 0 otherwise for all $k \in K$.

$same_{ij} = 1$ if nodes i and j are in the same cluster; 0 otherwise for all $i, j \in N$.

$x_{ik} = 1$ if node i is in cluster k; 0 otherwise for all $i \in N$ and $k \in K$.

Objective Function:

$$\text{Minimize} \sum_i \sum_j (1-p_{ij}) same_{ij} + \alpha \sum_k full_k \quad (1)$$

Subject to:

$$x_{ik} - x_{jk} + same_{ij} \leq 1 \qquad \forall i \in N,\ \forall j \in N,\ \forall k \in K \quad (2)$$

$$-x_{ik} + x_{jk} + same_{ij} \leq 1 \qquad \forall i \in N,\ \forall j \in N,\ \forall k \in K \quad (3)$$

$$x_{ik} + x_{jk} - same_{ij} \leq 1 \qquad \forall i \in N,\ \forall j \in N,\ \forall k \in K \quad (4)$$

$$x_{ik} \leq full_k \qquad \forall i \in N,\ \forall k \in K \quad (5)$$

$$\sum_k x_{ik} = 1 \qquad \forall i \in N \quad (6)$$

where $same_{ij} = \{0,1\}\ \forall\ i, j \in N$; $full_k \in \{0,1\}\ \forall\ k \in K$; $x_{ik} \in \{0,1\}\ \forall\ i \in N$ and $k \in K$.

The objective function (1) minimizes one minus the probability that nodes i and j are in the same cluster plus the penalty for having a large number of clusters. The idea in the first half of the equation is that the edges with a large weight (p_{ij}) would be included in the same cluster,

while edges with small weights would be encouraged to have the adjacent nodes in separate clusters. With higher values of α, multiplying by the number of clusters encourages keeping the number of clusters smaller.

Constraints (2), (3), and (4) ensure that if node *i* and node *j* are in cluster *k* then the values for x_{ik}, x_{jk}, and $same_{ij}$ align to the correct value of either 0 or 1. This is perhaps most easily portrayed in the following truth table, Table 4.1. From the table, one can see that the constraints stop the instances where x_{ik}, x_{jk}, and $same_{ij}$ violate the logic. If both nodes are in cluster *k*, then the values for x_{ik}, x_{jk}, and $same_{ij}$ should all equal 1. If only one node is in cluster *k*, then only the appropriate x_{ik} or x_{jk} would have a value of 1 and $same_{ij}$ would have a value of 0. It is trivial to validate that each of these "true" points from the table satisfy each of the three constraints.

$x_{i,k}$	$x_{j,k}$	$same_{i,j}$		Violated constraints
1	1	1	True	
1	0	0	True	
0	1	0	True	
0	0	1	False	(4)
1	1	0	False	(4)
0	1	1	False	(3)
1	0	1	False	(2)
0	0	0	True	

Table 4.1 Truth table agreements that correspond with constraints (2), (3), and (4).

The variables $full_k$ represent whether or not there is an element in cluster *k*. Thus, if $x_{jk} = 1$, then node *j* is in cluster *k* and thus cluster *k* is not empty. Clearly, constraint (5) forces this condition.

Constraint (6) is in place to make sure that for all possible values of *i* corresponding to each cluster *k,* only one x_{ik} is equal to 1, in order to ensure that each node belongs to only one cluster. This constraint is necessary due to the fact that the focus of this research was on developing quarantine groups for people and people cannot simultaneously belong to both the quarantined section and the non-quarantined section.

In some preliminary results, CPLEX 10.0 [82] could not solve CIP even for relatively

small instances. In fact, a graph with 30 nodes ran for more than 36 hours and still had not optimally solved the problem. Recently, substantial research has been performed on symmetry cuts [83, 84, 85]. The idea of a symmetry cut is to eliminate some of the multiple solutions that are identical.

4.3 Symmetry Cuts

The CIP has substantial symmetry, notice that the same solution can simply permute the clusters and if there are k clusters, then there are $k!$ identical solutions. An example of symmetry is when you have two solutions that are the same thing, such as the cluster solution $\{\{x, y, z\}, \{a, b\}, \{c, d\}\}$ is the same as the solution $\{\{c, d\}, \{a, b\}, \{x, y, z\}\}$ and solution $\{\{z, y, x\}, \{b, a\}, \{d, c\}\}$ as well as three others. Unfortunately, CIP allows the computer to consider each of these solutions as different, and unique, solutions.

In an attempt to reduce the time required to solve the CIP, three distinct symmetry cuts were developed. The symmetry cuts try to eliminate some or all of the $k!$ equivalent solutions to the IP.

The first symmetry constraint considered is to force node 1 to always be in cluster 1. Similarly, node 2 must always be in cluster 1 or 2. Continuing this line of reasoning the j^{th} node must be contained in the 1^{st}, 2^{nd}, ..., j^{th} cluster. This constraint can be expressed as:

$$x_{i1} + x_{i2} + \ldots + x_{ii} = 1 \qquad \forall\ i \leq k \text{ and } i \in N.$$

Another symmetry cut applied ensures that cluster 1 is filled before cluster 2 and similarly cluster 2 is filled before cluster 3. This ensures that the "full" clusters occur first in the solution. Thus, $full_k \geq full_{k+1}$.

$$full_k - full_{k+1} \geq 0$$

The final symmetry cut is the sum of coefficients in cluster k must be strictly greater than the sum of the coefficients cluster in k+1. These coefficients make each cluster different depending on the nodes that are included in that cluster. By arranging the clusters based on the coefficients, it helps ensure that each cluster is different. The basic idea is that a cluster with $\{v_1, v_4, v_7\}$ would be assigned a cluster with a lower number than the cluster $\{v_5, v_8\}$, since 1+4+7 < 5+8. Formally, this constraint can be written as follows.

$$\sum_{i \epsilon N} ix_{i,k} \leq \sum_{i \epsilon N} ix_{i,k+1} \qquad \forall\ k \epsilon \{1, 2, \ldots, k\text{-}1\}.$$

A small computational study was performed on these symmetry cuts. Surprisingly, none of the symmetry constraints were effective in reducing the run time of the CIP. However, these symmetry cuts are not the major achievement of this thesis and since they appear to not be useful for solving CIP, the computational study is not reported here. The next section describes how CIP is used to develop a heuristic that can generate excellent clusters for reasonably sized networks.

4.3 CHIP

Since the CIP cannot solve the clustering problem even on relatively small graphs, this research developed a heuristic. This heuristic uses the CIP as a neighborhood and thus this heuristic can be considered a large neighborhood heuristic. The benefits of large neighborhood heuristics are emphasized in Ahuja, et al. [1993].

The general idea of the Clustering Heuristic Integer Program (CHIP) is to begin with any feasible clustering solution. From any feasible solution, select a set of nodes V' and force the remaining nodes $N \setminus V'$ to remain in their clusters. Solve the CIP with this fixed restriction and let the new solution serve as the current solution. Then repeat the process, select a new set of nodes to remove from the solution, as many times as desired. As with most heuristics, if a better solution is encountered this best solution is recorded. The heuristic terminates by reporting the best solution. Formally,

Clustering Heuristic Integer Program (CHIP)

Set $z_{old} = \infty$

Initialize with a feasible solution: all nodes in cluster C_1

Determine solution value, z_{new}

While $z_{new} < z_{old}$

Set $z_{old} = z_{new}$

Randomly choose p nodes given by $p = \{v_1, \ldots, v_p\}$

Solve CIP with the additional restriction that every node in $V \setminus P$ remains in its current

cluster.

Set z_{new} to the current objective value and update the solution.

End While

Report the current clustering solution and corresponding z value.

In the next section CHIP is shown to be computationally superior to *K*-means. CHIP is a hill climbing heuristic and thus these impressive computational results are believed to be primarily due CHIP's ability to quickly optimize over a large number of neighboring solutions.

CHIP is a hill climbing heuristic. CHIP moves from z_{old} to z_{new} and only accepts an improvement in the solution value. Observe that at each iteration, $z_{old} \geq z_{new}$, since the existing solution is also a candidate solution for CIP and thus the objective value can be no worse than the existing solution. Consequently, each iteration of CHIP improves the existing objective value or it remains the same and thus it is a hill climbing heuristic.

In some hill climbing heuristics, the direction of steepest ascent is desired. In this case, the nodes are chosen randomly and so no attempt is made to find a direction of most improvement or even a direction of improvement. Thus, it might be possible to find better solutions by seeking sets of nodes that appear to generate poor solutions. This topic is left as future research.

CHIP is a large neighborhood search method since the randomly chosen p nodes can move from the current solution clusters to any of the other clusters, forming a new solution. Extending the argument from Chapter 2 regarding the size of the neighborhood leads to the following analysis.

If p out of n nodes are allowed to change clusters and there are currently k clusters, then the number of neighboring solutions is about $\binom{n}{p} k^p$ different solutions as long as there are at most k clusters. This can be seen by choosing any p of n nodes, $\binom{n}{p}$, to remove from the current solution. These p nodes can then be placed in any of the k clusters, k^p. Obviously some of these

p nodes could create new clusters and then the size of the neighborhood increases and it leads to a factor of $(k+p)^p$.

In the smallest cases of the computational studies, n = 50 and p = 5 and the smallest number of optimal clusters averaged about 4. In these smallest neighborhood sizes, there were still over 2 billion neighboring solutions in a single solutions neighborhood. In the larger cases n = 75 and k = 40, the neighborhood size is greater than the United State's national debt and approximately 2 quadrillion ($2*10^{15}$) neighboring solutions. Thus, CHIP is considered a large neighborhood search heuristic.

4.4 Computational Results

Three different classes of graphs, random, geographic random, and random geographic random, are used in conducting the computational comparison between CHIP and *K*-means. *K*-means is explained in further detail in Section 2.3.3 of this thesis. For each graph class, sixty instances are generated, 30 with 50 nodes and 30 with 75 nodes, with α values of 1, 10, and 100. The average solutions for both and run times for CHIP are presented below. The run time for *K*-means was less than a second for all 30 repetitions. All computational results are run on a Pentium Intel Core 2 Duo computer with a 2.2 GHz processor and 2.0 GB of RAM.

In an effort to be more equitable with *K*-means, no edge weights were assigned values. Rather the probability that a disease is transmitted from one individual to another is always 1 or 0. In studies where this probability is allowed to change, CHIP's solutions were nearly 10 times better than *K*-means. The primary reason for this dramatic difference is that *K*-means has no method to account for edges that are only partially weighted.

The first graph type is a random graph. In a random graph, an edge exists between two nodes with some probability p. Thus, there are no restrictions on where the nodes are positioned and no restriction on the number of edges incident to a node. For this study p is set to .1.

The second graph class is a random geographic graph. In a random geographic graph all nodes are randomly located in the zero-one square. An edge exists between two nodes if the distance between the two nodes is less than d. For this study d is set to .5.

The final graph class is a random geographic random graph, which is an incorporation of the previous two graph classes. Thus, a random geographic graph is created and an edge exists if

the two nodes are within a distance *d* of each other with probability *p*. For this study *d* is set to .5 and *p* is set to .1.

In order to run CHIP, certain parameters need to be input. The first input is the number of nodes in the graph and the maximum number of clusters. For this research, the maximum number of clusters was always set to be the same as the number of nodes. The user always has the option of adjusting the α value to encourage different numbers of clusters. In the first iteration of CHIP, all nodes are set to be in their own cluster and all subsequent iterations re-cluster the *p* random nodes.

Also important to note here is that if CHIP does not improve its *z*-value through three iterations, the *p* increases from five to eight for the next iteration. The value of *p* increases again to 10 if the *z*-value does not improve after another two iterations. Only when the *z*-value does not increase after performing two iterations of re-clustering 10 nodes does CHIP terminate and report the final solution value.

As with CHIP, *K*-means needs input before it can run. The number of nodes and the maximum number of clusters are the primary inputs needed. Since *K*-means normally requires the number of clusters to be input from the user, this research took the average number of clusters in CHIP and rounded up.

The average data from using the two different clustering methods can be seen by examining a single line in Tables 4.2, 4.3 and 4.4. The three different tables are divided by the three different graph types; random, geographic random, and random geographic random. The tables are organized by the different α values and number of nodes, *n*, along the left hand side and CHIP and *K*-means across the top. Each cell in the table represents the average of 30 iterations for that particular combination of graph type, α value, *n*, and clustering method.

The comparison of CHIP and *K*-means employs the solution values by taking the difference of the two and dividing by the average *K*-means solution value. Strictly speaking in average solution values, the equation is in this form: (*K*-means – CHIP)/ *K*-means.

Random Graphs		*K*-means			CHIP				
α	*n*	Solution	Clusters	Iterations	Solution	Clusters	Iterations	Run Time/ graph*	Comparison (z value)
1	50	159.70	19.93	3.90	64.53	31.17	66.10	35.30	59.59%
10	50	348.50	11.10	5.37	297.53	13.23	42.17	95.47	14.62%
100	50	965.67	4.77	5.80	897.30	4.00	38.13	64.10	7.08%
50 Average	**-**	**491.29**	**11.93**	**5.02**	**419.79**	**16.13**	**48.80**	**64.96**	**27.10%**
1	75	254.77	28.33	4.80	109.47	44.43	49.30	263.63	57.03%
10	75	518.50	29.53	4.70	382.57	19.43	57.57	376.73	26.22%
100	75	1501.83	5.53	7.13	1370.10	5.60	55.23	290.97	8.77%
75 Average	**-**	**758.37**	**21.13**	**5.54**	**620.71**	**23.16**	**54.03**	**310.44**	**30.67%**
Total Average	**-**	**624.83**	**16.53**	**5.28**	**520.25**	**19.64**	**51.42**	**187.70**	**28.89%**

Table 4.2 Comparison of CHIP and *K*-means on random graphs
***** *K*-means ran in less than one second, so the run times are not reported

CHIP's improvement over *K*-means on random graphs can be seen in the comparison values at the far right in Table 4.2. On the 50 node random graphs, the average solution values from CHIP are 27.10 percent better than the average solution values from *K*-means. CHIP performed three and a half percent better, at 30.67 percent, improvement over *K*-means on the 75 node random graphs. The total average across all six comparison values is 28.89 percent.

Geo-Random		*K*-means			CHIP				
α	*n*	Solution	Clusters	Iterations	Solution	Clusters	Iterations	Run Time/ graph*	Comparison (z value)
1	50	159.70	19.93	3.90	64.53	31.17	66.10	35.30	59.59%
10	50	252.67	8.63	5.60	218.03	9.30	39.83	70.23	13.71%
100	50	727.20	4.00	5.53	693.67	3.03	40.23	63.03	4.61%
50 Average	**-**	**379.86**	**10.86**	**5.01**	**325.41**	**14.50**	**48.72**	**56.19**	**25.97%**
1	75	168.30	25.30	5.17	100.20	37.83	47.73	245.03	40.46%
10	75	390.83	12.13	6.70	321.90	14.13	56.30	313.20	17.64%
100	75	1080.17	4.80	6.37	1055.37	4.30	62.77	316.93	2.30%
75 Average	**-**	**546.43**	**14.08**	**6.08**	**492.49**	**18.76**	**55.60**	**291.72**	**20.13%**
Total Average	-	**463.14**	**12.47**	**5.54**	**408.95**	**16.63**	**52.16**	**173.96**	**23.05%**

Table 4.3 Comparison of CHIP and *K*-means on geographic random graphs
* *K*-means ran in less than one second, so the run times are not reported

The improvement of CHIP over *K*-means on geographic random graphs can be seen in the comparison values at the far right in Table 4.3. On the 50 node geographic random graphs, the average solution values from CHIP are 25.97 percent better than the average solution values from *K*-means. On the 75 node geographic random graphs, CHIP performed 20.13 percent better than the *K*-means solution values. The total average across all six comparison values is 23.05 percent.

Rand-Geo-Rand		*K*-means			CHIP				
α	*n*	Solution	Clusters	Iterations	Solution	Clusters	Iterations	Run Time/ graph*	Comparison (z value)
1	50	148.17	18.37	4.30	74.67	25.97	33.27	33.27	49.61%
10	50	333.93	9.73	5.03	249.03	11.60	79.30	42.27	25.42%
100	50	924.37	3.93	5.53	854.13	3.90	47.43	47.43	7.60%
50 Average	**-**	**468.82**	**10.68**	**4.96**	**392.61**	**13.82**	**53.33**	**40.99**	**27.54%**
1	75	234.77	26.83	4.70	111.40	38.87	47.73	242.57	52.55%
10	75	504.93	14.77	6.33	371.90	16.97	63.50	362.57	26.35%
100	75	1426.57	4.90	7.10	1311.40	5.00	63.93	332.93	8.07%
75 Average	**-**	**722.09**	**15.50**	**6.04**	**598.23**	**20.28**	**58.39**	**312.69**	**28.99%**
Total Average	**-**	**595.46**	**13.09**	**5.50**	**495.42**	**17.05**	**55.86**	**176.84**	**28.27%**

Table 4.4 Comparison of CHIP and *K*-means on random geographic random graphs
* *K*-means ran in less than one second, so the run times are not reported

CHIP's improvement over *K*-means on random geographic random graphs can be seen in the comparison values at the far right in Table 4.4. On the 50 node random geographic random graphs, the average solution values from CHIP are 27.54 percent better than the average solution values from *K*-means. On the 75 node random geographic random graphs, CHIP performed 28.99 percent better than *K*-means. The total average across all six comparison values is 28.27 percent.

Visible in Table 4.5, the overall average improvement by CHIP over *K*-means is 25.57 percent. The largest discrepancy in CHIP's improvement is when α is adjusted from one to ten to one hundred. With α set to one, the average improvement was nearly 50 percent, showing that without the user indicating that there needs to be fewer clusters, CHIP greatly improves on *K*-means. The other averages apparent in Table 4.5 are that with α set to 1, CHIP performed nearly 50 percent better than *K*-means. Also notice that with α set to 10, CHIP performed over 20 percent better. Apparent here is that as α increases, the improvement over *K*-means decreases by more than half. Overall, CHIP results in better clustering solutions than *K*-means, whether α is set to one or to one hundred.

Alpha	50 Nodes	75 Nodes	
1	49.30%	50.01%	49.66%
10	17.92%	23.40%	20.66%
100	6.43%	6.38%	6.40%
	24.55%	26.60%	**25.57%**

Table 4.5 CHIP's average improvement over *K*-means.

The largest disadvantage to CHIP is that it uses too much memory when running in CPLEX, and because of that drawback it cannot run on graphs larger than 75 nodes. This disadvantage stopped this research from performing CHIP on the Clay Center example described in Chapter 3. Despite the fact that CHIP cannot process a full contact network of 4,600 nodes, if it was known that only 50 to 75 people had come into contact with one infectious person, CHIP is able to determine the proper quarantine area.

Although the CIP is not functional on large graphs, it led to the development of CHIP, which demonstrates dramatically better clustering solutions than *K*-means. This research could possibly lead to better clustering techniques across many different disciplines. In the future, data mining and marketing research, among others, may benefit from using CHIP to find better clusters.

CHAPTER 5 - Conclusion and Future Research

The original objective of this research was to develop a quarantine methodology based on a clustering algorithm. The primary focus of the study was on rural areas in Kansas due to the unique circumstances involved if an outbreak occurred in a rural area. In order to accurately determine a quarantine plan, it was necessary to develop a simulation that modeled the spread of a disease.

The resulting simulation core is highly adaptable to many diseases and contact networks. With new threats of natural disease such as the swine flu epidemic that began mid April in North America [86], having the ability to accurately simulate a disease and war-game different mitigation techniques could prove extremely valuable in saving numerous lives. The different parameters involved with the simulation core can be adjusted quickly by Dr. Easton's research team in conjunction with the EPICENTER at Kansas State University. Given the proper information concerning the contact network and the disease to be modeled, the simulation core could lead to modeling the spread of a disease through animal populations as well.

In order to develop ideal quarantine plans prior to an epidemic outbreak, this research moved forward on the idea that quarantines could be developed using a clustering method and contact networks for the individuals modeled. From this concept, the research created a clustering integer program (CIP). The CIP minimizes the objective function that groups like nodes and has a penalty for the number of clusters. The CIP was then tested on a number of graphs and moderately sized graphs (30 nodes) could not be solved.

However, CIP led to a large neighborhood, hill climbing heuristic, CHIP. CHIP has shown that it achieves approximately 25 percent better solutions than *K*-means, the most commonly used clustering method. CHIP's improvement on the common clustering methods has many implications across a broad number of disciplines including data mining and market research. Thus, CHIP not only provides valuable clustering solutions to be used in quarantine plans, but the research is extendable across many other areas of research.

5.1 Recommendations for Future Research

Due to the amount of on-going research in epidemics and clustering, there are numerous future research topics based on similar ideas to those presented in this thesis. One area of future research topics extends the simulation core and the improvements made on it. The other major topic of future research focuses on new advances to the CIP and CHIP.

One recommendation for future research is to adjust the simulation core for specificity and test the heuristic against an historical case of an epidemic outbreak. By testing the simulation core against an historical epidemic outbreak, researchers could possibly improve the simulation core to ensure that it accurately represents the spread of a disease and validate the contact network.

Another recommendation is to enhance the simulation core and extend the epidemic simulation to model people's movements across the entire state of Kansas. This would make it more diverse and expansive, leading to better results across large areas and diverse populations. In this manner, CDC's Regional Quarantine Station that covers Kansas could determine where the trouble spots might occur and where to stop possible pandemics before they expand too far.

Another topic for future research is to improve the Clustering Integer Program, so it solves larger, more realistic contact networks. Ideally, the CIP would be able to solve extremely large graphs with a minimum of a few hundred thousand nodes. The optimal solutions from the CIP could then make vast improvements in the clustering techniques currently used.

This research was not able to study graphs larger than 75 nodes due to the fact that on larger graphs CHIP could only perform a handful of iterations before CPLEX started reporting an error due to the amount of memory used. Therefore, another future research topic is to improve CHIP to require less memory when running on CPLEX so that it is able to run multiple iterations on larger contact networks.

Currently CHIP is a combination of the "random walk" idea and a hill climbing heuristic. The final recommendation for future research concerning CHIP is to ensure that the p nodes re-clustered in each iteration are the nodes that inflict the large detriments to the current clustering solution. In this manner, CHIP would be a faster hill climbing heuristic in that it would always go in a direction that improves the solution the most. By finding the p nodes that are re-solved in

each iteration and putting them in clusters that make the most improvement on the solution value, CHIP would be much less of a random walk and more a straight walk to a better solution.

As former President George W. Bush stated in his address at the National Defense University in 2004, “The greatest threat before humanity today is the possibility of a secret and sudden attack with chemical, or biological, or nuclear weapons.” With the possible threat of a bioterrorist attack or the unchecked outbreak of an endemic, having mitigation factors in place could greatly reduce the loss of life. The work presented here along with the future research topics could save a number of lives and increase humanity’s survivability.

Bibliography

[1] K. Romer and F. Mattern, "The design space of wireless sensor networks," *IEEE Wireless Communications*, vol. 11, Dec. 2004, pp. 54-61.

[2] M. Younis, M. Youssef, and K. Arisha, "Energy-aware management for cluster-based sensor networks," *Computer Networks*, vol. 43, Dec. 2003, pp. 649-68.

[3] T. Wu and S. Biswas, "A self-reorganizing slot allocation protocol for multi-cluster sensor networks," *4th International Symposium on Information Processing in Sensor Networks, IPSN 2005, April 25,2005 - April 27,2005*, Los Angeles, CA, United states: Institute of Electrical and Electronics Engineers Computer Society, 2005, pp. 309-316.

[4] K. Dasgupta, K. Kalpakis, and P. Namjoshi, "An efficient clustering-based heuristic for data gathering and aggregation in sensor networks," *WCNC 2003 - IEEE Wireless Communications and Networking Conference, 16-20 March 2003*, Piscataway, NJ, USA: IEEE, 2003, pp. 1948-53.

[5] M. Demirbas, A. Arora, V. Mittal, and V. Kulathumani, "Design and analysis of a fast local clustering service for wireless sensor networks," *Proceedings. First International Conference on Broadband Networks, 25-29 Oct. 2004*, Los Alamitos, CA, USA: IEEE Comput. Soc, 2004, pp. 700-9.

[6] A. Abbasi and M. Younis, "A survey on clustering algorithms for wireless sensor networks," *Computer Communications*, vol. 30, Oct. 2007, pp. 2826-41.

[7] S. Abidi and J. Ong, "A data mining strategy for inductive data clustering: a synergy between self-organising neural networks and K-means clustering techniques," *2000 TENCON Proceedings. Intelligent Systems and Technologies for the New Millennium, 24-27 Sept. 2000*, Piscataway, NJ, USA: IEEE, 2000, pp. 568-73.

[8] R. Wan, X. Yan, and X. Su, "A weighted fuzzy clustering algorithm for data stream," *ISECS International Colloquium on Computing, Communication, Control, and Management, CCCM 2008, August 03,2008 - August 04,2008*, Guangzhou, China: Inst. of Elec. and Elec. Eng. Computer Society, 2008, pp. 360-364.

[9] A. Mislove, M. Marcon, K.P. Gummadi, P. Druschel, and B. Bhattacharjee, "Measurement and analysis of online social networks," *IMC'07: 2007 7th ACM SIGCOMM Internet Measurement Conference, October 24,2007 - October 26,2007*, San Diego, CA, United states: Association for Computing Machinery, 2007, pp. 29-42.

[10] S. Ahnert and T. Fink, "Clustering signatures classify directed networks," *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, vol. 78, 2008.

[11] R. Kuo, K. Chang, and S. Chien, "Integration of self-organizing feature maps and genetic-algorithm-based clustering method for market segmentation," *Journal of Organizational Computing and Electronic Commerce*, vol. 14, 2004, pp. 43-60.

[12] K. Kim and H. Ahn, "A recommender system using GA K-means clustering in an online shopping market," *Expert Systems with Applications*, vol. 34, 2008, pp. 1200-1209.

[13] P. Ziegler, *The Black Death*, London: Collins, 1969.

[14] D. Barenblatt, *A Plague Upon Humanity: The Secret Genocide of Axis Japan's Germ Warfare Operation*, New York: HarperCollins, 2004.

[15] R. Diestel, *Graph Theory*, Springer, 2006.

[16] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin, *Network Flows: Theory, Algorithms, and Applications*, Prentice Hall, 1993.

[17] M.A. Schroeder and K.T. Newport, "Augmenting tactical communications networks to enhance survivability," *Proceedings - IEEE Military Communications Conference*, vol. 2, 1988, pp. 507-513.

[18] M.A. Schroeder and K.T. Newport, "TACTICAL NETWORK SURVIVABILITY THROUGH CONNECTIVITY OPTIMIZATION.," *MILCOM 87: 1987 IEEE Military Communications Conference - Conference Record. 'Crisis Communications: The Promise and Reality'.*, Washington, DC, USA: IEEE, 1987, pp. 590-597.

[19] D. Gossink, J. Scholz, and L. Zhang, "A comparison of techniques for optimisation of C3I processes realised on resource-limited networks," *Proceeding of Information Decision and Control, 8-10 Feb. 1999*, Piscataway, NJ, USA: IEEE, 1999, pp. 29-34.

[20] A. Sztykgold, G. Coppin, and O. Hudry, "Dynamic optimization of the strength ratio during a terrestrial conflict," *2007 IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning, ADPRL 2007, April 01,2007 - April 05,2007*, Honolulu, HI, United states: Inst. of Elec. and Elec. Eng. Computer Society, 2007, pp. 241-246.

[21] Hsinchun Chen, "Homeland security data mining using social network analysis," *2008 IEEE International Conference on Intelligence and Security Informatics (ISI 2008), 17-20 June 2008*, Piscataway, NJ, USA: IEEE, 2008, p. xxvii.

[22] J. Liebowitz, "Linking social network analysis with the analytic hierarchy process for knowledge mapping in organizations," *Journal of Knowledge Management*, vol. 9, 2005, pp. 76-86.

[23] E. Kilic and P. Dundar, "The edge-accessibility number via graph, operations and an algorithm," *Neural Network World*, vol. 17, 2007, pp. 213-223.

[24] M. Chan, D.Z. Chen, F.Y.L. Chin, and C.A. Wang, "Construction of the nearest neighbor embracing graph of a point set," *Journal of Combinatorial Optimization*, vol. 11, 2006, pp. 435-443.

[25] N. Harvey, R. Ladner, L. Lovasz, and T. Tamir, "Semi-matchings for bipartite graphs and load balancing," *Journal of Algorithms*, vol. 59, Apr. 2006, pp. 53-78.

[26] M. Karaata and K. Saleh, "A distributed self-stabilizing algorithm for finding maximum matching," *Computer Systems Science and Engineering*, vol. 15, May. 2000, pp. 175-80.

[27] Y. Chen, C. Wu, and M. Yao, "Dynamic topology construction for road network," *4th International Conference on Networked Computing and Advanced Information Management, NCM 2008, Sep 2-4 2008*, Piscataway, NJ 08855-1331, United States: Institute of Electrical and Electronics Engineers Computer Society, 2008, pp. 359-364.

[28] S. Lew and K. Tan, "Converting undirected graphs to feasible digraphs of one-way roads," *16th IASTED International Conference on Modelling and Simulation, May 18-20 2005*, Anaheim, CA, United States: Acta Press, 2005, pp. 209-214.

[29] J. Bala, P.W. Pachowicz, and H. Vafaie, "Rapid SAR target modeling through genetic inheritance mechanism," *Algorithms for Synthetic Aperture Radar Imagery IV, April 23,1997 - April 24,1997*, Orlando, FL, USA: Society of Photo-Optical Instrumentation Engineers, 1997, pp. 137-148.

[30] A. Hayrapetyan, D. Kempe, M. Pal, and Z. Svitkina, "Unbalanced graph cuts," *13th Annual European Symposium on Algorithms, ESA 2005, October 03,2005 - October 06,2005*, Palma de Mallorca, Spain: Springer Verlag, 2005, pp. 191-202.

[31] N. Peyrard and A. Franc, "Cluster variation approximations for a contact process living on a graph," *Physica A: Statistical Mechanics and its Applications*, vol. 358, 2005, pp. 575-592.

[32] P. Schumm, C. Scoglio, D. Gruenbacher, and T. Easton, "Epidemic spreading on weighted contact networks," *2nd International Conference on Bio-Inspired Models of Network, Information, and Computing Systems, BIONETICS 2007, December 10,2007 - December 12,2007*, Budapest, Hungary: Inst. of Elec. and Elec. Eng. Computer Society, 2007, pp. 201-208.

[33] N. Fefferman and K. Ng, "How disease models in static networks can fail to approximate disease in dynamic networks," *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, vol. 76, 2007.

[34] T. Britton, S. Janson, and A. Martin-Lof, "Graphs with specified degree distributions, simple epidemics, and local vaccination strategies," *Advances in Applied Probability*, vol. 39, 2007, pp. 922-948.

[35] Y. Chen, T. Wang, and D. Wong, "A graph partitioning problem for multiple-chip design," *1993 IEEE International Symposium on Circuits and Systems, 3-6 May 1993*, New York, NY, USA: IEEE, 1993, pp. 1778-81.

[36] S. Das, A. Hossain, S. Biswas, E. Petriu, M. Assaf, W. Jone, and M. Sahinoglu, "On a new graph theory approach to designing zero-aliasing space compressors for built-in self-testing," *IEEE Transactions on Instrumentation and Measurement*, vol. 57, Oct. 2008, pp. 2146-68.

[37] J. Beck, P. Prosser, and E. Selensky, "Graph transformations for the vehicle routing and job shop scheduling problems," *Graph Transformation. First International Conference, ICGT 2002. Proceedings, 7-12 Oct. 2002*, Berlin, Germany: Springer-Verlag, 2002, pp. 60-74.

[38] J. Beasley and N. Christofides, "Vehicle routing with a sparse feasibility graph," *European Journal of Operational Research*, vol. 98, May. 1997, pp. 499-511.

[39] P. McDonald, D. Geraghty, I. Humphreys, and S. Farrell, "Assessing environmental impact of transport noise with wireless sensor networks," *Transportation Research Record*, 2008, pp. 133-139.

[40] Y. Oshino, K. Tsukui, H. Hanabusa, and M. Kuwahara, "Investigation into the noise map based on traffic flow prediction in the citywide road network," *Acta Acustica United With Acustica*, vol. 92, May. 2006, pp. 100-1.

[41] A. Iyer and Jianming Ye, "A network based model of a promotion-sensitive grocery logistics system," *Networks*, vol. 38, Dec. 2001, pp. 169-80.

[42] A. Amberg and S. Voss, "A hierarchical relaxations lower bound for the capacitated arc routing problem," *Proceedings of the 35th Annual Hawaii International Conference on System Sciences, 7-10 Jan. 2002*, Los Alamitos, CA, USA: IEEE Comput. Soc, 2002, pp. 1415-24.

[43] D. Baum and I. Kovalenko, "Graph models for communication of mobile users in access areas," *Cybernetics and Systems Analysis*, vol. 39, Oct. 2003, pp. 716-27.

[44] S. Nawaz and S. Jha, "A graph drawing approach to sensor network localization," *2007 IEEE Internatonal Conference on Mobile Adhoc and Sensor Systems, MASS, October 08,2007 - October 11,2007*, Pisa, Italy: Inst. of Elec. and Elec. Eng. Computer Society, 2007.

[45] E. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, 1959, pp. 269–271.

[46] R.W. Floyd, "Algorithm 97: shortest path," *Communications of the ACM*, vol. 5, 1962, p. 345.

[47] A.V. Karzanov, "Determining the maximal flow in a network by the method of preflows," *Soviet Mathematics Doklady*, 1974, pp. 434-437.

[48] E.A. Dinic, "Algorithm for Solution of a Problem of Maximum Flow in Networks with Power Estimation." *Soviet Mathematics Doklady*, 1970, pp. 1277-1280.

[49] J. Edmonds and R.M. Karp, "THEORETICAL IMPROVEMENTS IN ALGORITHMIC EFFICIENCY FOR NETWORK FLOW PROBLEMS.," *Journal of the Association for Computing Machinery*, vol. 19, 1972, pp. 248-264.

[50] J.E. Hopcraft and R.M. Karp, "An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs," *SIAM J. Comput*, vol. 2, 1973, pp. 225-231.

[51] J.B. Kruskal Jr, "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proceedings of the American Mathematical Society*, 1956, pp. 48-50.

[52] R.C. Prim, "Shortest connection networks and some generalizations," *Bell System Technical Journal*, vol. 36, 1957, pp. 1389-1401.

[53] Y. Chen and H. Hu, "An overlapping cluster algorithm to provide non-exhaustive clustering," *European Journal of Operational Research*, vol. 173, 2006, pp. 762-780.

[54] Z. Sun, "A cluster algorithm identifying the clustering structure," *International Conference on Computer Science and Software Engineering, CSSE 2008, December 12,2008 - December 14,2008*, Wuhan, Hubei, China: Inst. of Elec. and Elec. Eng. Computer Society, 2008, pp. 288-291.

[55] Q. Ren, J. Tian, Y. He, and J. Cheng, "Automatic fingerprint identification using cluster algorithm," *Proceedings - International Conference on Pattern Recognition*, vol. 16, 2002, pp. 398-401.

[56] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, "On clustering validation techniques," *Journal of Intelligent Information Systems*, vol. 17, 2001, pp. 107-145.

[57] N. Bolshakova and F. Azuaje, "Cluster validation techniques for genome expression data," *Signal Processing*, vol. 83, 2003, pp. 825-833.

[58] D. Jiang, C. Tang, and A. Zhang, "Cluster analysis for gene expression data: A survey,"

IEEE Transactions on Knowledge and Data Engineering, vol. 16, 2004, pp. 1370-1386.

[59] D. Kayarat, "The Quadratic Polyhedral Clustering Algorithm: A New Method to Cluster Microarray Data," 2005, p. 66.

[60] Jiahai Wang and Yalan Zhou, "Stochastic optimal competitive Hopfield network for partitional clustering," *Expert Systems with Applications*, vol. 36, Mar. 2009, pp. 2072-80.

[61] Y. Naija, K. Blibech, S. Chakhar, and R. Robbana, "Extension of partitional clustering methods for handling mixed data," *IEEE International Conference on Data Mining Workshops, ICDM Workshops 2008, December 15,2008 - December 19,2008*, Pisa, Italy: Inst. of Elec. and Elec. Eng. Computer Society, 2008, pp. 257-266.

[62] H. Steinhaus, "Sur la division des corp materiels en parties," *Bull. Acad. Polon. Sci*, vol. 1, pp. 801–804.

[63] P. Brucker, "On the complexity of clustering problems," *Optimization and operations research*, vol. 157, 1978.

[64] I.H. Osman and J.P. Kelly, *Meta-Heuristics: Theory and Applications*, Springer, 1996.

[65] E. Aarts and J.K. Lenstra, *Local Search in Combinatorial Optimization*, Princeton University Press, 1997.

[66] D. Goehring, "Time constrained planning using simulated annealing," *Proceedings. The First International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems IEA/AIE - 88, 1-3 June 1988*, New York, NY, USA: ACM, 1988, pp. 1066-70.

[67] Y. Kim and M. Kim, "Stepwise-overlapped parallel annealing algorithm," *Electronics Letters*, vol. 25, 1989, pp. 1094-6.

[68] W. Gao, "Study on immunized ant colony optimization," *3rd International Conference on Natural Computation, ICNC 2007, August 24,2007 - August 27,2007*, Haikou, Hainan, China: Inst. of Elec. and Elec. Eng. Computer Society, 2007, pp. 792-796.

[69] R. Jovanovic, M. Tuba, and D. Simian, "An object-oriented framework with corresponding graphical user interface for developing ant colony optimization based algorithms," *WSEAS Transactions on Computers*, vol. 7, 2008, pp. 1948-1957.

[70] Y. Zhou, J. Wang, and J. Yin, "Genetic particle swarm optimization based on multiagent model for combinatorial optimization problem," *2008 IEEE International Conference on Networking, Sensing and Control, ICNSC, April 06,2008 - April 08,2008*, Sanya, China: Inst. of Elec. and Elec. Eng. Computer Society, 2008, pp. 293-297.

[71] Qingyuan He and Chuanjiu Han, "An improved particle swarm optimization algorithm with disturbance term," *Computational Intelligence and Bioinformatics. International Conference on Intelligent Computing, ICIC 2006. Proceedings, 16-19 Aug. 2006*, Berlin, Germany: Springer-Verlag, 2006, pp. 100-8.

[72] R. Anderson, *Population Dynamics of Infectious Diseases*, Chapman & Hall, 1982.

[73] F. Brauer, *Mathematical Models in Population Biology and Epidemiology*, New York: Springer, 2001.

[74] D. Iacoviello and G. Liuzzi, "Fixed/free final time SIR epidemic models with multiple controls," *International Journal of Simulation Modelling*, vol. 7, Jun. 2008, pp. 81-92.

[75] P. Neal, "Coupling of two SIR epidemic models with variable susceptibilities and infectivities," *Journal of Applied Probability*, vol. 44, Mar. 2007, pp. 41-57.

[76] Z. Zhao, L. Chen, and X. Song, "Impulsive vaccination of SEIR epidemic model with time delay and nonlinear incidence rate," *Mathematics and Computers in Simulation*, vol. 79, 2008, pp. 500-510.

[77] K. Zhang and Xiao-Qiang Zhao, "Spreading speed and travelling waves for a spatially discrete SIS epidemic model," *Nonlinearity*, vol. 21, Jan. 2008, pp. 97-112.

[78] L. Rvachev, "Simulation of large-scale epidemics on a digital computer," *Soviet Physics - Doklady*, vol. 13, Nov. 1968, pp. 384-6.

[79] P. Guimaraes, M. de Menezes, R. Baird, D. Lusseau, P. Guimaraes, and S. dos Reis, "Vulnerability of a killer whale social network to disease outbreaks," *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, vol. 76, Oct. 2007, pp. 042901-1.

[80] Chung-Yuan Huang, Ji-Lung Hsieh, Chuen-Tsai Sun, and Chia-Ying Cheng, "Teaching epidemic and public health policies through simulation," *WSEAS Transactions on Information Science and Applications*, vol. 3, May. 2006, pp. 899-904.

[81] C. Barrett, S. Eubank, and J. Smith, "If smallpox strikes Portland (simulation of the spread of disease in social networks)," *Scientific American (International Edition)*, vol. 292, Mar. 2005, pp. 54-61.

[82] "ILOG CPLEX: High-performance software for mathematical programming and optimization." Retrieved 28 April 2009 from http://www.ilog.com/products/cplex/

[83] G. Altarelli and F. Feruglio, "Tri-bimaximal neutrino mixing from discrete symmetry in extra dimensions," *Nuclear Physics B*, vol. 720, 2005, pp. 64-88.

[84] F. Focacci and M. Milano, "Global cut framework for removing symmetries," *Proceedings of 7th International Conference on Principles and Practice of Constraint Programming, 26 Nov.-1 Dec. 2001*, Berlin, Germany: Springer-Verlag, 2001, pp. 77-92.

[85] A. Gupta, V. Prasad, and L. Davis, "Extracting regions of symmetry," *2005 International Conference on Image Processing, 11-14 Sept. 2005*, Piscataway, NJ, USA: IEEE, 2006, pp. 133-6.

[86] "A/H1N1 influenza like human illness in Mexico and the USA: OIE statement." Retrieved 29 April 2009 from http://www.oie.int/eng/press/en_090427.htm