NETWORK MODEL FOR DISASTER MANAGEMENT

By

ASHWIN ARULSELVAN

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2009

To my family and friends

ACKNOWLEDGMENTS

I must thank my advisor Dr. Panos M. Pardalos for constant support and encouragement for the last five years of my graduate studies. He has been a good guide, friend and colloborator. He was responsible for my transformation from a student to a researcher. He helped me sustain my interests in graduate studies as a constant source of motivation.

I would like to express my gratitude my co-advisor Dr. Lily Elefteriadou. She has been a great support and gave me so much leverage in helping me finish the disseration. She has also been a good colloborator and great supervisor by being flexible and giving me independence in research.

I greatly appreciate the efforts of my committee members, Dr. Jonathan C. Smith, Dr. Joseph Geunes, Dr. My T. Thai for their time and constructive criticisms that helped in improving the quality of dissertation. I would also like to thank them for their valuable insights that provided directions to me.

I am greatly indebted to my parents and my brother for their immense moral support. This accomplishment would not have been possible without them.

I wish to thank to my friends: Altannar Chinchuluun, Bharath Badrinarayanan, Clayton Commander, Michael J. Hirsch, Steffen Rebennack, Sriram Sethuraman, Raagavendra Hareesh, Anand Srinivasan, Vera Tomaino, Balachandran Vaidyanathan, Ashish Nemani, Suchandan Guha, Manisha Goswami, Erhun Kundakcioglu, Petros Xanthopoulos, Mehmet Onal, Gubjrt Gylfadttir, Arni Jonsson, Ingrida Radziukyniene, Artyom Nahapetyan, Nikita Boyko, Sibel Sonuc, Andy Fan, Alex Grassas, Alexey Sorokin, Vladimir Boginski, Erika Short and Oleg Shylo. They have been my company in the last five years and they were present during the good and bad times. They were always proud of me and I am overwhelmed by their companionship.

Finally, I would like to thank all my colloborators: Vladimir Boginski, Clayton Commander, Michael J. Hirsch, Alla Kammaradiner, Steffen Rebennack, Oleg Shylo, Maruccio Resende, Petros Xanthopoulos, Qipeng Zheng and Pilar Mendoza. They taught me to be a good researcher and have been terrific working with them.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

NETWORK MODEL FOR DISASTER MANAGEMENT

By

Ashwin Arulselvan

Aug 2009

Chair: Dr. Panos M. Pardalos
Major: Department of Industrial & Systems Engineering

We propose network models to study applications related to risk control and disaster management. We also perform a detailed complexity analysis and provide solution approaches for these problems. More specifically, we examine three different problems in detail, study their complexity and provide techniques to solve them. The first problem studies the connectivity properties of a node deleted subgraph. The objective is to identify important nodes of a graph responsible for the connectivity of the graph. This problem finds applications in studying robustness of a network, identifying important individuals of a social network and critical nodes of a telecommunication or transportation network. We study the complexity of the problem and develop heuristic procedures to solve the problem. We then study path planning problems that arises in military applications. The problem involves in routing unmanned vehicles in a network in order to visit targets in the most efficient way. We look at the complexity of these problems and provide heuristic solutions to solve the problems. We then provide a comparison of the heuristic solution with the optimal solution. Finally, we examine the evacuation problem that arises in an emergency situation. We provide complexity analysis for the contraflow problem, where in we allow arc reversals. We then develop a branch and price mechanism for establishing optimal evacuation routes for a bimodal multicommodity flow problem. We provide computational results for planar and grid graphs.

CHAPTER 1
INTRODUCTION

The need for analysis of disaster management is necessary in many real world applications. Many studies in this area under different names such as network survivability, resilience studies, vulnerability analysis is already known. The management of disasters could either be proactive, where we prepare ourselves better for a disaster or a reactive, where we plan to cope up with the hazards caused by the disaster. In this dissertation, we present the some proactive strategies in chapters 2 and 3 and some reactive strategies in chapters 5 and 6. Chapter 4 deals with a survey of evacuation studies as the subsequent chapters 5 and 6 primarily deal with emergency management.

Identifying critical nodes in a graph is important to understand the structural characteristics and the connectivity properties of the network. In chapter 2, we focus on detecting critical nodes, or nodes whose deletion results in the minimum pair-wise connectivity among the remaining nodes. This problem, known as the CRITICAL NODE PROBLEM, has applications in several fields including transportation, biomedicine, telecommunications, and military strategic planning. We show that the decision version of the problem is NP-complete and derive a mathematical formulation based on integer linear programming. In addition, we propose a heuristic for the problem which is then enhanced by the application of a local improvement method. A computational study is presented in which we apply the integer programming formulation and the heuristic to real and randomly generated data sets. For all instances tested, the heuristic is able to efficiently provide optimal solutions in a fraction of the time required by a commercial software package. We also study a variation of this problem, which involves in minimizing the number of nodes deleted by constraining the size of the connected components in the node-deleted subgraph.

In chapter 3, we consider the problem of determining an optimal path for an unmanned aerial vehicle which needs to visit multiple targets. The objective is to minimize the travel distance while maximizing the utility of the visitation order. This is known as the TARGET VISITATION PROBLEM and has several applications including combat search and rescue, environmental assessment, and disaster relief. First, we provide a mathematical model based on integer linear programming and prove that the problem is NP-complete. Then we describe the implementation of a genetic algorithm for finding approximate solutions. The heuristic is then hybridized by the implementation of a local search procedure. Numerical results are presented demonstrating the effectiveness of the proposed procedure. We also consider the problem of maximizing the total connectivity for a set of wireless agents in a mobile ad hoc network. That is, given a set of wireless units each having a start point and a destination point, our goal is to determine a set of routes for the units which maximizes the overall connection time between them. Known as the COOPERATIVE COMMUNICATION PROBLEM IN MOBILE AD HOC NETWORKS (CCPM), this problem has several military applications including coordination of rescue groups, path planning for unmanned air vehicles, and geographical exploration and target recognition. The CCPM is NP-hard; therefore heuristic development has been the major focus of research. In this work, we survey the CCPM examining first some early combinatorial formulations and solution techniques. Then we introduce new continuous formulations and compare the results of several case studies. By removing the underlying graph structure, we are able to create a more realistic model of the problem as supported by the numerical evidence.

Over the years, investigators from various fields have studied the evacuation problem, which resulted in a multitude of evacuation models. In chapter 4, we consolidate this work and make an assay of the existing models available in the literature. The methodology or approaches employed in these models were broadly classified as either optimization or simulation methods and presented. Further, we analyze the features considered by either techniques by identifying the major factors influencing the evacuation efficiency and briefing the shortcomings in these techniques.

Chapter 5 provides a comprehensive study on network flow problems with arc reversal capabilities. The problem is to identify the arcs to be reversed in order to achieve a maximum flow from source(s) to sink(s). The problem finds its applications in emergency transportation management, where the lanes of a road network could be reversed to enable flow in the opposite direction. We present several network flow problems with the arc reversal capability and discuss their complexity in this chapter. More specifically, we discuss the polynomial time algorithms for the maximum dynamic flow problem with arc reversal capability having a single source and a single sink, and for the maximum (static) flow problem. The presented algorithms are based on graph transformations and reductions to polynomially solvable flow problems. In addition, we show that the quickest transshipment problem with arc reversal capability and the problem of minimizing the total cost resulting from arc switching costs are NP-hard.

Finally in chapter 6, we provide solutions to bimodal transportation problem in an emergency situation, which incorporates multicommodity flows. This problem is similar to a line planning problem. We have two modes of transportation, namely private cars and buses. We assume that the demands at every node for both cars and buses are known and the a set of feasible tours of buses has been established. We need to determine the efficient paths and optimal set of buses routes to satisfy the demand. We provide a path based formulation, which enables us to employ a branch and price approach to solve the problem. We discuss the assumptions, formulations, subproblems and branching strategies. The computational results for some grid and planar graphs are provided for the branch and price mechanism.

CHAPTER 2
CRITICAL NODE DETECTION PROBLEMS

2.1 Introduction

In this chapter, we study two variants of the CRITICAL NODE PROBLEM. In general, the objective of the CRITICAL NODE PROBLEM (CNP) is to find a set of k nodes in a graph whose deletion results in the maximum network fragmentation. By this we mean, maximize the number of components in the k-vertex deleted subgraph. Studies carried out in this line include those by Bavelas [22] and Freeman [78] which emphasize node centrality and prestige, both of which are usually functions of a nodes degree. However, they lacked applications to problems which emphasized network fragmentation and connectivity.

Given a graph and an integer k, the objective of the CRITICAL NODE PROBLEM (CNP) is to find a set of k nodes in the graph whose deletion results in the maximum network fragmentation. By this we mean, minimize the pair-wise connectivity between the nodes in the the k-vertex deleted subgraph. In the second version of the problem, we seek the minimum number of nodes whose deletion constrains the cardinality of connected component.

The CRITICAL NODE PROBLEMS has several applications in the field of social network analysis. Social networks have attracted a significant amount of attention in recent years. The study of these graphs is important to better understand several properties which are most common in network depictions of social interactions including cohesion, transitivity, and centrality of specific actors of the graph [20]. The study of the various properties of social networks such as diameter, radiality, and connectivity are responsible for social contagion and provide scope for containment of an epidemic outbreak. These properties also help in designing strategies for communication breakdowns in human and telecommunication networks [172].

The CNP finds applications in network immunization [54, 204] where mass vaccination is an expensive process and only a specific number of people, modeled as nodes of a graph, can be vaccinated. The immunized nodes cannot propagate the virus and the goal is to identify the individuals to be vaccinated in order to reduce the overall transmissibility of the virus. There are several vaccination strategies in the literature (see e.g., [54, 204]) offering control of epidemic outbreaks; however, none of the proposed are optimal strategies. The vaccination strategies suggest emphasizing the *centrality* of nodes as a major factor rather than *critical* nodes whose deletion will maximize the disconnectivity of the graph. Deletion of central nodes may not guarantee a fragmentation of the network or even disconnectivity, in which case disease transmission cannot be prevented. Because social networks model the patterns of humans, they very greatly over time. The relationships between people, represented by edges in the social network, are transient and there is a constant rewiring between the nodes as new relationships are established. The proposed critical node technique minimizes the transmission of the disease over an instance of the dynamic network.

Furthermore, the CNP can be applied to the study of covert terrorist networks, where a certain number of individuals have to be identified whose deletion will result in the maximum breakdown of communication between individuals in the network [123]. Likewise in order to stop the spreading of a virus over a telecommunication network, one can identify the critical nodes of the graph and take them offline. Similarly, if one's ultimate goal is to prevent communication on a wired telecommunication network, an efficient way of doing so would be to jam the critical nodes. This has been studied in the context of wireless networks by Commander et al. in [59].

Before proceeding, we mention one final area in which the CRITICAL NODE PROBLEM finds several applications, and that is in the field of transportation engineering [71]. Two particular examples are as follows. In general, for transportation networks, it is important to identify critical nodes in order to ensure they operate reliably for transporting people and goods throughout the network. Further, in planning for emergency evacuations, identifying the critical nodes of the transportation network is crucial. The reason is two-fold. First, knowledge of the critical nodes will help in planning the allocation of resources during the evacuation. Secondly, in the aftermath of a disaster they will help in re-establishing critical traffic routes.

Borgatti [27] has studied a similar problem, focusing on node detection resulting in maximum network disconnectivity. Other studies in the area of node detection such as centrality [22, 78] focus on the prominence and reachability to and from the central nodes. However, little emphasis is placed on the importance of their role in the network connectivity and diameter. Perhaps one reason for this is because all of the aforementioned references relied on simulation to conduct their studies. Although the simulations have been successful, a mathematical formulation is essential for providing insight and helping to reveal some of the fundamental properties of the problem [153]. In the next section, we present a mathematical model based on integer linear programming which provides optimal solutions for the CRITICAL NODE PROBLEM.

We organize this chapter by first formally defining the problem and discussing its computational complexity. Next, we provide an integer programming (IP) formulation for the corresponding optimization problem. In Section 2.2.4 we introduce a heuristic to quickly provide solutions for instances of the problem. We present a computational study in Section 2.2.5, in which we compare the performance of the heuristic against the optimal solutions which were computed using a commercial software package. We perform the same study on the cardinality constrained problem in section 2.3. Some concluding remarks are given in Section 2.4.

2.2 Critical Node Detection Problem

2.2.1 Problem Definition

The formal definition of the problem is given by:

CRITICAL NODE PROBLEM (CNP)

INPUT: An undirected graph $G = (V, E)$ and an integer k.

OUTPUT: $A = \arg\min \sum_{i,j \in (V \setminus A)} u_{ij}\big(G(V \setminus A)\big) : |A| \leq k$, where

$$u_{ij} \;:=\; \begin{cases} 1, & \text{if } i \text{ and } j \text{ are in the same component of } G(V \setminus A), \\ 0, & \text{otherwise.} \end{cases}$$

The objective is to find a subset $A \subseteq V$ of nodes such that $|A| \leq k$, whose deletion minimizes the pair-wise connectivity among the nodes in the induced subgraph $G(V \setminus A)$.

This problem is similar to MINIMUM k-VERTEX SHARING [151], where the objective is to minimize the number of nodes deleted to achieve a k-way partition. Here we are considering the complementary problem, where we know the number of vertices to be deleted and we try to maximize the number of components formed and implicitly limit the sizes of the components. Borgatti [27] has given a comprehensive illustration to facilitate the understanding of the objective function and its non-triviality.

2.2.2 Computational Complexity

We now prove that the recognition version of the CNP is NP-complete. Consider the recognition version of the CNP:

K-CRITICAL NODE PROBLEM (K-CNP)

INPUT: An undirected graph $G = (V, E)$ and an integer k.

QUESTION: Is there a set M, where M is the set of all maximal connected components of G obtained by deleting k nodes or less, such that $\sum_{\forall\ i \in M} \frac{\sigma_i(\sigma_i - 1)}{2} \leq K$, where σ_i is the cardinality of component i, for each $i \in M$?

In order to prove that the K-CNP is NP-complete, we make use of the following lemmata. In particular, we prove that optimizing the objective function not only maximizes the pair-wise disconnectivity among the nodes, but also minimizes the variance in the cardinalities of the components. Particularly, in Lemma 2.1 we show that for any two solutions resulting in the same number of components, if the cardinalities of the components are equal in one solution, and not equal in the other, then the objective value of the latter will always be worse than that of the former.

Lemma 2.1. *Let M be a partition of $G = (V, E)$ in to L components obtained by deleting a set D of nodes, where $|D| = k$. Then the objective function $\sum_{\forall\ i \in M} \frac{\sigma_i(\sigma_i - 1)}{2} \geq \frac{(|V|-k)\left(\frac{|V|-k}{L} - 1\right)}{2}$, with equality holding if and only if $\sigma_i = \sigma_j, \forall\ i, j \in M$, where σ_i is the size of i^{th} component of M.*

Proof.

Case 1: $\boldsymbol{\sigma_i \neq \sigma_j,\ \forall\ i, j \in M}$.

Note that $\sum_{i \in M} \sigma_i = |V| - k$. Then given a solution for an instance of the CNP, we have that

$$\frac{1}{L}\sum_{i \in M}\Big(\sigma_i - \frac{1}{L}\sum_{i \in M}\sigma_i\Big)^2 = \frac{1}{L}\sum_{i \in M}\sigma_i^2 - \Big(\frac{1}{L}\sum_{i \in M}\sigma_i\Big)^2 \tag{2–1}$$

$$\geq 0 \tag{2–2}$$

$$= \left(\frac{|V|-k}{L}\right)^2 - \left(\frac{|V|-k}{L}\right)^2. \tag{2–3}$$

This implies that

$$\frac{1}{L}\sum_{i \in M}\sigma_i^2 \geq \left(\frac{|V|-k}{L}\right)^2. \tag{2–4}$$

Therefore,

$$\frac{1}{L}\sum_{i \in M}\sigma_i^2 - \frac{1}{L}\sum_{i \in M}\sigma_i \geq \left(\frac{|V|-k}{L}\right)^2 - \frac{1}{L}\sum_{i \in M}\sigma_i \tag{2–5}$$

$$= \left(\frac{|V|-k}{L}\right)^2 - \frac{|V|-k}{L}. \tag{2–6}$$

Thus, we have that

$$\sum_{i \in M}\frac{\sigma_i(\sigma_i - 1)}{2} \geq \frac{(|V|-k)\left(\frac{|V|-k}{L} - 1\right)}{2}. \tag{2–7}$$

Case 2: $\boldsymbol{\sigma_i = \sigma_j,\ \forall\ i, j \in M}$.

In this case, each component of M will be of size $\frac{|V|-k}{L}$, which is obviously the average size of a component of M. Thus

$$\sum_{\forall\ i\in M} \frac{\sigma_i(\sigma_i-1)}{2} = \frac{(|V|-k)\big(\frac{|V|-k}{L}-1\big)}{2}. \tag{2–8}$$

Conversely, if (2–8) holds, but each component of M is not the same size, it follows that not all components will be of average size and hence

$$\frac{1}{L}\sum_{i\in M}\sigma_i^2 - \Big(\frac{1}{L}\sum_{i\in M}\sigma_i\Big)^2 \;>\; 0 \tag{2–9}$$

$$= \left(\frac{|V|-k}{L}\right)^2 - \left(\frac{|V|-k}{L}\right)^2. \tag{2–10}$$

Similar to the above result, we see that

$$\frac{1}{L}\sum_{i\in M}\sigma_i^2 > \left(\frac{|V|-k}{L}\right)^2. \tag{2–11}$$

Thus,

$$\frac{1}{L}\sum_{i\in M}\sigma_i^2 - \frac{1}{L}\sum_{i\in M}\sigma_i \;>\; \left(\frac{|V|-k}{L}\right)^2 - \frac{|V|-k}{L}. \tag{2–12}$$

$$\Rightarrow \sum_{i\in M}\frac{\sigma_i(\sigma_i-1)}{2} \;>\; \frac{(|V|-k)\left(\frac{|V|-k}{L}-1\right)}{2}. \tag{2–13}$$

This is a contradiction and we have the proof. $\square$

The following lemma provides a similar result as above. However in this case, the number of components induced by each solution are not assumed to be equal.

Lemma 2.2. *Let M_1 and M_2 be two sets of partitions obtained by deleting D_1 and D_2 sets of nodes respectively from graph $G=(V,E)$, where $|D_1|=|D_2|=k$. Let L_1 and L_2 be the number components in M_1 and M_2 respectively and $L_1 \geq L_2$. If $\sigma_i=\sigma_j, \forall\ i,j\in M_1$, then we obtain a better objective function value by deleting the set D_1.*

Proof. Let $f(M_1)$ and $f(M_2)$ be the objective function values obtained by deleting D_1 and D_2 respectively. Let us assume that $f(M_1) > f(M_2)$. Let $u=\frac{(|V|-k)}{L_2}$. From Lemma 2.1, we have that $\frac{L_2(u)(u-1)}{2} \leq f(M_2)$. Then we have

$$\frac{L_2(u)(u-1)}{2} \leq f(M_2) < f(M_1) = \frac{L_1\left(\frac{|V|-k}{L_1}\right)\left(\frac{|V|-k}{L_1}-1\right)}{2}. \tag{2–14}$$

Examining (2–14) carefully reveals the necessary contradiction. Notice that (2–14) states

$$\frac{L_2\left(\frac{|V|-k}{L_2}\right)\left(\frac{|V|-k}{L_2}-1\right)}{2} < \frac{L_1\left(\frac{|V|-k}{L_1}\right)\left(\frac{|V|-k}{L_1}-1\right)}{2} \tag{2–15}$$

$$\Rightarrow \frac{(|V|-k)(|V|-k-L_2)}{2} < \frac{(|V|-k)(|V|-k-L_1)}{2} \tag{2–16}$$

$$\Rightarrow L_2 > L_1. \tag{2–17}$$

This contradicts the hypothesis that $L_1 \geq L_2$, and we have the result. □

We can now prove the following theorem regarding the complexity of the CNP.

Theorem 2.1. *The* K-CRITICAL NODE PROBLEM *is* NP-*complete.*

Proof. To show this, we must prove that (1) K-CNP $\in$ NP; (2) Some NP-complete problem reduces to K-CNP in polynomial time.

K-CNP $\in$ NP since given any graph $G = (V, E)$, and deleting any set of at most k nodes, we can determine the objective value in $\mathcal{O}(|E|)$ time using a depth-first search [5].

To complete the proof, we show a reduction from the INDEPENDENT SET PROBLEM (ISP) [35], which is well-known to be NP-complete [83]. Given a graph $G = (V, E)$, the ISP seeks to determine if G contains an independent set of size k. This is equivalent to determining if there exists an empty subgraph of G of size k by deleting $|V| - k$ nodes and their adjacent edges.

Let $\bar{G} = (\bar{V}, \bar{E})$ be the graph obtained by replacing each node $u \in G$ by a T-clique with one of the clique nodes coinciding with u. Note that if $T = 1$, then we have the original graph G. Consider the K-CNP on $\bar{G}$ which asks if there exists a partition M of $\bar{G}$ obtained by deleting $|V| - k$ nodes such that

$$\sum_{\forall\ i \in M} \frac{\sigma_i(\sigma_i - 1)}{2} \leq \frac{kT(T-1)}{2} + \frac{(|V|-k)(T-1)(T-2)}{2}.$$

We claim there is a one-to-one correspondence between the ISP on G and the CNP on $\bar{G}$. If there is an independent set of size k in G, it is clearly a solution to the CNP, as we will have k components of size T and $|V| - k$ components of size $T - 1$. This would result in the required objective function value for the CNP. Conversely, if there is a partition of M satisfying the objective by deleting $|V| - k$ nodes, then we have an independent set of size k.

We give a constructive proof to show this. The first part involves showing that the objective value of the CNP on $\bar{G}$ is always better when the nodes of the original graph G are deleted from $\bar{G}$ as opposed to deleting clique nodes. For a given CNP solution, let us assume that in a clique, a non-coinciding node is deleted and the coinciding node from this clique is not deleted. If we swap these nodes in the solution, that is, if we delete the coinciding node and replace the non-coinciding node, then the objective function value either remains the same, or decreases if the number of components increases.

Now let us assume that a non-coinciding clique node and its coinciding node are deleted from $\bar{G}$. In this case, if we swap from the solution set the non-coinciding node with an undeleted coinciding node, then again the objective value will either decrease or remain unchanged. To see this, let us assume the component corresponding to the clique with both its coinciding and non-coinciding nodes deleted is of size $(T - b)$, where $b \geq 2$, as there may be other non-coinciding nodes deleted from this clique. Also, let the coinciding node that was not deleted be a part of some component of size $T + a$. Now the objective function before the swap will be

$$Z_1 = S + \frac{(T+a)(T+a-1)}{2} + \frac{(T-b)(T-b-1)}{2}, \tag{2–18}$$

where S is the contribution from other components present in the graph. After swapping these two nodes, the objective function value would be

$$Z_2 = S + \frac{(T-1)(T-2)}{2} + \frac{(a)(a-1)}{2} + \frac{(T-b)(T-b+1)}{2}. \tag{2–19}$$

Now, if we take the difference we see that

$$Z_1 - Z_2 = aT + b - 1 \geq 0, \ \forall\ T \geq 0. \tag{2–20}$$

Since we are deleting only $|V| - k$ nodes, we have a partition M with

$$\sum_{\forall\ i \in M} \frac{\sigma_i(\sigma_i - 1)}{2} \leq \frac{kT(T-1)}{2} + \frac{(|V|-k)(T-1)(T-2)}{2},$$

by deleting only the nodes of the original graph. Since none of the new nodes (i.e., the nodes from the T-cliques) are deleted from $\bar{G}$, the deletion of the $|V| - k$ nodes results in exactly $|V| - k$ components of size $T - 1$. This contributes exactly $\frac{(|V|-k)(T-1)(T-2)}{2}$ towards the objective function. The remaining kT nodes form at most k components. Hence from Lemma 2.2, this contributes at least $\frac{kT(T-1)}{2}$ to the objective function. From Lemma 2.1, the kT nodes involve exactly k components of size T, representing the T-cliques of $\bar{G}$, with one node in each T-clique present in the original graph G, and none of them connected to each other. Hence deletion of $|V| - k$ nodes from $\bar{G}$ results in k independent nodes in the original graph G. This completes the proof. □

2.2.3 Integer Programming Formulations

When studying combinatorial problems, integer programming models are usually quite helpful for providing some of the formal properties of the problem [153]. With this in mind we now develop a linear integer programming formulation for the CNP.

To begin with, define the surjection $u : V \times V \mapsto \{0, 1\}$ as above. Further, we introduce a surjection $v : V \mapsto \{0, 1\}$ defined by

$$v_i := \begin{cases} 1, \text{ if node } i \text{ is deleted in the optimal solution,} \\ 0, \text{ otherwise.} \end{cases} \tag{2–21}$$

Then the CRITICAL NODE PROBLEM admits the following integer programming formulation

$$\textbf{(CNP-1)} \quad \text{Minimize} \quad \sum_{i,j \in V} u_{ij} \tag{2–22}$$

s.t.

$$u_{ij} + v_i + v_j \geq 1, \ \forall \ (i,j) \in E, \tag{2–23}$$

$$u_{ij} + u_{jk} - u_{ki} \leq 1, \ \forall \ (i,j,k) \in V, \tag{2–24}$$

$$u_{ij} - u_{jk} + u_{ki} \leq 1, \ \forall \ (i,j,k) \in V, \tag{2–25}$$

$$-u_{ij} + u_{jk} + u_{ki} \leq 1, \ \forall \ (i,j,k) \in V, \tag{2–26}$$

$$\sum_{i \in V} v_i \leq k, \tag{2–27}$$

$$u_{ij} \in \{0,1\}, \ \forall \ i,j \in V, \tag{2–28}$$

$$v_i \in \{0,1\}, \ \forall \ i \in V. \tag{2–29}$$

Note the objective is to find the set of k nodes whose removal results in a graph which has the minimum pari-wise connectivity between the remaining nodes. This is accomplished by the objective function. The first set of constraints in (2–23) implies that if nodes i and j are in different components and if there is an edge between them, then one of them must be deleted. Furthermore, constraints (2–24)-(2–26) together imply that for all triplets of nodes i, j, k, that if i and j are in same component and j and k are in same component, then necessarily k and i must be in the same component. Constraint (2–27) ensures that the total number of deleted nodes is less than or equal to k. Finally, (2–28) and (2–29) define the proper domains for the variables used. Thus, a solution to the integer programming formulation **CNP-1** characterizes a feasible solution to the CNP. On the other hand, it is clear that a feasible solution to the CNP will define at least one feasible solution to **CNP-1**. Therefore, **CNP-1** is a correct formulation for the CNP.

We note here that in all likelihood, there exist alternative mathematical programming formulations for the CNP. For example, notice that the conditions which satisfy the circular constraints (2–24), (2–25), and (2–26) in **CNP-1** can be satisfied by the single constraint $u_{ij} + u_{jk} + u_{ki} \neq 2, \forall\ (i, j, k) \in V$. By appropriately defining a new set of binary variables, this constraint set could be incorporated into the model. This might be useful for breaking down the symmetry of the problem if one was attempting to exploit the polyhedral structure of the model.

Notice that if the objective was a function of the number of components, then an approximation for the MAXIMUM K-CUT PROBLEM [83, 119] could be employed by modifying the cost function of the Gomory-Hu tree [92]. An even simpler approach would be to identify the cut vertices in the graph, if any exist. Studies to assess the vulnerability of a network with similar objective functions are studied in [137, 149]. The objective function in [149] is to maximize traffic flow while deleting a set of k edges [149] from the graph. In [137], a very similar objective to the one proposed in this work is presented. Whereas our objective is to minimize the pair-wise connectivity between the nodes after deleting k nodes, the objective in [137] maximizes the node disconnectivity between a set of source nodes and sink nodes by deleting a set of k arcs.

Recall that $\sum_{i,j \in V} u_{ij}$ is a measure of the total pair-wise connectivity of the graph. Notice that since u_{ij} is binary and equal to 1 if and only if i and j are in the same component in the optimal solution, the objective function could be rewritten as

$$\sum_{j \in S} \frac{\sigma_j(\sigma_j - 1)}{2}, \tag{2–30}$$

where S is set of all components and σ_j is the size of the j-th component, which can be easily identified by fast algorithms like breadth or depth first search algorithms in $\mathcal{O}(|E|)$ time using an adjacency list representation of the network [5, 61]. We will use (2–30) as the objective function optimized by the heuristic in the following section. In addition to the relative ease of calculating the cardinality of the components of a graph, there is an intuitive explanation for the choice of (2–30) as our objective function. As we proved in Lemma 2.1 and Lemma 2.2 above, optimizing (2–30) maximizes the number of connected

```
procedure CriticalNode($G, k$)
1    MIS $\leftarrow$ MaximalIndepSet($G$)
2    while ($|\text{MIS}| \neq |V| - k$) do
3        $i \leftarrow \arg\min\left\{\sum_{j \in S} \frac{\sigma_j(\sigma_j - 1)}{2} : S \in G(\text{MIS} \cup \{j\}), j \in V \setminus \text{MIS}\right\}$
4        MIS $\leftarrow$ MIS $\cup \{i\}$
5    end while
6    return $V \setminus$ MIS    /* set of $k$ nodes to delete */
end procedure CriticalNode
```

Figure 2-1. Heuristic for detecting critical nodes.

components while simultaneously minimizing the variance in the component sizes. For example, consider an arbitrary unweighted graph with 150 nodes. According to our objective, it is more preferable to have a partition with 3 components each with 50 nodes, as opposed to a partition with 5 components with one having 146 nodes and the rest of them having a single node.

2.2.4 Heuristic for Detecting Critical Nodes

Pseudo-code for the proposed heuristic is provided in Figure 2-1. To begin with, the algorithm finds a maximal independent set (MIS). This set is initially empty, and is computed sequentially as follows. First, a single vertex is added to the set. Next by iterating through the vertices, a node that is not adjacent to the starting node is added to the MIS. Then a vertex adjacent to neither of these is added, and so on. This is continued until we can find no more vertices to include, and thus the set is maximal independent.

After the initial MIS is computed, in the loop from lines 2-5, the heuristic greedily selects the node $i \in V$ not currently in the MIS which returns the minimum objective function for the graph $G(\text{MIS} \cup \{i\})$. The set MIS is augmented to include node i, and the process repeats until $|\text{MIS}| = |V| - k$. At this time, the method terminates and the set of critical nodes to be deleted is given as those nodes $j \in V$ such that $j \in V \setminus \text{MIS}$.

The intuition behind using an independent set is that the subgraph induced by this set is empty. Stated otherwise, the deletion of those nodes that are *not* in the independent set will result in an empty subgraph. Notice that this will provide the optimal solution for an instance of the CNP if $|\text{MIS}| \geq |V| - k$. However, if the size of the MIS is less than $|V| - k$, we simply keep adding nodes which provide the best objective value to the set until it reaches the desired size. The heuristic is computationally efficient and the complexity is given in the following theorem.

Theorem 2.2. *The proposed algorithm has complexity* $\mathcal{O}(|V|^2|E|)$.

Proof. To begin with, finding the MIS using the sequential method described above requires linear time. Next, the **while** loop from lines 2-5 will iterate at most $\mathcal{O}(|V| - k)$ times. In each iteration, the number of search operations decreases from $|V| - 1$ to $|V| - (|V| - k) = k$. Note that we are performing the search of a sparse graph, which is initially empty. There will be one comparison step for every search performed in order to determine the node that provides the minimum increase in the objective function. This will in turn be dominated by the complexity of the search procedure which requires $\mathcal{O}(|E|)$ time. Hence, the total number of iterations will be

$$\mathcal{O}(|V| - 1 + |V| - 2 + \cdots + |V| - |V| + k) = \mathcal{O}\left(\sum_{i=1}^{|V|-1} i - \sum_{i=1}^{k-1} i\right) = \mathcal{O}(|V|^2 - k^2) = \mathcal{O}(|V|^2).$$

Thus the overall complexity is $O(|V|^2|E|)$, and the proof is complete. □

The proposed algorithm finds a feasible solution to the CRITICAL NODE PROBLEM; however, the solution is not guaranteed to be globally or locally optimal. Therefore, we can enhance the heuristic with the application of a local search routine as follows. Consider the pseudo-code presented in Figure 2-2. The routine receives as input the solution from the `CriticalNode` heuristic and performs a 2-exchange local search. Let $f : V \mapsto \mathbb{Z}$ be a function returning the objective function value for a given set in the sense of (2–30) above. That is, consider a pair of nodes i and j such that $i \in \text{MIS}$ and $j \notin \text{MIS}$. Then for all such pairs, we set $j \in \text{MIS}$ and $i \notin \text{MIS}$ and examine the change in the objective function. If it improves, then the swap is kept; otherwise, we undo the swap and continue to the next node pair. Notice that the loop from lines 3-16 repeats while

```
procedure LocalSearch($V \setminus$ MIS)
1   $X^* \leftarrow$ MIS
2   $local\_improvement \leftarrow$ .TRUE.
3   while $local\_improvement$ do
4     $local\_improvement \leftarrow$ .FALSE.
5     if $i \in$ MIS and $j \notin$ MIS then
6       MIS $\leftarrow$ MIS $\setminus i$
7       MIS $\leftarrow$ MIS $\cup j$
8       if $f(\text{MIS}) < f(X^*)$ then
9         $X^* \leftarrow$ MIS
10        $local\_improvement \leftarrow$ .TRUE.
11      else
12        MIS $\leftarrow$ MIS $\setminus j$   /* undo swap */
13        MIS $\leftarrow$ MIS $\cup i$
14      end if
15    end if
16  end while
17  return $(V \setminus X^*)$   /* set of $k$ nodes to delete */
end procedure LocalSearch
```

Figure 2-2. Local search algorithm for critical node heuristic.

the solution is not locally optimal. This general statement can lead to implementation problems and it is a common practice to limit the number of local search iterations by some user defined value, say U. The intuition is that the as $U \to \infty$, the solution becomes optimal with respect to its local neighborhood.

Theorem 2.3. *If the number of iterations of the local search is bounded by a constant $U \in \mathbb{R}$ as described above, then the complexity of the procedure is $\mathcal{O}(|V|^2 U)$.*

Proof. The is clear as the while loop from lines 3-16 will iterate U times. Since each iteration requires an examination of $|V|^2$ components, we have the proof. □

Finally, we can combine the construction and local improvement algorithms into one multi-start heuristic `CriticalNodeLS` as shown in Figure 2-3. This procedure produces `MaxIter` local optima and the overall best solution from all iterations is returned. In order to implement the multi-start framework, the starting node for each maximal independent set is randomly chosen. Since the initial MIS is created deterministically, this node is only

```
procedure CriticalNodeLS($G, k$)
1   $X^* \leftarrow \emptyset$
2   $f(X^*) \leftarrow \infty$
3   for $j = 1$ to MaxIter do
4       $X \leftarrow$ CriticalNode($G, k$)
5       $X \leftarrow$ LocalSearch(X)
6       if $f(X) < f(X^*)$ then
7           $X^* \leftarrow X$
8       end if
9   end
10  return $(V \setminus X^*)$    /* set of $k$ nodes to delete */
end procedure CriticalNodeLS
```

Figure 2-3. Heuristic with local search for detecting critical nodes.

accepted as a starting node if it has not been previously selected. Therefore, we see that `MaxIter` will be bounded above by $|V|$. This simple randomization scheme ensures that different areas of the solution space are explored in each iteration.

Theorem 2.4. *The* `CriticalNodeLS` *heuristic has overall complexity of* $\mathcal{O}(T(|V|^2|E| + |V|^2U))$, *where* $T =$ `MaxIter`, *and* U *is the iteration limit on the local search.*

Proof. This result follows directly from Theorem 2.2 and Theorem 2.3 above. □

2.2.5 Computational Results

The proposed heuristic was implemented in the `C++` programming language and compiled using GNU `g++` version 3.4.4, using optimization flags `-O2`. It was tested on a `PC` equipped with a 1700 MHz Intel® Pentium® M processor and 1.0 gigabytes of RAM operating under the Microsoft® Windows® XP Professional environment. The parameter `MaxIter` was set equal to $|V|$, and the number of iterations of the local search U was 2. It is reasonable to forgo the implementation with U, and simply allow the local search to examine all swaps until the solution is locally optimal. Alternatively, one could allow the local search to iterate for some maximum number of iterations after which no improving solution was found. Our experiments indicated that for the graphs tested, a value of $U = 2$ was sufficient to provide excellent solutions within reasonable computing times.

As a basis of comparison, we have implemented the integer programming model for the CRITICAL NODE PROBLEM using the CPLEX^{TM} version 9 optimization suite from ILOG [63]. CPLEX contains an implementation of the simplex method [110], and uses a branch and bound algorithm [199] together with advanced cutting-plane techniques [118, 154].

We tested the IP model and heuristic on a set of randomly generated graphs ranging in size from 75 to 150 nodes with varying densities. The graphs were generated with version 1.4 of the publicly available Barabási graph generator by Dreier [70]. For each random instance, we report solutions for 3 values of k, the number of nodes to be deleted. In addition, we have tested the algorithms on the terrorist network compiled by Krebs [123] shown in Figure 2-10. This network depicts the relationships between the terrorists involved in the horrific attacks of September 11, 2001. The graph was constructed after the attacks with data which were publicly available before 9/11.

We begin by providing the results from the terrorist network [123] shown in Figure 2-10. This graph has 62 nodes and 153 edges. Notice that node 38 is the central node with degree 22. We applied the IP formulation and the heuristic to this network with 6 values of k. The results are provided in Table 2-1. Notice that for all values of k, the heuristic computed the optimal solution requiring on average 0.013 seconds of computation time. The average time to compute the optimal solution using CPLEX was 5387.31 seconds. Clearly even for this relatively small network, the heuristic is the method of choice. Figure 2-5 shows the resulting graph of the terrorist network according to the optimal solution to the CNP for the instance of $k = 20$.

In order to determine the scalability and robustness, the proposed heuristic was tested on a set of randomly generated scale-free graphs. Table 3-4 presents the results of the heuristic and the optimal solver when applied to the random instances. For each instance, we report the number of nodes and arcs, value of k being considered, the optimal solution and computation time, and finally the heuristic solution and the corresponding

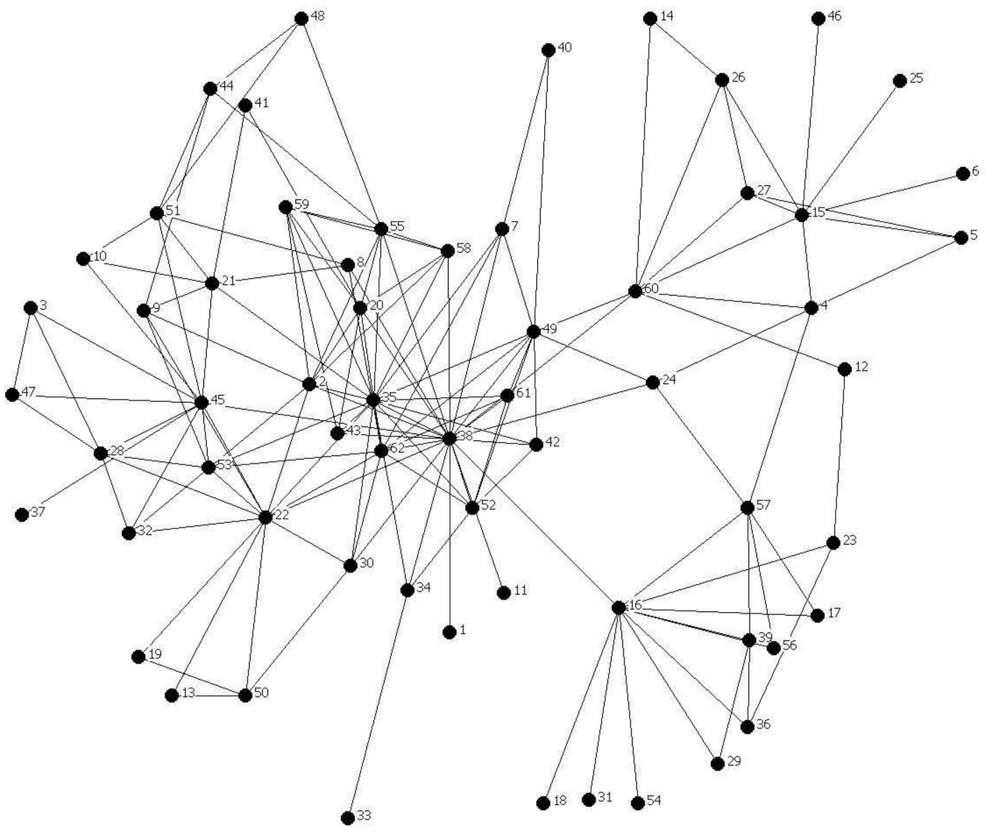


Figure 2-4. Terrorist network compiled by Krebs.

computation time. For each graph, we report solutions for 3 different values of k. The graphs were generated with version 1.4 of the publicly available Barabási graph generator by Dreier [70].

Notice that for all instances tested, our method was able to compute the optimal solution. Furthermore, the required time to compute the optimal solution was less than one second for all but one instance, averaging only 0.33 seconds for all 27 instances. On the other hand, CPLEX required 289.44 seconds on average to compute the optimal solution, requiring over 5000 seconds in the worst case. Our computational experiments

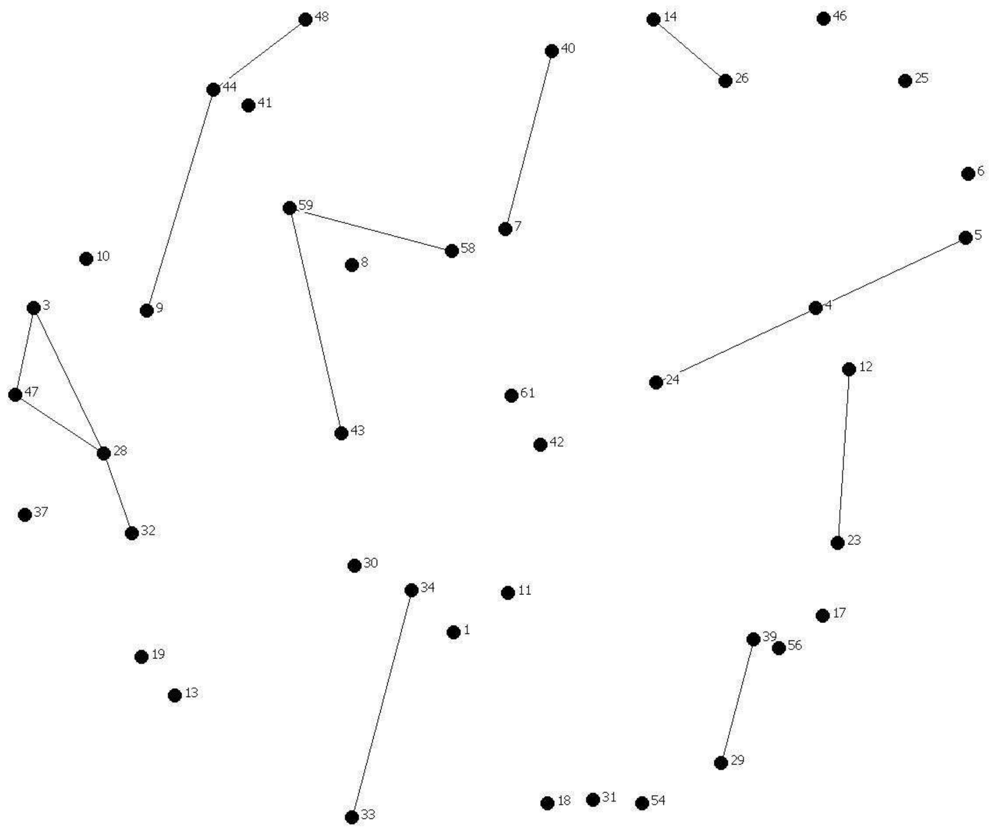


Figure 2-5. Optimal solution when $k = 20$.

indicate that the proposed heuristic is able to efficiently provide excellent solutions for large-scale instances of the CNP[*] .

2.3 Cardinality Constrained Problem

We now provide the formulation for a slightly modified version of the CNP based on constraining the connectivity index of the nodes in the graph. Given a graph $G = (V, E)$, the *connectivity index* of a node is defined as the number of nodes reachable from that vertex (see Figure 2-6 for examples). To constrain the network connectivity in optimization models, we can impose constraints on the connectivity indices.

[*] The experimental data including the instances tested are available at the following url: `http://plaza.ufl.edu/clayton8/cnp/`

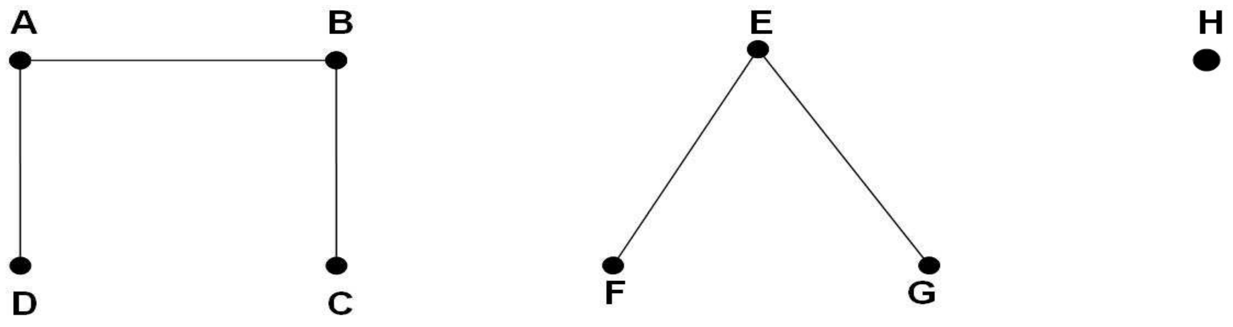


Figure 2-6. Connectivity Index of nodes A,B,C,D is 3. Connectivity Index of E,F,G is 2. Connectivity Index of H is 0.

This leads to a cardinality constrained version of the CNP which we aptly refer to as the CARDINALITY CONSTRAINED CRITICAL NODE DETECTION PROBLEM (CC-CNP). The objective is to detect a set of nodes $A \subseteq V$ such that the connectivity indices of the nodes in the vertex deleted subgraph $G(V \setminus A)$ is less than some threshold value, say L. Using the same definition of the variables as in the previous subsection, we can formulate the CC-CNP as the following integer linear programming problem.

$$\textbf{(CC-CNP-1)} \quad \text{Minimize} \quad \sum_{i \in V} v_i \tag{2–31}$$

$$\text{s.t.}$$

$$u_{ij} + v_i + v_j \geq 1, \ \forall \ (i,j) \in E, \tag{2–32}$$

$$u_{ij} + u_{jk} + u_{ki} \neq 2, \ \forall \ (i,j,k) \in V, \tag{2–33}$$

$$\sum_{i,j \in V} u_{ij} \leq L, \tag{2–34}$$

$$u_{ij} \in \{0,1\}, \ \forall \ i,j \in V, \tag{2–35}$$

$$v_i \in \{0,1\}, \ \forall \ i \in V, \tag{2–36}$$

where L is the maximum allowable connectivity index for any node in V.

First, we see that the objective function given clearly minimizes the number of nodes deleted. Constraints (2–32) and (2–33) follow exactly as in the CNP formulation. The only difference is now we must constrain the connectivity index of each node. This is accomplished by constraint (2–34). Finally constraints (2–35) and (2–36) define the domains of the decision variables.

The proof of NP-completeness is obtained from the result proved by Krishnamoorthy and Deo [124] for a class of node deletion problems.

Lemma 2.3. *[124] Let π be a specified graph property that is determined by its components, and suppose there is a graph F with a node "s" such that the following hold:*

(1) The graph F and the subgraph resulting after deleting node "s" from F satisfy .

(2) If a node x is added to the graph F and nodes x and "s" are joined by an edge, then the resulting graph is a forbidden induced subgraph for property .

then the node-cover problem is polynomially transformable to the node deletion problem for property π.

Let us consider the graphs with the property that the size of any connected component in the graph is less than or equal to L, where $L > 0$. Then CC - CNP is a node deletion problem [124] for this property. Additionally, the property satisfies the conditions stated in Lemma 1 (one can take any connected component of size L as graph F). Thus we have a polynomial time reduction from the NODE COVER PROBLEM which is well-known to be NP-complete [82]. A precise generalization of the property is provided in [131] as hereditary and non-trivial. A property π is hereditary if a graph satisfies π , then every node and edge induced subgraph of the graph satisfies π and it is non-trivial if there are infintiely many graphs that satisfy the property. The node deletion problem for hereditary properties was later proved to be max-SNP hard as the transformation provided was approximation preserving [136]. This implies that there is no polynomial time approximation scheme to solve the problem unless P=NP.

2.3.1 CC-CNP Heuristic

We can appropriately modify the heuristics for CNP to solve instances of CC-CNP. To do this, notice that now we are only concerned with the connectivity indices of the nodes. Stated differently, we are only concerned with the sizes of the components in

```
procedure ConstrainedCriticalNode($G, L$)
1   MIS $\leftarrow$ MaximalIndepSet($G$)
2   OPT $\leftarrow$ FALSE
3   NoAdd $\leftarrow 0$
4   while (OPT .NOT.TRUE) do
5     for ($i = 1$ to $|V|$) do
6       if $\left(\frac{|s|(|s|-1)}{2} \leq L \ \forall \ s \in S \subseteq G(\text{MIS} \ \cup \{i\}) : i \in V \setminus \text{MIS}\right)$ then
7         MIS $\leftarrow$ MIS $\cup \{i\}$
8       else
9         NoAdd $\leftarrow$ NoAdd $+1$
10      end if
11      if (NoAdd $= |V| - |\text{MIS}|$) then
12        OPT $\leftarrow$ TRUE
13        BREAK
14      end if
15    end for
16  end while
17  return $V \setminus$ MIS     /* set of nodes to delete */
end procedure ConstrainedCriticalNode
```

Figure 2-7. Heuristic for the CARDINALITY CONSTRAINED CRITICAL NODE PROBLEM.

the vertex deleted subgraph. Unlike before, there is no limit on the number of critical nodes we choose, so long as the connectivity constraints are satisfied. We could generate pathological instances to demonstrate its inefficiency, so we provide a genetic algorithm in a later section.

Pseudo-code for the proposed algorithm is provided in Figure 2-7. The heuristic starts off the same as before by identifying a maximal independent set (MIS). Then, the boolean variable OPT is set to `FALSE`. Finally in line 3, a variable NoAdd is initialized to 0. This variable determines when to exit the main loop from lines 4-16. After this loop is entered, the procedure iterates through the vertices and determines which can be added back to the graph while still maintaining feasibility. If vertex i can be added, MIS is augmented to include i in step 7, otherwise NoAdd is incremented. If NoAdd is ever equal to $|V| - |\text{MIS}|$, then no nodes can be returned to the graph and OPT is set to `TRUE`. Then loop is then exited and the algorithm returns the set of nodes to be deleted, i.e. $V \setminus \text{MIS}$.

Theorem 2.5. *The worst-case complexity of the* `ConstrainedCriticalNode` *heuristic is* $\mathcal{O}(|V|^2 + |V||E|)$.

Proof. This proof is similar to the proof of Theorem 2.2 above. The loop from lines 4-16 will iterate at most $\mathcal{O}(|V|)$ times. Each loop requires at most $\mathcal{O}(|V| + |E|)$ time to verify the if a solution will remain feasible after a node is re-included in the graph. Thus we have the result. □

2.3.2 Genetic Algorithm for the CC-CNP

```
procedure GeneticAlgorithm
1    Generate population $P_k$
2    Evaluate population $P_k$
3    while terminating condition not met do
4        Select individuals from $P_k$ and copy to $P_{k+1}$
5        Crossover individuals from $P_k$ and put in $P_{k+1}$
6        Mutate individuals from $P_k$ and put in $P_{k+1}$
7        Evaluate population $P_{k+1}$
8        $P_k \leftarrow P_{k+1}$
9        $P_{k+1} \leftarrow \emptyset$
10   end while
11   return best individual in $P_k$
end procedure GeneticAlgorithm
```

Figure 2-8. Pseudo-code for a generic genetic algorithm.

Genetic algorithms (GAs) mimic the biological process of evolution. In this subsection, we describe the implementation of a GA for the CC-CNP. Recall the general structure of a GA as outlined in Figure 2-8. When designing a genetic algorithm for an optimization problem, one must provide a means to encode the population, define the crossover operator, and define the mutation operator which allows for random changes in offspring to help prevent the algorithm from converging prematurely [13].

For our implementation, we use binary vectors as an encoding scheme for individuals within the population of solutions. When the population is generated, (Figure 2-8, line 1), a random deviate from a distribution which is uniform onto $(0, 1) \in \mathbb{R}$ is generated for each node. If the deviate exceeds some specified value, the corresponding allele is assigned value 1, indicating this node should be deleted. Otherwise, the allele is given a 0, implying it is not deleted. In order to evaluate the fitness of the population, per line 2, we must determine whether each individual solution is feasible or not. Determining feasibility is a relatively straightforward task and can accomplished in $\mathcal{O}(|V| + |E|)$ using a depth-first search [5].

In order to evolve the population over successive generations, we use a reproduction scheme in which the parents chosen to produce the offspring are selected using the binary tournament method [144, 198]. Using this method, two chromosomes are chosen at random from the population and the one having the best fitness, i.e. the lowest objective function value, is kept as a parent. The process is then repeated to select the second parent. The two parents are then combined using a crossover operator to produce an offspring [107].

To breed new solutions, we implement a strategy known as *parameterized uniform crossover* [186]. This method works as follows. After the selection of the parents, refer to the parent having the best fitness as MOM. For each of the nodes (alleles), a biased coin is tossed. If the result is heads, then the allele from the MOM chromosome is chosen. Otherwise, the allele from the least fit parent, call it DAD, is selected. The probability that the coin lands on heads is known as CrossProb, and is determined empirically. Figure 3-3 provides an example of a potential crossover when the number of nodes is 5 and CrossProb = 0.65 [13].

Coin Toss	T	H	H	T	H
MOM	0.56	0.81	0.22	0.7	0.86
DAD	0.29	0.49	0.98	0.12	0.32
Offspring	0.29	0.81	0.22	0.12	0.86

Figure 2-9. An example of the crossover operation. In this case, CrossProb = 0.65.

After the child is produced, the mutation operator is applied. Mutation is a randomizing agent which helps prevent the GA from converging prematurely and escape to local optima. This process works by flipping a biased coin for each allele of the chromosome. The probability of the coin landing heads, known as the mutation rate (`MutRate`) is typically a very small user defined value. If the result is heads, then the value of the corresponding allele is reversed. For our implementation, `MutRate` $= 0.03$.

After the crossover and mutation operators create the new offspring, it replaces a current member of the population using the so-called *steady-state* model [55, 107, 144]. Using this methodology, the child replaces the least fit member of the population, provided that a clone of the child is not an existing member in the population. This method ensures that the worst element of the population is monotonically improving in every generation. In the subsequent iteration, the child becomes eligible to be a parent and the process repeats. Though the GA does converge in probability to the optimal solution, it is common to stop the procedure after some "terminating condition" (see Figure 2-8, line 3) is satisfied. This condition could be one of several things including, a maximum running time, a target objective value, or a limit on the number of generations. For our implementation, we use the latter option and the best solution after `MaxGen` generations is returned.

2.3.3 Computational Results

All of the proposed heuristics were implemented in the `C++` programming language and complied using GNU `g++` version 3.4.4, using optimization flags `-O2`. It was tested on a `PC` equipped with a 1700MHz Intel® Pentium® M processor and 1.0 gigabytes of RAM operating under the Microsoft® Windows® XP Professional environment.

In order to determine the scalability and robustness, the proposed heuristic was tested on a set of randomly generated scale-free graphs. Table 3-4 presents the results of the heuristic and the optimal solver when applied to the random instances. For each instance, we report the number of nodes and arcs, the value of k being considered, the optimal solution and computation time required by CPLEX, and finally the heuristic solution and

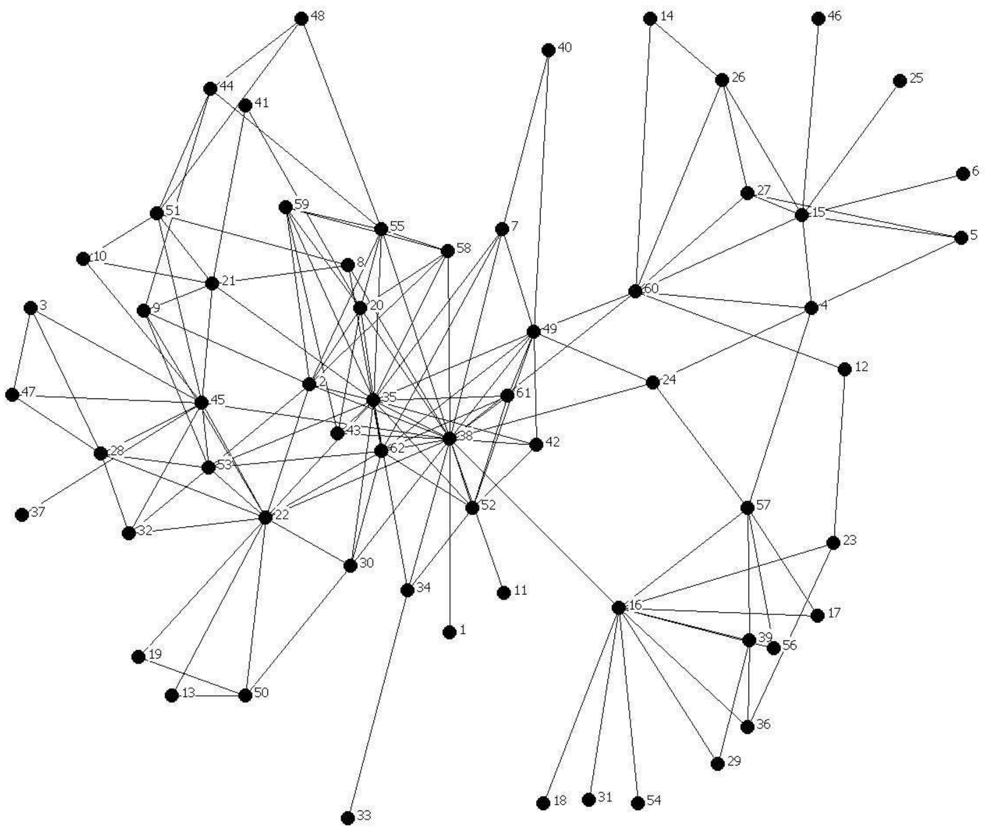


Figure 2-10. Terrorist network compiled by Krebs [123].

the corresponding computation time. For each graph, we report solutions for 3 different values of k.

Notice that for all instances tested, our method was able to compute the optimal solution. Furthermore, the required time to compute the optimal solution was less than one second for all but one instance, averaging only 0.33 seconds for all 27 instances. On the other hand, CPLEX required 289.44 seconds on average to compute the optimal solution, requiring over 5000 seconds in the worst case. Our computational experiments indicate that the proposed heuristic is able to efficiently provide excellent solutions for large-scale instances of the CNP.

2.3.4 CC-CNP Results

We continue with the results of the two algorithms developed for the CC-CNP, namely the combinatorial algorithm and the genetic algorithm. As above, we tested the IP model and both heuristics on the terrorist network [123] and a set of randomly generated graphs. For each instance tested, we report solutions for 3 values of L, the connectivity index threshold. Finally, we have implemented the integer programming model for the CC-CNP using CPLEX$^{\text{TM}}$.

Table 2-3 presents computational results of the IP model and heuristic solutions when tested on the terrorist network data. Notice that for all 5 values of L tested, the genetic algorithm and the combinatorial algorithm with local search (ComAlg + LS) computed optimal solutions. Figure 2-11 shows the optimal solution for the case when $L = 4$.

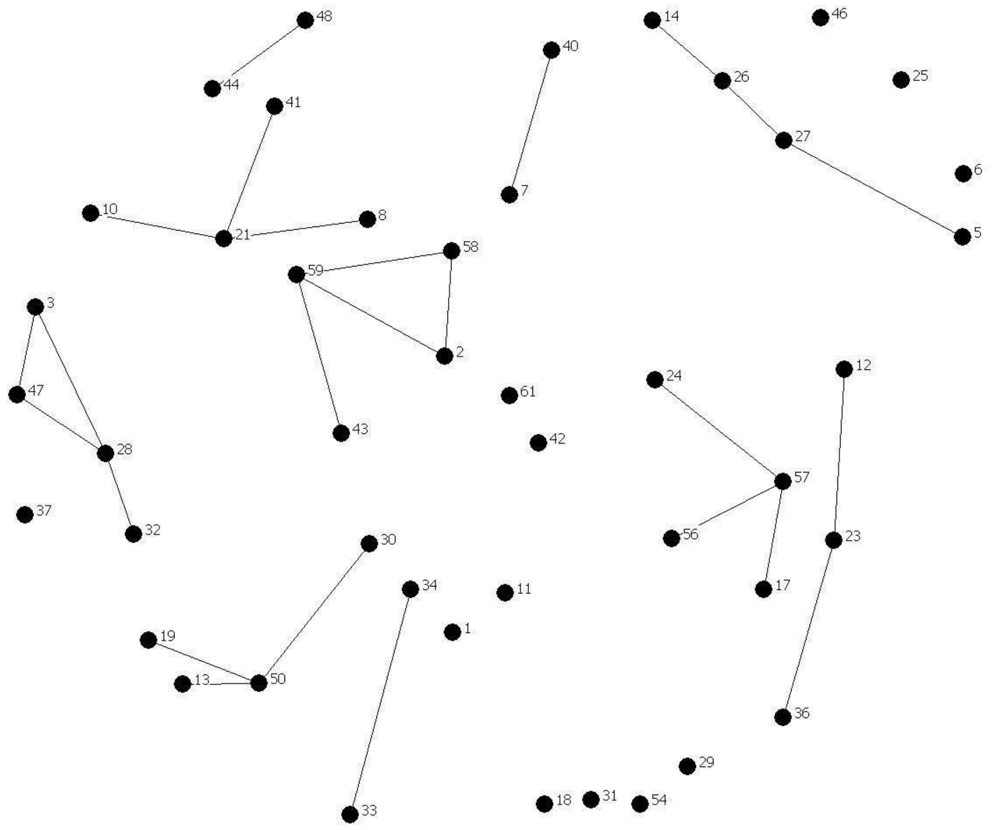


Figure 2-11. Optimal solution when $L = 4$.

We now consider the performance of the algorithms when tested on the randomly generated data sets containing up to 50 nodes taken from [11]. The results are shown in Table 2-4. For these relatively small instances, we were able to compute the optimal solutions using CPLEX. For each instance, we provide solutions for 3 values of L, the maximum connectivity index. Notice that for these problems, the genetic algorithm computed optimal solutions for each instance tested in a fraction of the time required by CPLEX. The combinatorial heuristic found optimal solutions for all but 3 cases requiring approximately half of the time of the GA.

Table 2-5 presents the solutions for the random instances from 75 to 150 nodes [11, 15]. Again, in order to demonstrate the robustness of the heuristics, we provide solutions for 3 values of L for each instance. In this table, we provide the results for the genetic algorithm and combinatorial heuristic with and without the local search enhancement. CPLEX was unable to compute optimal solutions within reasonable time limits for any of the instances represented in this table.

We see from this table that the in terms of solution quality the GA is the best performing method. The $\texttt{ComAlg} + \texttt{LS}$ also favors well, but requires more computation time than the GA and requires more computing time on average. The combinatorial algorithm without the local search procedure produces solution which are arguably reasonable given that the required computation time is over 36 times faster than the GA, while the solutions are only 1.2 times worse than those computed by the GA. Nevertheless, the genetic algorithm required only 5.748 seconds on average to compute the best solution. The trade-off of solution quality versus computation time is a decision that would be made by an operator depending on the size of the network and the time constraints imposed on detecting the critical nodes of a given graph.

2.4 Concluding Remarks

In this chapter, we proposed several methods of for the the detection of the critical nodes whose deletion results in the maximum network disconnectivity. In general, the problem of detecting critical nodes has a wide variety of applications from jamming communication networks and other anti-terrorism applications, to epidemiology and transportation science [11, 15].

In particular, we examined two problems, namely the CRITICAL NODE PROBLEM (CNP) as well as the CARDINALITY CONSTRAINED CNP (CC-CNP). Given a graph and an integer k, the objective of the CNP is to detect a set of k critical nodes whose deletion results in the maximum number of disconnected components whose cardinalities have the minimum variance. The definition of the CC-CNP is slightly different in that instead of given $k \in \mathbb{Z}$, the maximum number of nodes to delete, we are given some value $L \in \mathbb{Z}$ which represents the maximum connectivity index a node may have. The objective in this case is to delete the minimum number of nodes while ensuring that the connectivity index of each node does not exceed L.

The proposed problems were modeled as integer linear programming problems. Then we proved that the corresponding decision problems are NP-complete. Furthermore, we proposed a several heuristics for efficiently computing quality solutions to large-scale instances. The heuristic proposed for the CNP was a combinatorial algorithm which exploited properties of the graph in order to compute basic feasible solutions. The method was further intensified by the application of a local search mechanism. By using the integer programming formulation we were able to determine the precision of our heuristic by comparing their relative solutions and computation times for several networks. The computational experiments indicated that the heuristic found optimal solutions for all instances tested in a fraction of the time required by the commercial IP solver CPLEX.

For the CC-CNP we proposed two algorithms, namely a modified version of the combinatorial algorithm described above and a genetic algorithm [90]. Once again, the computational experiments indicated that both methods are robust and are able to efficiently compute approximate solutions for instances up to 150 nodes.

We also conclude with a few words on the possibility of future expansion of this work. A heuristic exploration of cutting plane algorithms on the IP formulation would be an interesting alternative. Other heuristic approaches worthy of investigation include hybridizing the genetic algorithm with the addition of a local search or path-relinking enhancement procedure [87]. Finally, the local search used in the combinatorial algorithm was a simple 2-exchange method, which was the cause of a significant slow down in computation as noted in Table 2-5. A more sophisticated local search such as a modification of the one proposed by Resende and Werneck [174, 175] should be a major focus of attention.

Furthermore, a node weighted version will be an interesting study. As it is rational to perceive applications containing weighted networks in which the cost of deleting one node is different from another. Also, pertaining to applications outside the scope of jamming networks, a study of epidemic threshold variation with respect to the heuristic results will help determine the impacts on contagion suppression in biological and social networks.

Table 2-1. Results of IP model and heuristic on terrorist network data from Krebs.

Instance	IP Model		Heuristic	
Nodes Deleted (k)	Objective Value	Execution Time (s)	Objective Value	Execution Time (s)
20	20	12.69	20	0.01
15	61	277.77	61	0.01
10	169	3337.06	169	0.02
9	214	2792.33	214	0.02
8	282	15111.94	282	0.01
7	327	10792.08	327	0.01

Table 2-2. Results of IP model and heuristic on randomly generated scale free graphs.

Instance			IP Model		Heuristic		Heuristic + LS	
Nodes	Arcs	Deleted Nodes (k)	Obj Value	Comp Time (s)	Obj Value	Comp Time (s)	Obj Value	Comp Time (s)
75	140	20	36	66.7	92	0.12	36	0.03
75	140	25	18	33.28	39	0.28	18	0.03
75	140	30	7	4.23	18	0.02	7	0.04
75	210	25	26	93.71	78	0.1	26	0.04
75	210	30	8	3.57	31	0.05	8	0.05
75	210	35	2	4.36	16	0.18	2	0.04
75	280	33	26	749.19	54	0.00	26	0.04
75	280	35	20	164.34	38	0.09	20	0.06
75	280	37	13	83.98	24	0.39	13	0.11
100	194	25	44	151.14	142	0.731	44	0.09
100	194	30	20	59.66	72	0.56	20	0.11
100	194	35	10	8.51	33	0.66	10	0.12
100	285	40	23	136.47	48	1.151	23	0.11
100	285	42	17	263.82	38	0.4	17	0.17
100	285	45	11	16.78	29	0.53	11	0.23
100	380	45	22	128.13	58	0.58	22	0.15
100	380	47	16	243.07	42	1.191	16	0.16
100	380	50	10	228.72	23	0.31	10	0.11
125	240	33	62	5047.51	97	0.721	62	0.30
125	240	40	29	118.92	49	1.562	29	0.24
125	240	45	16	17.09	32	0.14	16	0.39
150	290	40	40	41.6	125	1.832	40	0.47
150	290	50	12	26.29	64	2.773	12	0.831
150	290	60	1	24.92	35	1.091	1	0.851
150	435	61	19	29.55	53	2.313	19	0.741
150	435	65	13	31.45	37	0.991	13	1.952
150	435	67	11	37.91	31	0.52	11	0.801

Table 2-3. Results of IP model and heuristics on terrorist network data from [123].

Instance	IP Model		Genetic Alg		ComAlg		ComAlg + LS	
Max Conn. Index (L)	Obj Val	Comp Time (s)	Obj Val	Comp Time (s)	Obj Val	Comp Time (s)	Obj Val	Comp Time (s)
3	21	188.98	21	0.25	22	0.01	21	0.1
4	17	886.09	17	0.741	19	0.01	17	0.45
5	15	30051.09	15	0.871	20	0.18	25	1.331
8	–	–	13	0.39	14	0.05	13	0.07
10	–	–	11	0.741	12	0.07	11	0.05

Table 2-4. Results of the IP model and genetic algorithm and the combinatorial heuristic on randomly generated scale free graphs.

Instance			IP Model		Genetic Alg		ComAlg + LS	
Nodes	Arcs	Max Conn. Index (L)	Obj Value	Comp Time (s)	Obj Value	Comp Time (s)	Obj Value	Comp Time (s)
20	45	2	9	0.04	9	0.02	9	0.03
20	45	4	6	0.13	6	0.04	6	0.862
20	45	8	5	0.39	5	0.04	5	1.482
25	60	2	11	0.07	11	0.49	11	0.08
25	60	4	9	14.1	9	2.113	10	0.01
25	60	8	7	26.64	7	0.05	8	0.06
30	50	2	11	0.07	11	0.06	11	0.01
30	50	4	8	0.1	8	0.05	8	0
30	50	8	6	1152.15	6	0.09	6	0
30	75	4	10	18.77	10	0.14	10	0.02
30	75	6	9	442.41	9	0.09	9	0.04
30	75	10	7	64.94	7	0.18	8	0
35	60	2	12	0.13	12	0.14	12	0.14
35	60	4	8	29.89	8	0.711	8	0
35	60	6	7	31.61	7	0.31	7	0.01
40	70	2	15	0.17	15	0.1	15	0.101
40	70	4	11	341.97	11	0.06	11	0
40	70	6	8	78.94	8	0.2	8	0.04
45	80	2	16	0.24	16	0.06	16	0.1
45	80	4	11	48.17	11	0.05	11	0.02
45	80	6	8	118.23	8	0.09	8	0.071
50	135	2	19	0.36	19	0.27	19	0.05
50	135	4	15	165.18	15	0.63	15	0.291
50	135	6	14	5722.88	14	0.721	14	0.03
Total (Sum)			24	8257.58	24	6.705	27	3.417

Table 2-5. Comparative results of the genetic algorithm and the combinatorial heuristic when tested on the larger random graphs. Due to the complexity, we were unable to compute the corresponding optimal solutions.

Instance			Genetic Algorithm		ComAlg		ComAlg + LS	
Nodes	Arcs	Max Conn. Index (L)	Obj Value	Comp Time (s)	Obj Value	Comp Time (s)	Obj Value	Comp Time (s)
75	140	5	18	1.622	21	0	18	1.502
75	140	8	14	1.442	20	0.02	14	1.181
75	140	10	12	1.231	20	0.12	12	3.364
75	210	5	23	1.532	29	0.01	23	18.476
75	210	8	21	2.443	23	0.01	22	2.934
75	210	10	20	2.794	24	0.09	20	21.17
75	280	5	31	3.464	35	0.101	31	3.144
75	280	8	29	2.874	31	0.05	29	3.746
75	280	10	28	3.775	30	0.13	28	4.787
100	194	5	22	5.317	33	0.02	22	2.774
100	194	10	17	3.224	22	0.241	17	6.499
100	194	15	15	2.954	22	0.021	15	0.44
100	285	5	33	5.08	38	0.02	33	1.262
100	285	10	28	4.376	31	0.05	28	11.076
100	285	15	27	5.728	28	0.16	27	1.142
100	380	5	40	9.052	47	0.051	42	5.739
100	380	10	36	11.506	41	0.02	37	3.866
100	380	15	35	6.198	40	0.39	36	3.034
125	240	5	29	7.951	37	0.251	31	1.472
125	240	10	24	9.984	29	0.07	24	1.993
125	240	15	22	5.888	26	0.18	22	9.233
150	290	5	31	7.981	40	0.421	30	5.798
150	290	10	26	4.967	32	0.2	25	5.107
150	290	15	23	5.457	29	1.101	23	19.889
150	435	5	49	9.143	57	0.06	49	6.459
150	435	10	40	19.407	50	0.44	41	5.518
150	435	15	38	9.703	45	0.07	38	13.699
Total (Sum)			731	155.183	880	4.297	737	165.304

CHAPTER 3
PATH PLANNING PROBLEMS

3.1 Target Visitation Problem

3.1.1 Introduction

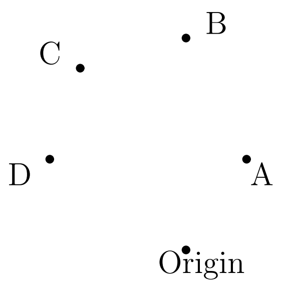
Path planning problems are among the most studied topics in operations research [37, 38, 101, 147]. In fact, the TRAVELING SALESMAN PROBLEM (TSP), arguably the most famous of all optimization problems falls in to this class [129]. In this chapter, we consider the problem of determining the optimal route for an unmanned aerial vehicle (UAV) which needs to visit multiple targets and return to its point of origin. The objective is to minimize the total distance traveled and maximize the utility of the visitation sequence. This is known as the TARGET VISITATION PROBLEM (TVP) and has several applications including combat search and rescue, disaster relief, and environmental assessment [100].

Work on the TVP is limited. It was first posed in a paper by Grundel and Jeffcoat dating from 2004 [100]. In this work, the authors describe the problem and the implementation of a greedy randomized adaptive search procedure (GRASP) for computing approximate solutions. The aforementioned paper was intended to provide an introduction to the problem, not an extensive computational analysis. Instead, the authors provide a combinatorial formulation and examine the similarities between the TVP and two other well-known problems, namely the TRAVELING SALESMAN PROBLEM [129] and the LINEAR ORDERING PROBLEM [83]. We provide a similar analysis in a later subsection.

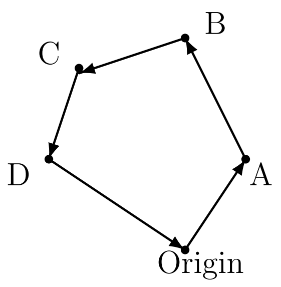
We study another closely related problem in this chapter. The problem deals with maximizing communication among agents while routing them in a cooperative network. Research in the area of cooperative networks has surged in recent years [37, 38, 101, 147]. This particular branch of telecommunications is leading the way for future technologies and the development of new network organizations [172]. In particular, so-called *mobile ad hoc networks* (MANETs) are at the forefront of the work in autonomous cooperative networks [58]. MANETs are comprised of a set of loosely coupled agents which communicate via a shared radio channel with other agents within a specified range. The unique feature of MANETs that separates them from traditional cellular networks is the fact that the topology of MANETs is dynamic. That is, with each movement of the agents, a new topology is established.

The lack of a pre-established infrastructure is an attractive feature of MANETs. They are particularly useful in situations where communication is required, but no fixed telecommunication system exists. MANETs are also helpful when a set of mobile users need to be in constant contact with each other. Specific examples include combat search and rescue teams, and medical teams. In the wake of disasters such as the terrorist attacks of September 11, 2001, and Hurricane Katrina, the nation saw first hand that communication among the emergency responders was critical to the success of the rescue operations.

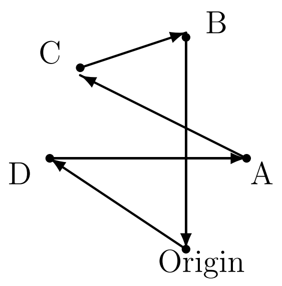
In this chapter, we describe the implementation of a genetic algorithm for finding approximate solutions for the TVP. The encoding scheme is based on random keys [23]. The heuristic is then hybridized by the implementation of a local search procedure. Numerical results are presented demonstrating the effectiveness of the proposed procedure. The remainder of the chapter is organized as follows. Next, we provide a mathematical model for the TVP and prove that finding an optimal solution is NP-complete. Then, in Section 3.1.3 we describe a hybrid genetic algorithm for solving large instances. Computational results are provided in Section 3.1.4 comparing the proposed heuristic to a standard genetic algorithm and the optimal solutions as computed by a commercial integer programming package. In Section 3.2, we study the Communication Models for a Cooperative Network of Autonomous Agents (CCPM). Section 3.2.1, we provide the problem formulation for the CCPM. Then in Section 3.2.2 we present a review of the previous work in the area of communication in cooperative networks. In Section 3.2.3, we derive two mixed integer formulations for the CCPM using the combinatorial problem



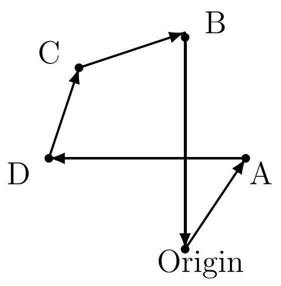
A The waypoints to be visited.



B The minimum distance solution.



C The best sequence solution.



D The optimal TVP solution.

Figure 3-1. This example compares the optimal solution for the TVP instance with the related TSP and LOP solutions.

as a guide. We provide some preliminary numerical results in Section 3.2.4 and discuss conclusions and directions of future research in Section 3.2.5.

3.1.1.1 Motivation

As technologies advance, the use of unmanned aerial vehicles (UAVs) for civilian and military applications is increasing. Civilian applications include environmental assessment and search and rescue. Moreover, UAVs have been used in military applications for decades and help to ensure that coalition forces maintain a competitive advantage in the global war on terrorism. Often times, path planning for the particular mission, be it civilian or military, is a non-trivial process. Two important, often competing factors are the overall distance traveled by the UAV and the sequence in which various “targets” or “points of interest” are visited [100]. Before moving on to the rigorous mathematical formulation, we provide a simple example demonstrating the idea behind the TVP and motivating its use in several practical military and civilian applications. In [100], the authors provide a similar example in the context of a UAV surveilling an environmental mishap, which we follow with a modification of the theme.

Suppose that the set of points in Figure 3-1(a) represent a collection of villages in which a sought after terrorist is suspected of hiding out. The point labeled “Origin” is the location of the coalition force. Moreover assume the available intelligence data suggests that the most likely hiding spot of the terrorist is point D, and the second most likely location is point A. In an application such as the one described, it is well-known that the person of interest moves frequently, and that the older the intelligence date is, the less accurate it becomes. Suppose the coalition force has the ability to launch a miniature UAV from the Origin and have it pass through a pre-established set of waypoints before returning to the starting point. During its flight, the UAV is capable of telemetering data back to the coalition force helping to establish the known location of the terrorist they seek.

The remaining subfigures demonstrate the optimal solutions when various objectives are considered. In Figure 3-1(b), only the overall distance traveled by the UAV is taken in to account. Notice here that point D, the most probable location of the terrorist is visited last. Clearly this is not a desirable visitation sequence. In Figure 3-1(c), maximizing the preferences in which the villages are visited is the only objective considered. This sequence, while better than the previous is still long and could perhaps be shortened a bit without sacrificing too much time between visiting the highly probably waypoints. The route given in Figure 3-1(d) does precisely this. Here, a combination of distance and preference is considered in the path planning. We see that this route provides the optimal mixture of visiting “high chance” waypoints quickly, so that the coalition force may act on the intelligence they receive. This simple example demonstrates the importance of considering both distance and visitation sequence when solving a path planning problem for a UAV. As we will see in the next section, when considered individually, these two objectives generalize two well-studied problems in combinatorial optimization. As it turns out, both are NP-hard which provides some indication as to the complexity of the TVP.

3.1.2 Problem Description

In this section we provide a formal description of the TARGET VISITATION PROBLEM and discuss complexity issues. We also discuss the similarities between the TVP and other combinatorial problems. What follows is a brief survey of this nature.

3.1.2.1 Related Problems

Here we briefly examine the similarities between the TARGET VISITATION PROBLEM and two well-known problems in discrete optimization, namely the TRAVELING SALESMAN PROBLEM and the LINEAR ORDERING PROBLEM.

Traveling Salesman Problem. The TRAVELING SALESMAN PROBLEM (TSP) is the most studied and widely recognized problem in operations research [51, 66, 83, 129, 159, 160, 199]. It has been the focus of research for over 50 years and remains a challenge even today. Given a graph $G = (V, E)$, a subset of edges $T \subseteq E$ is said to be a *traveling salesman tour* if it is a simple cycle of G having length $|V|$. In this context (and our's) a tour is a Hamiltonian cycle, but this easily generalizes depending on one's definition of a tour.

Suppose now that the graph is weighted (G, c), where c_{ij} represents the cost of traversing arc $(i, j) \in E$ and that $|V| = n$. The objective is now to find a tour of minimum cost. Then if we represent a tour as a permutation π of the nodes, then the goal is to find π that minimizes

$$Z(\pi) = \sum_{i=1}^{n-1} c_{\pi(i),\pi(i+1)} + c_{\pi(n),\pi(1)}. \tag{3–1}$$

The original integer programming formulation of the TSP is due to Dantzig, Fulkerson, and Johnson (DFJ) [66] and continues to be a popular formulation today. Let $x : E \mapsto \{0, 1\}$ be a decision variable associated with each arc. Then the DFJ formulation of the

TSP is given as the following 0-1 integer program [66].

$$\mathcal{DFJ}: \qquad \max \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{3–2}$$

subject to

$$\sum_{\substack{i=1\\ i\neq j}}^{n} x_{ij} = 1, \ \forall\, j \in V, \tag{3–3}$$

$$\sum_{\substack{j=1\\ j\neq i}}^{n} x_{ij} = 1, \ \forall\, i \in V, \tag{3–4}$$

$$\sum_{i\in S}\sum_{j\in S} x_{ij} \leq |S| - 1, \ \forall\, S \subset V, \ 2 \leq |S| \leq n-1, \tag{3–5}$$

$$x_{ij} \in \{0,1\}, \ \forall\, i,j \in V. \tag{3–6}$$

The DFJ formulation contains $n(n-1)$ integer variables and 2^n constraints. While minimizing the total tour cost, the sets of constraints in (3–3) and (3–4) ensure that each node is visited exactly once. Constraint set (3–5) prevents subtours, by ensuring that feasible solutions are biconnected [199]. The major drawback of the DFJ formulation is the exponential number of subtour elimination constraints.

Another formulation of the TSP that is widely used in practice is due to Miller, Tucker, and Zemlin (MTZ), and was first published in 1960 [141]. The MTZ formulation reduces the number of subtour elimination constraints to be polynomially bounded, at the expense of increasing the number of decision variables. Let $s : V \mapsto \mathbb{R}$ be a bijection where

$$s_i := \text{ the relative position of node } i \text{ in the tour.} \tag{3–7}$$

For example, if node 2 is visited third in the tour, then $s_2 = 3$. The s_i variables are commonly referred to as *sequencing variables* and can have many interpretations [69]. Without the loss of generality suppose that the depot, or point of origin is defined to be node 1. Using the decision variables defined above and the sequencing variables, we can formulate the MTZ TSP as follows:

$$\mathcal{MTZ}: \qquad \max \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{3–8}$$

subject to

$$\sum_{\substack{i=1 \\ i \neq j}}^{n} x_{ij} = 1, \ \forall\, j \in V, \tag{3–9}$$

$$\sum_{\substack{j=1 \\ j \neq i}}^{n} x_{ij} = 1, \ \forall\, i \in V, \tag{3–10}$$

$$y_i - y_j + n \cdot x_{ij} \leq n - 1, \ \forall\, i, j \in V \setminus \{1\}, i \neq j, \tag{3–11}$$

$$x_{ij} \in \{0, 1\}, \ \forall\, i, j \in V, \tag{3–12}$$

$$y_i \in \mathbb{R}, \ \forall\, i \in V \setminus \{1\}. \tag{3–13}$$

The MTZ formulation contains n^2 decision variables, and $\mathcal{O}(n^2)$ constraints. The sequencing variables used in this model allow considerable flexibility and the ability to model related problems easily. We take advantage of this for our formulation of the TVP in the following section. It has been shown however that the MTZ formulation is weaker than the DFJ formulation [158]; however, the model can be strengthened by lifting additional edges as shown by Desrochers and Laporte [69].

These represent just a few of the formulations which have been proposed for solving TRAVELING SALESMAN PROBLEMS. For extensive review and analysis of other formulations, the reader is referred to [127] and [158]. As for the computational complexity of the TSP, it is well-known to be NP-hard. Further, nonmetric instances of the TSP cannot be approximated within a constant factor unless P = NP [83]. A thorough review of the problem is available in [103].

Linear Ordering Problem. The LINEAR ORDERING PROBLEM (LOP) is another optimal sequencing problem. Given a set N of n items and a corresponding matrix $\mathbf{D} = \{d_{i,j}\}_{n \times n}$, which represents the preferences for ordering item i before item j, the objective is to find an ordering of the items which maximizes the preferences [43]. Applications abound including ranking athletes or sports teams, ranking preferences to obtain ancestry relationships, and in economics [86]. In terms of the matrix $\mathbf{D}$, the optimal solution is a permutation π of the rows and columns of $\mathbf{D}$ such that the sum of the values in the upper triangle are maximized. The LOP can represented as a combinatorial optimization problem as follows:

$$\max Z(\pi) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} d_{\pi(i),\pi(j)},$$

where $\pi(i)$ represents the item in position i of the permutation [100]. An equivalent graph theoretical problem is to find an acyclic tournament in a complete weighted graph in which the sum of the arc weights is maximal [130].

An integer programming formulation for the TVP can be computed as follows. Let $x : N \times N \mapsto \{0, 1\}$ be defined as

$$x_{ij} := \begin{cases} 1, & \text{if } i \text{ is ordered before } j, \\ 0, & \text{otherwise.} \end{cases} \tag{3–14}$$

Then the LOP admits the following integer programming formulation:

$$\mathcal{LOP}: \quad \max \sum_{(i,j)\in N} d_{ij}x_{ij} \tag{3–15}$$

subject to

$$x_{ij} + x_{ji} = 1, \ \forall\ i, j \in N, \tag{3–16}$$

$$x_{ij} + x_{jk} + x_{ki} \leq 2, \ \forall\ i, j, k \in N, i \neq j, i \neq k, j \neq k, \tag{3–17}$$

$$x_{i,j} \in \{0, 1\}, \ \forall i, j \in N. \tag{3–18}$$

The LOP formulation consists of n^2 decision variables and $\mathcal{O}(n^3)$ constraints. Constraint set (3–16) ensures that each item is considered only once, thus enforcing the strict precedence relation. The constraints in (3–17) ensure that the solution is acyclic.
Not surprisingly, the decision version of the LOP is known to be NP-complete, and the corresponding optimization problem is NP-hard [98, 99].

As we can see, the TARGET VISITATION PROBLEM combines attributes of both the TRAVELING SALESMAN PROBLEM and the LINEAR ORDERING PROBLEM. We are now ready to formally define the TVP and examine several math programming formulations.

3.1.2.2 Target Visitation Problem

An instance of the TARGET VISITATION PROBLEM consists of a set $N = \{1, 2, \ldots, n\}$ of targets located at distinct points. There is also an associated distance matrix $\mathbf{D} = \{d_{i,j}\}_{m \times m}$, where $m := n + 1$. The $d_{i,j}$ entries represent the distances between nodes $i, j \in N$. Note that the distances may be asymmetric, i.e. $d_{i,j} \neq d_{j,i}$ necessarily. Also, for all targets i, there is a value $d_{0,i}$ which represents the distance from the UAVs point of origin to target i. Furthermore, a matrix $\mathbf{R} = \{\rho_{i,j}\}_{n \times n}$ is provided where $\rho_{i,j}$ represents the preference or utility of visiting target i before target j. This can be interpreted as the assumed "threat level" or relative importance of visiting one target before another. The intuition is that targets with higher priorities should be visited earlier in the sequence.

As mentioned in [100], obtaining the values of $d_{i,j}$ is usually an easy task since literal distance measures or other metrics such as travel time are available or are trivial to calculate. However, deriving the values of $\rho_{i,j}$, the value of visiting target i before target j is not always so simple. There are several methods used by military planners when developing routes for the TVP. The most common method, and the one we adopt in this chapter, is known as "target value reconciliation" [100]. In this method a group of experts offer a set of pair-wise rankings for the targets from which the preference matrix is derived. More specifically, for all targets i and j, each expert is to specify a preference of visiting target i before j [49]. The value of $\rho_{i,j}$ is simply the cumulative number of experts who prefer to visit i before j.

A feasible solution for the TARGET VISITATION PROBLEM is one in which the UAV leaves its point of origin, visits all targets exactly once, and returns to the origin. The objective is to minimize the distance traveled while maximizing the utility of the visitation sequence [100]. Let π be a permutation of the set of integers $[1, \ldots, n+1) \cap \mathbb{Z}$, such that

$j =: \pi(i)$ implies that target j is the i^{th} position of the visitation sequence. With this, we can formulate the TVP as the following combinatorial optimization problem first given in [100]:

$$\text{Maximize } Z(\pi) = \Big[\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} \rho_{\pi(i),\pi(j)}\Big] - \Big[d_{0,\pi(1)} + \sum_{k=1}^{n-1} d_{\pi(k),\pi(k+1)} + d_{\pi(n),0}\Big] \tag{3–19}$$

Permutation based models of combinatorial problems are often useful for gaining an intuitive understanding of the problem. However, integer programming models are usually the most helpful for providing some of the formal properties of the problem [153]. With this in mind we now develop a linear integer programming formulation for the TVP.

The TARGET VISITATION PROBLEM can be conveniently described as a combinatorial problem on a graph. Consider a doubly weighted directed graph (G, d, ρ), where $V = \{0, 1, 2, \ldots, n\}$ is the set of nodes. Suppose that V represents the set of targets, hence $n = |N|$. We include an extra node which represents the origin. Also, assume without the loss of generality that G is a complete graph. For each edge $(i, j) \in E$, there is an associated weight $d_{i,j}$ which represents the distance from target i to target j. Furthermore, for each edge $(i, j) \in E$, there is an associated value $\rho_{i,j}$ which is the preference for the corresponding target pair as described above. Now, let $x : V \times V \mapsto \{0, 1\}$ be a surjection defined by

$$x_{i,j} := \begin{cases} 1, \text{ iff } i = \pi(k) \Rightarrow j = \pi(k+1), \text{ for } k \in \mathbb{Z}_n, \\ 0, \text{ otherwise,} \end{cases} \tag{3–20}$$

where π is defined as above. Said differently, $x_{i,j} = 1$ implies that $(i, j) \in E$ is a link in the tour. Next, we introduce another surjective function $w : V \times V \mapsto \{0, 1\}$ defined by

$$w_{i,j} := \begin{cases} 1, \text{ iff } i = \pi(k) \Rightarrow j = \pi(l), \text{ for } k, l \in \mathbb{Z}_n \text{ such that } k < l, \\ 0, \text{ otherwise.} \end{cases} \tag{3–21}$$

Finally, we have a bijection $y : V \mapsto \mathbb{R}$, where

$$y_i := \text{ sequence in which target } i \text{ is visited.} \tag{3–22}$$

With this we can formulate the TARGET VISITATION PROBLEM as the following integer linear program.

$$\mathcal{TVP}1: \qquad \max \sum_{i=1}^{n} \sum_{\substack{j=1 \\ i \neq j}}^{n} \rho_{i,j} w_{i,j} - \sum_{i=0}^{n} \sum_{\substack{j=0 \\ i \neq j}}^{n} d_{i,j} x_{i,j} \tag{3–23}$$

subject to

$$\sum_{\substack{i=0 \\ i \neq j}}^{n} x_{i,j} = 1, \ \forall\, j \in V, \tag{3–24}$$

$$\sum_{\substack{j=0 \\ j \neq i}}^{n} x_{i,j} = 1, \ \forall\, i \in V, \tag{3–25}$$

$$y_i - y_j + n \cdot x_{i,j} \leq n - 1, \ \forall\, i, j \in V \setminus \{0\}, i \neq j, \tag{3–26}$$

$$w_{i,j} + w_{j,i} \leq 1, \ \forall\, i, j \in V, i \neq j, \tag{3–27}$$

$$y_i - y_j + n \cdot w_{i,j} \geq 0, \ \forall\, i, j \in V \setminus \{0\}, i \neq j, \tag{3–28}$$

$$x_{i,j} \in \{0, 1\}, \ \forall\, i, j \in V, \tag{3–29}$$

$$w_{i,j} \in \{0, 1\}, \ \forall\, i, j \in V, \tag{3–30}$$

$$y_i \in \mathbb{R}, \ \forall\, i \in V \setminus \{0\}. \tag{3–31}$$

This formulation has a total of $3n^2 - 5n + 4$ constraints and $2n^2 - n - 1$ integer variables. Notice that in graph theoretical terms, the objective of the TVP is to find a Hamiltonian cycle which is of minimum weight, but which also maximizes the visitation preferences. This is accomplished by the objective function in (3–32). The $2n$ assignment constraints in (3–33) and (3–34) ensure that each target is visited only once in the tour. The $n^2 - 3n + 2$ constraints in (3–35) are subtour elimination constraints and hence prevent disjoint cycles from occurring in the tour.

The constraints in (3–27) ensure that only one of $w_{i,j}$ or $w_{j,i}$ is nonzero for all (i,j) pairs. In order to ensure that $w_{i,j} = 1$ only when i is visited before j, we have the $\mathcal{O}(n^2)$ constraints in (3–28). They insist that $w_{i,j}$ is nonzero only when $y_i < y_j$. Finally, (3–37), (3–38), and (3–39), define the proper domains for the variables used. Thus, a solution to the integer programming formulation $\mathcal{P}1$ characterizes a feasible solution to the TVP. On the other hand, it is clear that a feasible solution to the TVP will define at least one feasible solution to $\mathcal{P}1$. Therefore, $\mathcal{P}1$ is a correct formulation for the TVP.

We have shown that formulation TVP1 is correct formulation for the TVP. However, it is possible to formulate a more compact integer programming model which reduces the number of constraints by n^2. This model is based on the MTZ formulation for the TSP. It is given as follows.

$$\mathcal{TVP}2: \quad \max \sum_{i=1}^{n}\sum_{\substack{j=1\\ i\neq j}}^{n} \rho_{i,j} w_{i,j} - \sum_{i=0}^{n}\sum_{\substack{j=0\\ i\neq j}}^{n} d_{i,j} x_{i,j} \tag{3–32}$$

subject to

$$\sum_{\substack{i=0\\ i\neq j}}^{n} x_{i,j} = 1, \ \forall\, j \in V, \tag{3–33}$$

$$\sum_{\substack{j=0\\ j\neq i}}^{n} x_{i,j} = 1, \ \forall\, i \in V, \tag{3–34}$$

$$y_i - y_j + n \cdot w_{i,j} \le n - 1, \ \forall\, i,j \in V \setminus \{0\}, i \neq j, \tag{3–35}$$

$$x_{i,j} \le w_{i,j}, \ \forall\, i,j \in V, \tag{3–36}$$

$$x_{i,j} \in \{0,1\}, \ \forall\, i,j \in V, \tag{3–37}$$

$$w_{i,j} \in \{0,1\}, \ \forall\, i,j \in V, \tag{3–38}$$

$$y_i \in \mathbb{R}, \ \forall\, i \in V \setminus \{0\}. \tag{3–39}$$

Constraint sets (3–33) and (3–34) imply that the indegree and outdegree, in the edge induced subgraph of the solution set, has to be one for all the nodes. Constraint set (3–35) and (3–36) together ensures that there cannot be subtours, as this is just the MTZ based formulation for TSP. In order to prove that $w_{ij} \neq w_{ji}$, let us consider constraint set (3–35). If i is visited before j, then we have $y_i - y_j < 0$ and maximization of the objective function ensures that w_{ij} is one. If j is visited before i, then we have $y_i - y_j > 0$ and hence w_{ij} has to be zero.

The TVP represents a set of combinatorial decisions that must be made [153]. Clearly, for any asymmetric instance consisting of n targets there are $n!$ possible routes to consider. Now that we have an integer programming model for the TVP, we can examine the computational complexity. It is not surprising that finding an optimal solution is NP-hard as we will now show by proving that the recognition version of the problem is NP-complete. The recognition version of the TVP can be stated as follows: (**K-tvp**) Given an instance of the TARGET VISITATION PROBLEM, does there exist a tour of cost less than or equal to K?

Theorem 3.1. *The decision version of the* TARGET VISITATION PROBLEM *(***K-tvp***) is* NP*-complete.*

Proof. To show this, we must prove that (1) K-TVP $\in$ NP; (2) Some NP-complete problem reduces to K-TVP in polynomial time.

Clearly K-TVP $\in$ NP since any solution can be verified in polynomial time to be feasible or not.

To complete the proof, we show a simple reduction from the HAMILTONIAN CYCLE PROBLEM which is well-known to be NP-complete [83]. Let $G = (V, E)$ be a graph in which a Hamiltonian cycle has to be determined. Construct a complete graph $\bar{G} = (V, \bar{E})$ with arc distance 1 if $(i, j) \notin E$ and 0 otherwise. Furthermore, construct the preference matrix such that $\rho_{i,j} = k$, where $k \in \mathbb{R}$, for all (i, j) pairs. The objective of the decision problem K-TVP is to determine if a solution exists with cost $K \leq k \cdot n$. A 'yes' instance of the HAMILTONIAN CYCLE PROBLEM on G corresponds to a 'yes' instance for the K-TVP on $\bar{G}$. Notice that the cost of the K-TVP tour is simply the sum of the preference costs, i.e. $k \cdot n$, as the distance component of the objective function is zero.

To prove the converse, observe that the cost of any K-TVP tour is at least $k \cdot n$. Thus a 'yes' instance of K-TVP would mean that all of the arcs in the tour have zero cost. This implies all of the arcs in the K-TVP tour are present in the graph G and also form a valid tour in G. This results in a 'yes' instance for the HAMILTONIAN CYCLE PROBLEM. Thus the proof is complete. $\square$

As noted in [100], it might be the case that for a particular instance of the TVP, the terms of one of the matrices in the objective function may dominate the other. However, both distance and utility are important factors and should be given equal attention in a solution. Therefore, we use a simple balancing heuristic first given by Grundel and Jeffcoat in [100]. Let π_r be a random permutation of the targets to be visited. Further, define $\gamma \in \mathbb{R}$ such that $\tilde{\mathbf{R}} := \gamma \mathbf{R}$. In order to normalize the $\mathbf{D}$ and $\mathbf{R}$ matrices, we adjust the particular value of γ so that

$$\frac{\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \tilde{\rho}_{\pi_r(i),\pi_r(j)}}{d_{0,\pi_r}(1) + \sum_{i=1}^{n-1} d_{\pi_r(i),\pi_r(i+1)} + d_{\pi_r(n),0}} \approx 1. \tag{3–40}$$

Then without the loss of generality, the parameter γ can be used to weight either matrix if it is determined that the one of the distance or preference components is more important than the other. Hence increasing the value of γ places more importance on the utility of the visitation sequence relative to the total distance traveled [100].

The complexity of the TVP motivates the need for efficient heuristics since finding optimal solutions for large instances is impractical. Therefore, in the next section we propose the use of genetic algorithm for finding near optimal solutions for the TVP.

3.1.3 Genetic Algorithm

Genetic algorithms (GAs) get their name from the biological process which they mimic. Motivated by Darwin's Theory of Natural Selection [67], these algorithms evolve a *population* of solutions, called *individuals*, over several *generations* until the best solution is eventually reached. Each component of an individual is called a *allele*. Individuals in the population mate through a process called *crossover*, and new solutions having traits, i.e. alleles of both parents, are produced. In successive generations, only those solutions having the best *fitness* are carried to the next generation in a process which mimics the fundamental principle of natural selection, *survival of the fittest* [90]. Figure 3-2 provides pseudo-code for a standard genetic algorithm. Though the GA does converge in probability to the optimal solution, it is common to stop the procedure after some "terminating condition" (see line 3) is satisfied. This condition could be one of several things including, a maximum running time, a target objective value, or a limit on the number of generations. For our implementation, we use the latter option and the best solution after `MaxGen` generations is returned.

```
procedure GeneticAlgorithm
1    Generate population $P_k$
2    Evaluate population $P_k$
3    while terminating condition not met do
4        Select individuals from $P_k$ and copy to $P_{k+1}$
5        Crossover individuals from $P_k$ and put in $P_{k+1}$
6        Mutate individuals from $P_k$ and put in $P_{k+1}$
7        Evaluate population $P_{k+1}$
8        $P_k \leftarrow P_{k+1}$
9        $P_{k+1} \leftarrow \emptyset$
10   end while
11   return best individual in $P_k$
end procedure GeneticAlgorithm
```

Figure 3-2. Pseudo-code for generic genetic algorithm.

When designing a genetic algorithm for an optimization problem, one must provide a means to encode the population, define the crossover operator, and define the *mutation* operator which allows for random changes in offspring to help prevent the algorithm from converging prematurely. The encoding scheme we propose for our GA is based on random keys and follows exactly as described by Bean [23]. As mentioned in [23], GAs often have a difficult time maintaining feasibility of solutions in successive generations. This problem is overcome by the use of random keys as an encoding mechanism for the population. Random keys work by encoding the solution vector using random numbers. The feasibility issue is then moved into the objective function, and subsequently all offspring produced are guaranteed to be feasible solutions.

For the GA implementation for the TVP, we have the following definitions. As mentioned above, solutions are represented by a random vector. To determine the visitation sequence, a random deviate from a distribution which is uniform onto $(0, 1) \in \mathbb{R}$ is generated for each target. The tour is determined by sorting the random numbers and sequencing the targets in descending order of the sort. For example, suppose there are $n = 3$ targets to visit. Then a chromosome such as

$$(.34, .71, .28)$$

would correspond to the visitation sequence

$$2 \rightarrow 1 \rightarrow 3.$$

The objective value of the sequence can be evaluated, thus determining the fitness of the chromosome.

3.1.3.1 Evolutionary Mechanisms

In order to evolve the population over successive generations, we use a reproduction method which copies the best individuals in the current generation to the next. We aptly refer to this set the `BEST` set. This technique ensures that the best solution is monotonically improving in every generation [23]. To breed new solutions, we implement a strategy known as *parameterized uniform crossover* [186]. This method works by selecting two solutions to serve as parents. In our implementation, one parent is chosen at random from the `BEST` set, and the other is chosen from the entire population (including `BEST`). Then, for each target to be visited, a biased coin is tossed. If the result is heads, then the allele of the `BEST` parent is chosen, otherwise the allele is taken from the other parent.

The probability that the coin lands on heads is known as `CrossProb`, and is determined empirically. Figure 3-3 provides an example of a potential crossover when the number of targets is 5 and `CrossProb` = 0.65.

Coin Toss	T	H	H	T	H
Parent 1	0.56	0.81	0.22	0.7	0.86
Parent 2	0.29	0.49	0.98	0.12	0.32
Offspring	0.29	0.81	0.22	0.12	0.86

Figure 3-3. An example of the crossover operation. In this case, `CrossProb` = 0.65.

Finally, the mutation operator is defined as follows. Instead of introducing random perturbations to selected offspring, we instead replace a set of individuals having the worst fitness with new solutions generated at random from the same distribution as the original population. This replacement set is referred to as the `WORST` set. Using this method, we are able to ensure that the GA does not converge prematurely. This is a common method, sometimes referred to as *immigration* and appears throughout the literature [23, 94]. An overall pictorial view of the generational evolution of the proposed GA is provided in Figure 3-4.

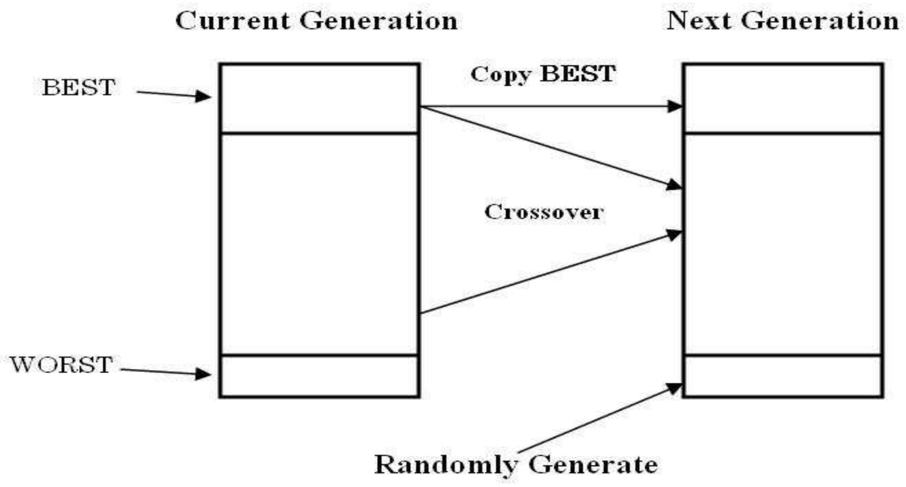


Figure 3-4. Graphical representation of generational evolution.

3.1.3.2 Local Search

In addition to the standard GA, we propose a hybridization technique to produce better solutions. In particular we implement a 2-exchange local search on each of the offspring produced by crossover operator. Pseudo-code for this heuristic is provided in Figure 3-5. A 2-exchange local search is a hill-climbing procedure which examines pairs of alleles and performs a swap. If the resulting swap increases the fitness of the individual, the swap is kept. Otherwise, it is undone an another pair is examined. Such local improvement methods abound in the literature and are used to enhance methods such as greedy randomized adaptive search procedures (GRASP) [173], tabu search [85], and other combinatorial optimization heuristics [1].

```
procedure LocalSearch($X$)
  $X^* \leftarrow X$
  $f(X^*) \leftarrow f(X)$
  temp $\leftarrow 0$
  while $X$ is not locally optimal do
    for $i = 1$ to $|X|$ do
      for $j = 1$ to $|X|$ do
        temp $\leftarrow X(i)$
        $X(i) \leftarrow X(j)$
        $X(j) \leftarrow$ temp
        if $f(X) > f(X^*)$ then
          $X^* \leftarrow X$
        else
          temp $\leftarrow X(i)$ /* undo swap */
          $X(i) \leftarrow X(j)$
          $X(j) \leftarrow$ temp
        end if
      end for
    end for
  end while
  return ($X^*$)
end procedure LocalSearch
```

Figure 3-5. 2-exchange local search.

3.1.4 Computational Results

The proposed heuristic was implemented in the C++ programming language and complied using GNU g++ version 3.4.4, using optimization flags -O2. It was tested on a PC equipped with a 1700MHz Intel® Pentium® M processor and 1.0 gigabytes of RAM operating under the Microsoft® Windows® XP Professional environment.

As a basis for comparison, we examine the results for the hybrid genetic algorithm (HGA) with the standard genetic algorithm (GA). In addition, we have implemented the integer programming model for the TARGET VISITATION PROBLEM using the CPLEX™ version 10 optimization suite from ILOG [63]. CPLEX contains an implementation of the simplex method [110], and uses a branch and bound algorithm [199] together with advanced cutting-plane techniques [118, 154]. The instances were tested using the CPLEX default settings. The algorithms were tested on a set of randomly generated instances varying in size from 8-16 targets* . Due to the complexity of the problem, CPLEX was unable to obtain optimal solutions for instances with $|N| > 16$. For each instance, the number of "experts" used to derive the utility matrix is 10. Also, the matrices were balanced using the heuristic described in Equation (3–40) above. For each instance, the maximum distance between the targets varied from 20 to 150 units. The distance matrix for each instance was generated uniformly at random with some user defined upper and lower bounds. The instances were created using a simple problem generator written in C++. The instances used in this chapter are non-symmetric and non-metric; however, this option is built in to the generator. It is a reasonable assumption to have non-symmetric instances, since in a real-world scenario the matrix $\mathbf{D}$ might represent other factors than simply the distance between two targets such as an extra cost or risk associated with visiting a particular target. In this case $d_{ij} \neq d_{ji}$ necessarily. Furthermore, it is assumed that the UAV is capable of traveling all cycles in the graph. Depending on the individual

* The test problems may be downloaded from http://plaza.ufl.edu/clayton8/tvp.tar.gz.

application and factors such as the number of targets and the size of the battlespace, it may be reasonable to impose a hard constraint on the total distance traveled in the TVP tour. This would model problem scenarios when battery consumption and fuel capacity are critical. In the instances tested we do not impose such constraints; however doing so would most likely reduce the overall CPLEX running time.

We mention here that in the instances considered, the priority functions have been normalized by the distance function in order to avoid the domination of one cost function by the other. This justifies the cost of objective function. For a better realization of this fact, we made test runs on the instances solving both the TSP and LOP problems separately. These results are included in Table 3-1. To further illustrate this, Table 3-2 provides the objective function value of the LOP, TVP, and TSP when one objective is considered and solved to optimality. The LOP solution (non-optimal component) corresponding to the TSP optimal route (optimal component) and the TSP solution (non-optimal component) corresponding to the optimal LOP route (optimal component) have been presented. The non-optimal components are weaker when compared to each component in the optimal TVP solution. This numerical evidence supports the claim that in order to find high quality TVP routes, it is not advisable to decouple the problem into simply a TRAVELING SALESMAN PROBLEM or LINEAR ORDERING PROBLEM, but rather to consider both objectives in concert.

The alternative way of solving a biobjective optimization problem is by determining a set of efficient solutions. An solution is efficient if both the cost functions are dominated by any other solution [161]. This is achieved by treating the problem with a single objective function and recursively solving the problem by recomputing the new costs based on the cost obtained from the previous iteration. This however assumes that the both the cost functions involves with the same problem. In our case, we have one cost function to be employed for a TSP problem and another cost function for a LOP problem and an efficient algorithm for solving both the problems has to be designed for this purpose, which is beyond the scope of this chapter.

Table 3-1. Comparative results of the optimal solutions to the corresponding TSP, LOP, and TVP for each instance. The absolute value of the TSP solutions are reported.

Instance		LOP	TVP	TSP
Name	Targets	Optimal Solution	Optimal Solution	Optimal Solution
`rand8-1`	8	110.085	60.2766	31
`rand8-2`	8	197.139	115.944	55
`rand8-3`	8	335.74	195.333	76
`rand8-4`	8	59.2222	29.0074	20
`rand8-5`	8	646.16	314	221
`rand10-1`	10	259.926	157.404	74
`rand10-2`	10	247	208	21
`rand10-3`	10	734.569	520.679	149
`rand10-4`	10	720.167	532.5	97
`rand10-5`	10	565.781	365.125	105
`rand12-1`	12	208.29	124.179	56
`rand12-2`	12	491.677	318.38	104
`rand12-3`	12	646.772	420.959	142
`rand12-4`	12	944.093	594.546	169
`rand12-5`	12	640.055	472.354	89
`rand14-1`	14	204.414	137.609	39
`rand14-2`	14	549.708	405.774	72
`rand14-3`	14	897.804	631.711	153
`rand14-4`	14	292.389	176.631	65
`rand14-5`	14	976.921	679.625	131
`rand16-1`	16	518.62	381.934	63
`rand16-2`	16	706.98	431.531	164
`rand16-3`	16	571.735	415.339	99
`rand16-4`	16	707.22	421.658	162
`rand16-5`	16	364.152	249.939	68

3.1.4.1 Numerical Results

We begin by presenting some comparative results of the heuristics. As Gonçalves et al. correctly indicate, despite the massive amounts of literature on genetic algorithms, there is little knowledge of how best to tune the parameters for a given application [95]. For all of the instances tested, the parameters used for the genetic algorithm (GA) and

Table 3-2. The corresponding objective function values of each of the LOP, TSP, and TVP are given for each instance. For each column, one of the objectives is considered and the problem solved to optimality. The solution of the remaining two problems is given when evaluated with the optimal function value.

Instance		LOP opt		TVP opt		TSP opt	
Name	Targets	TSP	LOP	TSP	LOP	TSP	LOP
rand8-1	8	-100	110.085	-31	91.2766	-31	91.2766
rand8-2	8	-174	197.139	-77	192.944	-55	161.486
rand8-3	8	-242	335.74	-115	310.333	-76	219.592
rand8-4	8	-39	59.222	-29	58.0075	-20	44.6444
rand8-5	8	-502	646.16	-221	574	-221	501.84
rand10-1	10	-139	259.926	-94	251.404	-74	203.68
rand10-2	10	-55	247	-37	245	-21	195
rand10-3	10	-648	734.569	-199	719.679	-149	575.743
rand10-4	10	-401	720.167	-120	652.5	-97	580
rand10-5	10	-344	565.781	-123	488.125	-105	391.979
rand12-1	12	-184	208.29	-64	188.179	-56	170.941
rand12-2	12	-328	491.677	-110	428.38	-104	392.211
rand12-3	12	-379	646.772	-149	569.959	-142	514.653
rand12-4	12	-721	944.093	-279	873.546	-169	674.352
rand12-5	12	-513	640.055	-138	610.354	-89	512.341
rand14-1	14	-150	204.414	-48	185.609	-39	170.287
rand14-2	14	-336	549.708	-84	489.774	-72	427.967
rand14-3	14	-707	897.804	-157	788.711	-153	670.773
rand14-4	14	-280	292.389	-93	269.631	-65	195.421
rand14-5	14	-774	976.921	-176	855.625	-131	708.104
rand16-1	16	-399	518.62	-92	473.934	-63	421.125
rand16-2	16	-671	706.98	-185	616.934	-164	564.846
rand16-3	16	-392	571.735	-110	525.338	-99	457.987
rand16-4	16	-693	707.22	-178	599.658	-162	565.597
rand16-5	16	-290	364.152	-83	332.939	-68	288.484

the hybrid genetic algorithm (HGA) are given in Table 3.1.4.1. It has been shown in the literature that parameters similar to those we implemented have been effective when implementing a hybrid GA [32, 93–95]. Table 3-4 presents the comparative results of the HGA and standard GA applied to 25 randomly generated instances. The number of targets ranges from 8 to 16. Five instances were tested for each value of $|N|$. The table provides the instance name and the corresponding number of targets. Next the optimal solutions are provided along with the corresponding computation time required by

Table 3-3. Parameters used for the GA and HGA heuristics.

$\texttt{CrossProb} = 0.7$	Population Size ($\texttt{PopSize}$) $= 2 * \lvert N \rvert$
$\texttt{MaxGen} = 10000$	$\lvert\texttt{BEST}\rvert = .1 * \texttt{PopSize}$
$\lvert\texttt{WORST}\rvert = .2 * \texttt{PopSize}$	

CPLEX. We tested each of the heuristics 250 times on each instance, and we provide the maximum, minimum, and average solutions computed for each. Finally for both heuristics, we provide the average computation time for the 250 runs as well as the average deviation from the optimum.

Notice that for all 6250 experiments, the hybrid GA computed optimal solutions 99.93% of the time requiring 2.687 seconds of computation time on average. The compares favorably with the average time required by CPLEX to compute the optimal solutions which was 601.428 seconds. We note here that a heuristic solution was used as a starting point for the computation of the optimal solutions for the instances containing 16 targets. Without this, the running time for CPLEX for these problems was on the order of 75000 seconds. We see also that the standard genetic algorithm performed reasonably well, with an average optimality gap of 4.429%. In addition, the standard GA scaled well averaging less than one half second of computation time for all instances. However, we see that ultimately the hybridized algorithm was the most robust of the two methods. For the HGA, the increase is the average solution time for the large instances is arguably offset by its stellar performance. To the contrary, one might argue that given more time the performance of the standard GA would match that of the hybrid method. However, in the next subsection we perform a probabilistic analysis on the time required for each heuristic to compute a target value, and we shall see that this argument is ultimately untrue.

3.1.4.2 Time-to-Target Plots

In this subsection, we investigate the empirical distributions of the heuristic running times. It has been observed [6, 21, 72] that solution times for stochastic heuristics such

Table 3-4. This table provides the numerical results for a set of randomly generated instances. The first columns provide information about the instance. Next, the optimal solution and required computation time is listed. Both the HGA and the standard GA were ran 250 times on each instance, and we provide the maximum, minimum, and average solutions computed by each for all 250 tests. The average computation time required by each heuristic to compute the best solution is also listed.

Instance		IP Model		Hybrid GA					Standard GA				
Name	Targets	Optimal Solution	Execution Time (s)	Max Soln	Min Soln	Avg. Soln	Avg. Time (s)	Avg. Dev (%)	Max Soln	Min Soln	Avg. Soln	Avg. Time (s)	Avg. Dev (%)
rand8-1	8	60.2766	0.01	60.2766	60.2766	60.2766	0.005	-	60.2766	56.3404	59.8895	0.054	0.642
rand8-2	8	115.944	0.02	115.944	115.944	115.944	0.047	-	115.944	112.653	115.681	0.044	0.27
rand8-3	8	195.333	0.01	195.333	195.333	195.333	0.011	-	195.333	194.96	188.555	0.032	5.006
rand8-4	8	29.0074	0.02	29.0074	29.0074	29.0074	0.027	-	29.0074	25.8592	28.888	0.052	0.412
rand8-5	8	314	0.03	314	314	314	0.001	-	314	314	314	0.008	-
rand10-1	10	157.404	3.01	157.404	157.404	157.404	0.111	-	157.404	140.133	140.133	0.123	10.972
rand10-2	10	208	2.87	208	204	207.984	0.307	0.008	208	200	207.36	0.044	0.308
rand10-3	10	520.679	0.01	520.679	520.679	520.679	0.092	-	520.679	437.12	518.698	0.049	0.380
rand10-4	10	532.5	2.45	532.5	532.5	532.5	0.059	-	532.5	489.667	529.891	0.107	0.49
rand10-5	10	365.125	4.87	365.125	365.125	365.125	0.034	-	365.125	303.615	349.457	0.068	4.291
rand12-1	12	124.179	45.37	124.179	124.179	124.179	0.129	-	124.179	106.645	121.022	0.148	2.54
rand12-2	12	318.38	61.17	318.38	318.38	318.38	0.039	-	318.38	266.641	308.668	0.128	3.050
rand12-3	12	420.959	51.89	420.959	420.959	420.959	0.191	-	420.959	341.866	403.31	0.0951	4.193
rand12-4	12	594.546	16.44	594.546	594.546	594.546	0.427	-	594.546	487.099	580.956	0.137	2.286
rand12-5	12	472.354	14.68	472.354	472.354	472.354	0.305	-	472.354	409.102	456.735	0.131	3.307
rand14-1	14	137.609	303.55	137.609	137.609	137.609	0.792	-	137.609	110.948	128.208	0.225	6.832
rand14-2	14	405.774	370.01	405.774	397.503	405.741	2.128	0.008	405.774	334.807	383.21	0.29	5.561
rand14-3	14	631.711	184.82	631.711	614.917	631.644	2.795	0.011	631.711	508.412	594.765	0.267	5.849
rand14-4	14	176.631	301.71	176.631	172.789	176.603	3.148	0.016	176.631	146.979	164.377	0.250	6.938
rand14-5	14	679.625	2700.78	679.625	679.625	679.625	1.546	-	679.625	530.617	638.161	0.225	6.101
rand16-1	16	381.934	1353.77	381.934	376.039	381.62	8.954	0.082	381.934	298.207	351.275	0.351	8.027
rand16-2	16	431.531	2556.77	431.531	414.606	428.75	10.082	0.645	431.531	333	387.659	0.322	10.167
rand16-3	16	415.338	462.5	415.338	408.324	415.074	13.358	0.064	415.338	332.868	380.319	0.329	8.431
rand16-4	16	421.658	2810.45	421.658	409.171	419.305	12.903	0.558	417.109	314.109	386.355	0.397	8.372
rand16-5	16	249.939	3788.49	249.939	243.534	249.099	9.676	0.336	249.939	187.592	234.192	0.326	6.3

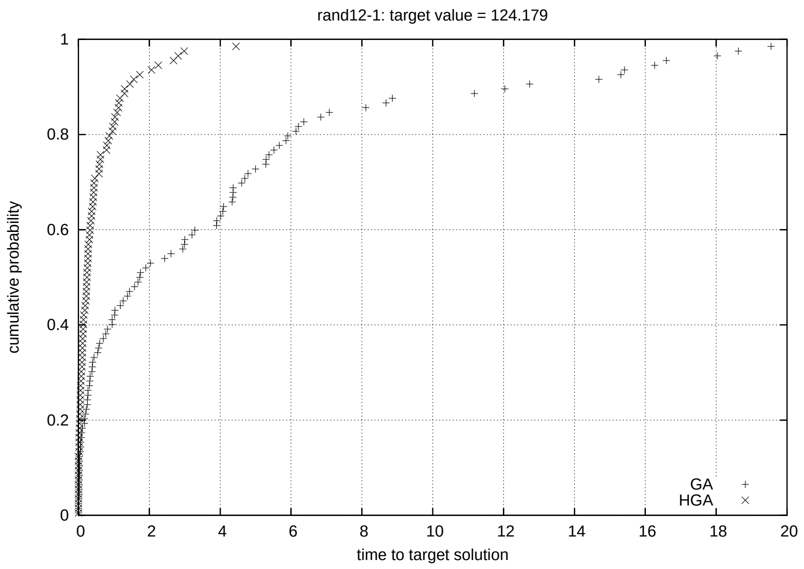
as genetic algorithms, tabu search, and GRASP fit a two-parameter shifted exponential distribution [7]. More specifically, let $P : \mathbb{R} \mapsto [0,1]$ be a probability measure on a Borel set. Then the probability of *not* finding a target solution in t time units is given by $P(t) := e^{-(t-\mu)/\lambda}$, where $\lambda \in \mathbb{R}^+$ and $\mu \in \mathbb{R}$.

For each instance, we make 100 runs of both the hybrid GA and the standard GA. The runs are assumed to be independent since each was done using a different seed for the random number generator. For each instance/heuristic pair, the algorithms ran until they calculated a target solution and the required computation time was recorded. Then for each instance, the running times for each heuristic was sorted in descending order. The i^{th} sorted running time, t_i is associated with the probability $p_i := (i - \frac{1}{2})/100$ and the point $z_i = (p_i, t_i)$, for all $i = 1, \ldots, 100$ [72]. The z_i points were then plotted in what is known as a *Time-to-Target Plot* (TTTplot) using the publicly available `perl` software `tttplots`[†] by Aiex, Resende, and Ribeiro [7].

Since it is unreasonable to provide a TTTplot for each instance tested, instead we provide a representative subset all the instances in Figures 3-6 - 3-8. Notice that for all cases the hybrid GA converges faster than the standard GA which is visualized by the fact that the HGA curves are completely to the left of the standard GA curves in all the TTTplots. The TTTplots imply that for a fixed amount of time, the hybrid method has a higher probability of reaching the target solution. For example, consider the plot for instance `rand12-1` shown in Figure 3-6. We see that given one second of computing time, the probability that the hybrid GA will compute the optimal solution is 0.926, compared with a probability of 0.443 for the standard GA. Likewise, for a fixed probability, the plot indicates that the hybrid method will find the target solution quicker

[†] Available at `http://www.research.att.com/~mgcr/tttplots/`.

than the simple genetic algorithm. Consider the TTTplot for instance `rand14-2` in Figure 3-7. In order for the GA to compute the target solution for a fixed probability of 0.6, approximately 10.6 seconds of computation time are required. The hybrid GA requires 2.32 seconds to find the target solution with 60% success. The TTTplots particularly highlight the scalability and robustness of the hybrid genetic algorithm when tested on larger instances as indicated by the near vertical plots of the HGA probabilities in Figure 3-8. The main point to be made here is not to argue that the hybrid genetic algorithm outperforms the standard metaheuristic in terms of objection function value. Of course, it has been shown that the standard genetic algorithm will converge to the optimal solution with probability 1. What we have demonstrated is that by enhancing the metaheuristic with the application of a local search, we are able to dramatically decrease the computation time required to converge to the optimal solution. We can conclude that for very large-scale instances of combinatorial problems such as the TVP, the advantage of using the hybrid GA enables us to find high quality solutions much faster than the basic genetic algorithm. For problems involving military applications such as the TVP time is usually critical. Time spent searching for a good solution can lead to the loss of equipment or the death of personnel in the battlefield. In these cases, the advantage of the hybrid method is clear. The quicker a solution can be computed, the faster the system can be deployed and the competitive advantage is retained on the battlefield.



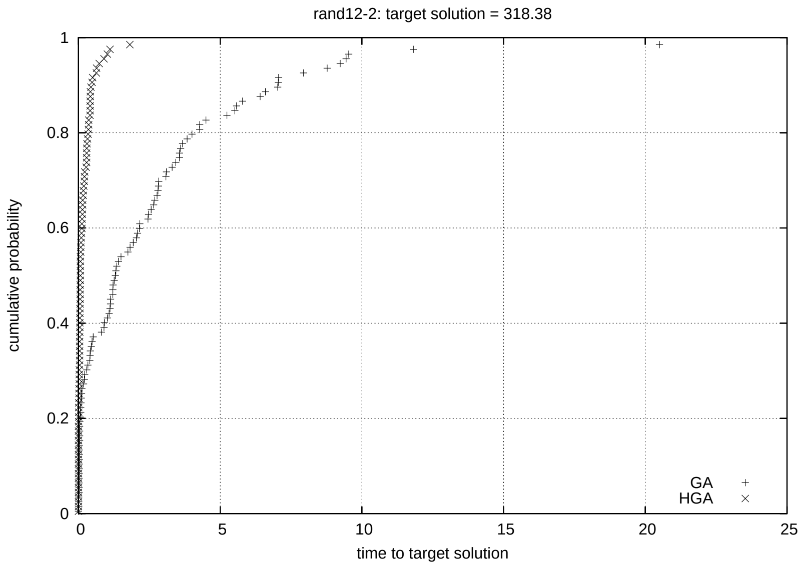
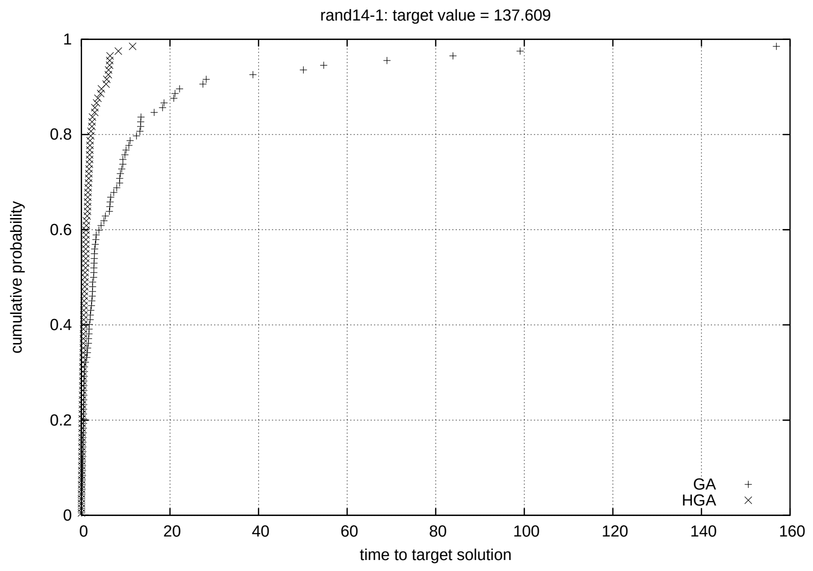


Figure 3-6. Time-to-Target plot comparing the Hybrid GA and standard GA for instance `rand12-1`. The target value is the optimal solution for the problem.



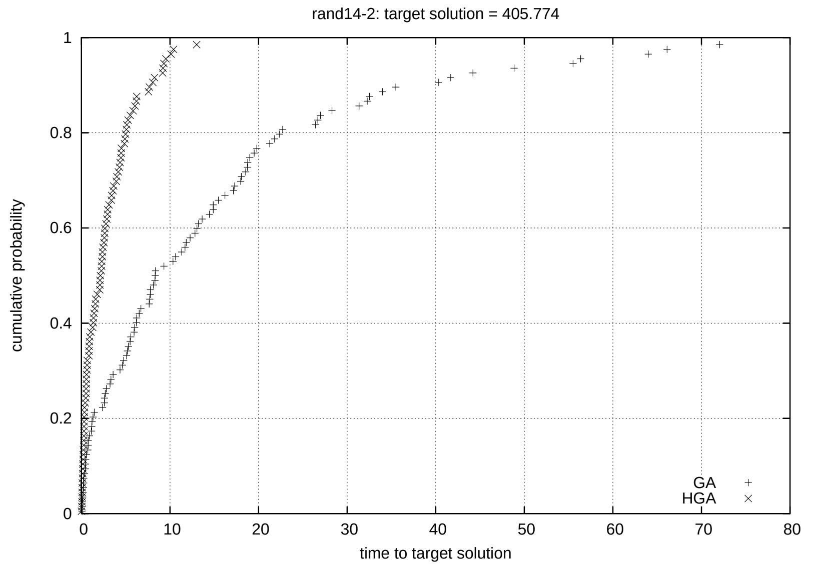
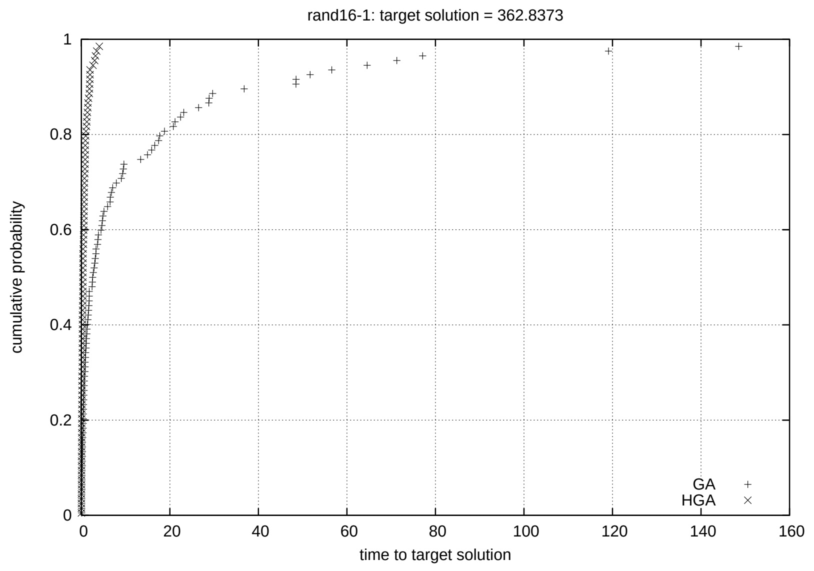


Figure 3-7. Time-to-Target plot for instance `rand14-2`. As above, the target values is the optimal solution.



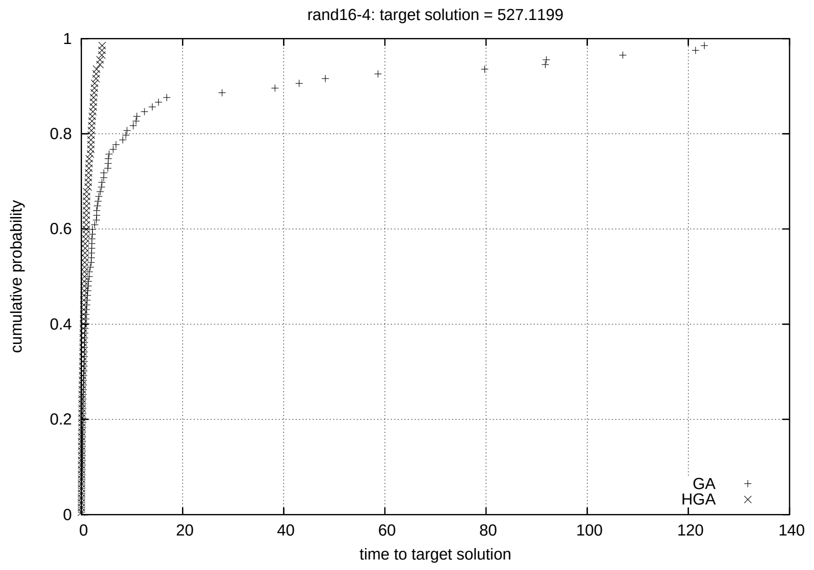


Figure 3-8. Time-to-Target plot for instances `rand16-1`. The target value is .95 times the optimal solution.

3.2 Communication Models for a Cooperative Network of Autonomous Agents

Most cooperative networks require coordination among the group of users in order to accomplish the objective. The coordination of the system usually depends on communication being guaranteed amongst the agents. In typical ad hoc networks, bandwidth and communication time are very limited resources. Therefore, we see that the lack of a central command center for MANETs, while appealing from a distributed perspective, does lead to several problems in terms of routing, communication, and path-planning [36, 56]. Perhaps the most important among these, and the focus of this chapter, is the study of communication models in the network. In particular, we study the problem of coordinating a set of wireless agents involved in a task that requires them to travel from a source location to a destination. The objective is to determine the paths, or trajectories for the agents which maximizes the connectivity between them subject to constraints on the initial and final configurations, and several limitations on the movements of the agents [152]. This problem is known as the COOPERATIVE COMMUNICATION PROBLEM IN MOBILE AD HOC NETWORKS (CCPM), and is known to be NP-hard [83]. In the next section, we review the current work on the CCPM, which is primarily focused on heuristics for the problem posed as a discrete optimization problem.

3.2.1 Problem Formulation

Consider an undirected graph $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$ represents the set of available positions for the wireless agents. Each node in V is assumed to be connected only to nodes that can be reached in one time step. Also, define $N(v) \subseteq 2^V$, for $v \in V$, to represent the set of neighbors, or nodes, which are adjacent to node v. Let U represent the set of agents, $S = \{s_1, s_2, \ldots, s_{|U|}\} \subseteq V$ the set of initial positions, and $D = \{d_1, d_2, \ldots, d_{|U|}\} \subseteq V$ the set of destination positions for the agents. Given a time horizon T, the objective of the problem is to determine a set of routes for the agents, such that each agent $u_i \in U$ starts at source node s_i and finishes at its respective destination node d_i after at most T units of time [58].

For each agent $u \in U$, the function $p_t : U \to V$ returns the position of the agent at time $t \in \{1, 2, \ldots, T\}$. Then at each time instant t, an agent $u \in U$ can either remain in its current location, i.e. $p_{t-1}(u)$, or move to a node in $N(p_{t-1}(u))$.

We can represent a route for an agent $u \in U$ as a path $\mathcal{P} = \{v_1, v_2, \ldots, v_k\} \subseteq V$ where $v_1 = s_u$, $v_k = d_u$, and, for $i \in \{2, \ldots, k\}$, $v_i \in N(v_{i-1}) \cup \{v_i\}$. Finally, if $\{\mathcal{P}_i\}_{i=1}^{|U|}$ is the set of trajectories for the agents, we are given a corresponding vector $\mathcal{L}$ such that $\mathcal{L}_i$ is a threshold on the size of path $\mathcal{P}_i$. This value is typically determined by fuel or battery life constraints on the wireless agents.

We assume that the agents have omnidirectional antennas and that two agents in the network are connected if the distance between them is less than some radius $r \in \mathbb{R}$. The particular value of r is determined by the capabilities of the wireless equipment, such as the antenna strength and power amplifier. More specifically, let $\delta : V \times V \to \mathbb{R}$ represent the Euclidean distance between a pair of nodes in the graph. Then, we can define a function $c : V \times V \to \{0, 1\}$ such that

$$c(p_t(u_i), p_t(u_j)) = \begin{cases} 1, & \text{if } \delta(p_t(u_i), p_t(u_j)) \leq r \\ 0, & \text{otherwise.} \end{cases} \tag{3–41}$$

With this, we can define the CCPM as the following optimization problem as given by Commander et al. [58]:

$$\max \quad \sum_{t=1}^{T} \sum_{u,v \in U} c(p_t(u), p_t(v)) \tag{3–42}$$

$$\text{s.t.} \quad \sum_{j=2}^{n_i} \delta(v_{j-1}, v_j) \leq \mathcal{L}_i, \ \forall \ \mathcal{P}_i = \{v_1, v_2, \dots, v_{n_i}\} \tag{3–43}$$

$$p_1(u) = s_u \quad \forall \ u \in U \tag{3–44}$$

$$p_T(u) = d_u \quad \forall \ u \in U, \tag{3–45}$$

where constraint (3–43) ensures that the length of each path $\mathcal{P}_i$ is less than or equal to its maximum allowed length $\mathcal{L}_i$.

It has been determined [152] that the problem described above is NP-hard. This can be shown by a reduction from the well known 3SAT problem. Moreover, it is NP-hard even to find an optimal solution for one stage of the problem at a given time t. To see this, consider an algorithm that maximizes the number of connections at time t, by defining the positions for members of the network; clearly the algorithm just described computes the value of the maximum clique on the underlying unit graph [57]. Running this algorithm for different sets U^i, with $|U^i| = i$ and i varying from 1 to T, the algorithm stops when the number of connections is less then $\binom{i}{2}$, and the value returned is $i - 1$. Computing optimal solutions for the MAXIMUM CLIQUE PROBLEM on a unit graph is known to be NP-hard [53].

Due to the computational complexity of the problem, real-world instances cannot be solved exactly. Therefore, we turn our attention to the design and implementation of efficient heuristics to solve large-scale instances within reasonable computing times. In the following section, we review the recent work in this area and describe the implementation of the first advanced metaheuristic for the CCPM based on the Greedy Randomized Adaptive Search Procedure [173].

3.2.2 Previous Work

Since the introduction of the CCPM by Oliveira and Pardalos [152], heuristic design has been a major focus [57, 58]. In [58], the authors introduced a construction heuristic for the CCPM based on shortest paths [5]. The goal was to provide a way to quickly calculate sets of feasible trajectories for the agents. Pseudo-code for this algorithm is provided in Figure 3-9. The procedure takes as input an instance of the CCPM consisting of the graph G, the set of agents U, source nodes S, destination nodes D, and a maximum travel time T. The total number of connections (`c`) represents the value of the objective function from equation (3–42) and is initialized to zero. The set of trajectories for the agents (`solution`) is initialized to the empty set. In line 3, we compute the shortest source-destination path for each agent using the Floyd-Warshal algorithm [75, 195]. For each agent $i \in U$, the corresponding shortest path is assigned as the trajectory $\mathcal{P}_i$ for the agent. The trajectory is feasible if agent i is able to reach its destination d_i in at most T time units. Any agent which reaches its target location in less than T time steps will remain there until all other agents reach their respective destinations. If any path is infeasible, the algorithm terminates. Otherwise, the number of connections is updated and the process repeats until all agents have been considered.

The aforementioned algorithm provides feasible solutions for instances of the CCPM in $\mathcal{O}(|V|^3)$ time. However, the trajectories calculated are not guaranteed to be locally optimal, let alone globally optimal. Therefore, a local neighborhood search enhancement was applied. In general, a local search method receives a feasible solution as input and returns a solution that is no worse than the one input, with respect to the given neighborhood structure. Finding a locally optimal solution in the local search phase

```
procedure ShortestPath($G, U, S, D, T$)
1   c $\leftarrow 0$
2   solution $\leftarrow \emptyset$
3   Compute all shortest paths SP($s_i, d_i$) for each pair $(s_i, d_i) \in S \times D$
4   for $i = 1$ to $|U|$ do
5     $\mathcal{P}_i \leftarrow$ SP($s_i, d_i$)
6     if length of $\mathcal{P}_i > T$ then
7       return $\emptyset$
8     else
9       solution $\leftarrow$ solution $\cup\ \mathcal{P}_i$
10      c $\leftarrow$ c + new connections generated by $\mathcal{P}_i$
11    end
12  end
13  return (c, solution)
end procedure ShortestPath
```

Figure 3-9. Pseudo-code for the shortest-path construction heuristic.

depends, among other things, on the actual structure and density of the neighborhood. Let $\mathcal{S}$ be the set of feasible solutions for an instance Π of the CCPM. Then for some $s \in \mathcal{S}$ the neighborhood of s, denoted $\mathcal{N}(s)$, can be defined as the set of all solutions $\bar{s} \in \mathcal{S}$ that differ from s in exactly one route. Notice that the number of feasible paths between any source-destination pair is exponential, and could lead to unreasonable computation times. Therefore instead of exhaustively searching the entire neighborhood the authors probe only $|U|$ neighbors at each iteration (one for each source-destination pair). Also, because of the exponential size of the neighborhood, the maximum number of iterations performed was limited to a constant `MaxIter`.

Pseudo-code for the local improvement heuristic can be seen in Figure 3-10. Let f represent the objective function for the CCPM as given in equation (3–42) above. New routes are computed using a randomized version of the standard depth-first search (DFS) [5]. As mentioned in [58], at each step of the randomized DFS, the node selected to explore is uniformly chosen among the available children of the current node. Randomization helps to find a route that may improve the solution, while avoiding being trapped at a local optimum after only a few iterations. The local search is a standard hill-climbing method [118]. Beginning with the feasible solution from the shortest path constructor, the local search begins computing new trajectories for the agents by using the randomized DFS to explore the neighborhood as described above. The method iterates over all the agents and repeats a total of `MaxIter` iterations after which the current best solution is deemed locally optimal and returned.

```
procedure HillClimb(solution)
    c ← $f$(solution)
    while solution not locally optimal and iter < MaxIter do
        for $i = 1$ to $|U|$ do
            solution ← solution \ $\{\mathcal{P}_i\}$
            $\bar{\mathcal{P}}_i$ ← DFS($s_i, d_i$)
            c' ← $f$(solution ∪ $\bar{\mathcal{P}}_i$)
            if length of $\bar{\mathcal{P}}_i < T$ and c' > c then
                c ← c'
                iter ← 0
            else
                Restore path $\mathcal{P}_i$
            end
        end for
        iter ← iter+1
    end while
    return (solution)
end procedure HillClimb
```

Figure 3-10. Pseudo-code for the Hill Climbing intensification procedure.

We now describe the implementation of a more advanced randomized multi-start heuristic for the CCPM based on the Greedy Randomized Adaptive Search Procedure (GRASP) [173] framework. GRASP is a two-phase metaheuristic for combinatorial optimization that aims to find very good solutions though the controlled use of random sampling, greedy selection, and local search. GRASP has been used extensively in the last decade on numerous optimization problems and produces excellent results in practice [73]. Let F be the set of feasible solutions for the a problem Π, where each solution $S \in F$

```
procedure OnePass($G, U, S, D, T$)
    solution $\leftarrow$ ShortestPath($G, U, S, D, T$)
    solution $\leftarrow$ HillClimb(solution)
    return (solution)
end procedure OnePass
```

Figure 3-11. Pseudo-code for the one-pass heuristic.

is composed of k discrete components $a_1, \ldots, a_k$. GRASP constructs a sequence $\{S\}_i$ of solutions for Π, such that each S_i is feasible for Π. At the end, the algorithm returns the best solution found.

```
procedure GRASP(MaxIter)
    $X^* \leftarrow \emptyset$
    for $i = 1$ to MaxIter do
        $X \leftarrow$ ConstructionSolution($G, g, X$)
        $X \leftarrow$ LocalSearch($X$, MaxIterLS)
        if $f(X) \geq f(X^*)$ then
            $X^* \leftarrow X$
        end
    end
    return $X^*$
end procedure GRASP
```

Figure 3-12. GRASP for maximization.

Pseudo-code for the GRASP is provided in Figure 3-12. Notice that each GRASP solution is built in two stages, called *greedy randomized construction* and *intensification* phases. The construction phase receives as parameters an instance of the problem, a ranking function $g : A(S) \rightarrow R$ (where $A(S)$ is the domain of feasible components $a_1, \ldots, a_k$ for a partial solution S), and a constant $0 < \alpha < 1$. It starts with an empty partial solution S. Assuming that $|A(S)| = l$, the algorithm creates a list of the best ranked αl components in $A(S)$, and returns a uniformly chosen element x from this list. The current partial solution is augmented to include x, and the procedure is repeated until the solution is feasible, i.e. until $S \in F$.

The intensification phase consists of the implementation of a hill-climbing procedure. Given a solution $S \in F$, let $N(S)$ be the set of solutions that can found from S by changing one of the components $a \in S$. Then, $N(S)$ is called a neighborhood of S. The improvement algorithm consists of finding, at each step, the element S^* such that

$$S^* = \text{argmax}_{\bar{s} \in N(S)} f(\bar{s}),$$

where $f : F \to R$ is the objective function of the problem. At the end of each step, we assign $S \leftarrow S^*$ if $f(S) > f(S^*)$. The algorithm will converge to a local optimum, in which case the procedure above will generate a solution S^* such that $f(S^*) \geq f(S)$ for each $S \in N(S^*)$.

To apply GRASP to the CCPM, we need to specify the set A, the greedy function g, the parameter α, and the neighborhood $N(S)$, for $S \in F$. The components of each solution S are feasible moves of a member of the ad hoc network from a node v to a node $w \in N(v) \cup \{v\}$. The complete solution is constructed according to the following procedure. Start with a random $u \in U$ and find the shortest path P from s_u to d_u. If the total distance of P is greater than D_u, then the instance is clearly infeasible, and the algorithm ends. Otherwise, the algorithm considers each feasible move. A feasible move connects the final node of a sub-path P_v, for $v \in U \setminus \{u\}$, to another node w, such that the shortest path from w to d_v has distance at most $D_v - \sum_{e \in P_v} \text{dist}(e)$. The set of all feasible moves in a solution is defined as $A(S)$.

The greedy function g returns for each move in $A(S)$ the number of additional connections created by that move. As described above, the construction procedure will rank the elements of $A(S)$ according to g, and return one of the best $\alpha \cdot |A(S)|$ elements. This is repeated until a complete solution for the problem is obtained.

The improvement phase is defined by the perturbation function, which consists of selecting a wireless agent $u \in U$ and rerouting it, i.e. finding a complete path using the procedure described above for each time step 1 to T. The set of all perturbations of a solution S is its neighborhood $N(S)$. At each step, all elements $u \in U$ are tested, and the procedure stops when no such element u that improves the current solution can be found [57].

3.2.3 Continuous Formulations

In this section, we present continuous formulations of the CCPM [12]. These formulations will provide a more realistic scenario than the discrete formulation provided above, in that movement is not restricted to a discrete set of positions. We will assume that the agents are operating in a battlespace $\mathcal{Q} \subseteq \mathbb{R}^d$, where $\mathcal{Q}$ is a compact, convex set with unit volume and the Euclidean norm $|| \cdot ||_2$ in $\mathbb{R}^d$. For our purposes, we are going to consider the planar case, i.e. $d = 2$, with the understanding that extensions to higher dimensions are easily achieved. Suppose there are M wireless agents in the ad hoc network. The M agents are assumed to be omnidirectional and are modeled as point masses. We allow the agents to move freely within $\mathcal{Q}$ at some bounded velocity.

3.2.3.1 Formulation 1: A Continuous Analog of CCPM-D

In order to derive a continuous formulation, we need to to define an objective function that is consistent with that of the discrete formulation. Let R_{ij} be the communication constant for agents i and j. That is, R_{ij} is the radius of communication for the two agents. One possible objective is to maximize the *Heaviside function*, defined as

$$H_1\left[R_{ij} - ||\vec{x}(t)^i - \vec{x}(t)^j||_2\right] = \begin{cases} 1, \text{ if } ||\vec{x}(t)^i - \vec{x}(t)^j||_2 \leq R_{ij} \\ 0, \text{ if } ||\vec{x}(t)^i - \vec{x}(t)^j||_2 > R_{ij}. \end{cases} \tag{3–46}$$

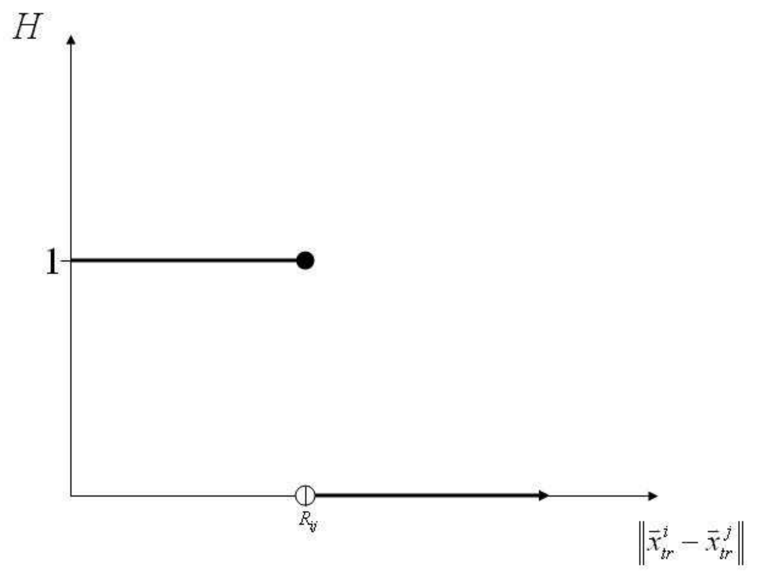


Figure 3-13. The Heaviside function, H_1.

A graphical representation of H_1 is displayed in Figure 3-13. While this function will work as an objective, it is very extreme in the sense that there is a large jump from perfect communication at distances less than or equal to R_{ij} to no communication as soon as the distance becomes larger than R_{ij}. A more desirable function is one that approximates H_1 but degrades in a continuous fashion from perfect to no communication.

We consider two alternatives to H_1. The first is a piecewise continuous, linear function defined by

$$H_2\left[R_{ij} - ||\vec{x}^i_t - \vec{x}^j_t||_2\right] = \begin{cases} 1, \text{ if } ||\vec{x}^i_t - \vec{x}^j_t||_2 \leq R_{ij} \\ 2 - \frac{||\vec{x}^i_t - \vec{x}^j_t||_2}{R_{ij}}, \text{ if } R_{ij} < ||\vec{x}^i_t - \vec{x}^j_t||_2 \leq 2R_{ij} \\ 0, \text{ if } ||\vec{x}^i_t - \vec{x}^j_t||_2 \geq 2R_{ij}. \end{cases} \tag{3–47}$$

This function, whose graph is provided in Figure 3-14, has a value equal to one if agents i and j are within the communication radius R_{ij} of one another. The function then decreases constantly until the agents have distance $2R_{ij}$, at which time they are unable to communicate.

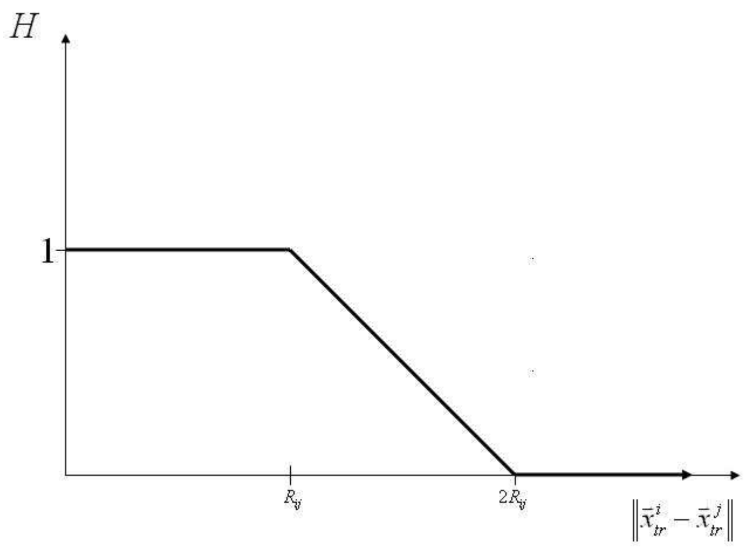


Figure 3-14. H_2, continuous approximation to H_1.

The third and final objective function we will consider is a continuously differentiable decreasing function of the distance between agents i and j. This function, displayed in Figure 3-15, is defined by

$$H_3\left[||\vec{x}^i_t - \vec{x}^j_t||_2, R_{ij}\right] \quad = \quad e^{-\left(\frac{||\vec{x}^i_t - \vec{x}^j_t||_2}{R_{ij}}\right)^2}. \tag{3–48}$$

This is perhaps the best approximation of H_1 in that it can be interpreted as the probability of agents i and j directly communicating as a function of the distance between them.

Now that we have found a suitable objective function we can define the remaining parameters and constraints of the problem. Let $\vec{x}^i(t)$ be the position of agent i at time t. Similarly, let $\vec{v}^i(t)$ be the velocity of agent i at time t. The relationship between velocity and position is the standard one, given by $\vec{v}^i(t) = \frac{dx^i(t)}{dt}$. In order to formulate the continuous time analog of the CCPM, we must constrain the maximum speed of each agent. This is the continuous time analog of the constraints on the maximum distance traveled in the discrete formulation, between any two time steps. If $s_i \in \mathbb{R}^2$ is the starting

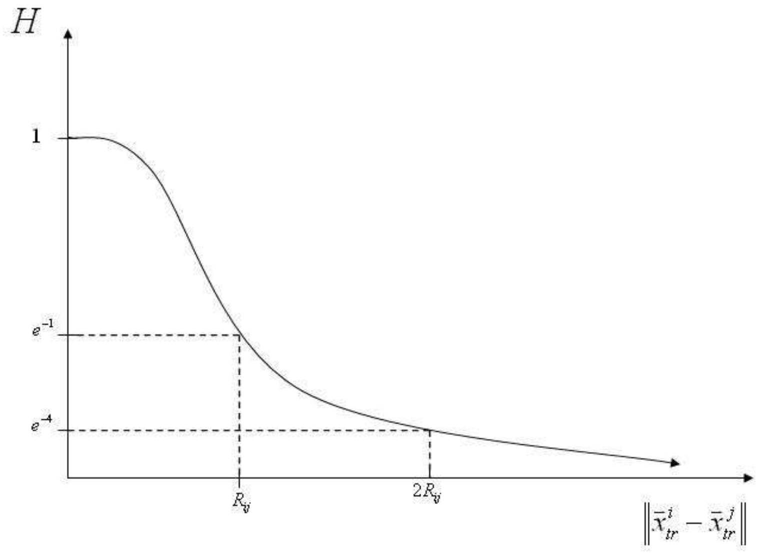


Figure 3-15. H_3, continuously differentiable approximation of H_1.

position of agent i, and $d_i \in \mathbb{R}^2$ is the destination point of agent i, then we can formulate the CONTINUOUS COOPERATIVE COMMUNICATION PROBLEM ON MOBILE AD HOC NETWORKS (CCPM-C) as follows.

$$\max \quad \int_0^T \sum_{i<j} H_3\big[||\vec{x}^i(t) - \vec{x}^j(t)||_2, R_{ij}\big] \tag{3–49}$$

s.t.

$$\vec{x}^i(0) = s_i, \ \forall \ i = 1, \ldots, M \tag{3–50}$$

$$\vec{x}^i(T) = d_i, \ \forall \ i = 1, \ldots, M \tag{3–51}$$

$$||\vec{v}^i(t)|| \leq V_i, \ \forall \ i = 1, \ldots, M, t \in [0, T] \tag{3–52}$$

$$\vec{x}^i(t) \in \mathbb{R}^2, \ \forall \ i = 1, \ldots, M, t \in [0, T] \tag{3–53}$$

The above formulation provides a set of trajectories along which the agents can be routed in order to ensure that the communication between them is maximized. Clearly, if the agents remain in a tightly coupled formation then communication will be maximized; however, unlike the discrete version of the problem in which the vertices of a graph had to be traversed, in a continuous setting this problem is relatively easy and worse yet, not very interesting. Alternatively, consider a set of UAVs involved in a search-and-rescue or reconnaissance mission. For obvious reasons, missions of this sort generally require the UAVs to traverse a large portion of the battlespace before arriving at their destinations. In this case, the above formulation is not helpful. With this in mind, we move on to develop a second continuous formulation which not only maximizes the communication between the agents, but also maximizes the coverage of predefined regions of the battlespace.

3.2.3.2 Formulation 2: A Continuous Formulation Ensuring Location Visitations

In the following paragraphs, we derive a second continuous formulation which guarantees that certain locations will be visited by the UAVs as they traverse the battlespace from their sources to their respective destinations. Previous work on target visitation problems appear in [14]. Once again, we are considering a set of M UAVs. We keep the assumption that the ith UAV starts at a position $s_i = (s_{ix}, s_{iy})$, at time 0, and ends at position $d_i = (d_{ix}, d_{iy})$, at time T. The ith UAV, at time $t \in [0, T]$, has position $\vec{x}^i(t) = (x_i(t), y_i(t))$. Assume that the following holds:

$$\vec{x}^i(t) \in [x_{low}, x_{high}] \times [y_{low}, y_{high}] \;\forall\; i = 1, \ldots, M,\; t \in [0, T].$$

Furthermore, assume that there exists J positions in the domain, each of which must be visited by at least one UAV in the time interval $[0, T]$. These positions are given by $Q_j = (\bar{x}_j, \bar{y}_j)$, for each $j = 1, \ldots, J$. Lastly, the ith UAV has a maximum speed given by $\epsilon_i^{\max}$ for each $i = 1, \ldots, M$ and we assume the minimum speed to be zero.

In order to implement a solution technique in a digital computer, we make use of the $\mathcal{L}_1$-norm as a measure of the distance between two points and discretize the time domain into ρ equal time steps, $\Delta t = T/(\rho - 1)$. Let $t_k = k\Delta t, \;\;$ for each $\;k = 0, \ldots, \rho - 1$. Thus the position of the ith UAV at time step k is given by $\vec{x}^i(t_k) = (x_i(t_k), y_i(t_k))$, for each $i = 1, \dots, M$, and for each $k = 0, \dots, \rho - 1$.

Then the problem, which is denoted as **CCPM-C**, can be written as:

$$\min \quad \sum_{i_1<i_2} \sum_{k=0}^{\rho-1} \big[|x_{i_1}(t_k) - x_{i_2}(t_k)| + |y_{i_1}(t_k) - y_{i_2}(t_k)|\big] \tag{3–54}$$

s.t.

$$x_i(0) = s_{ix}, \quad y_i(0) = s_{iy}, \ \forall\ i = 1, \dots, M \tag{3–55}$$

$$x_i(T) = d_{ix}, \quad y_i(T) = d_{iy}, \ \forall\ i = 1, \dots, M \tag{3–56}$$

$$\epsilon_i^{\max} \geq \frac{1}{\Delta t}\big[|x_i(t_k) - x_i(t_{k-1})| + |y_i(t_k) - y_i(t_{k-1})|\big]\ \forall\ i, \quad \forall\ k = 1, \dots, \rho - 1 \tag{3–57}$$

$$\beta_{ijk}\big[|x_i(t_k) - \bar{x}_j| + |y_i(t_k) - \bar{y}_j|\big] = 0\ \forall\ i,\ \forall\ j \in J, \forall\ k \tag{3–58}$$

$$\sum_{i=1}^{M} \sum_{k=0}^{\rho-1} \beta_{ijk} \geq 1\ \forall\ j \tag{3–59}$$

$$x_{low} \leq x_i(t_k) \leq x_{high} \quad y_{low} \leq y_i(t_k) \leq y_{high}\ \forall\ i,\ \forall\ k \tag{3–60}$$

$$\beta_{ijk} \in \{0, 1\} \quad \forall\ i,\ \forall\ j,\ \forall\ k. \tag{3–61}$$

Theorem 3.1. *The above formulation for* ***CCPM-C*** *is correct.*

Proof. Clearly, the objective function minimizes the pairwise distances between the agents. Thus as the distance between the agents decreases, the communication increases. Constraints (3–55) and (3–56) respectively specify the starting points and the destination points for the agents. The constraint set(3–57) bounds the speed of the agents. Next (3–58) and (3–59) ensure that at least one agent co-locates with the set of points in the domain which must be visited. More specifically, (3–58) implies that for all points $j \in J$, there must be a time when an agent occupies position j. In the constraint, this is accomplished by ensuring the distance between the visiting agent and point j is 0.

Constraint (3–59) implies that at least one agent must visits each point $j \in J$. Finally, constraints (3–60) and (3–61) define the domain of the decision variables. $\square$

We can linearize the mixed integer programming formulation in (3–54)-(3–61) as follows. To begin with, replace the objective function (3–54) with

$$\sum_{i_1<i_2} \sum_{k=0}^{\rho-1} \hat{x}_{i_1 i_2 k} + \hat{y}_{i_1 i_2 k} \tag{3–62}$$

with the additional constraints

$$\begin{aligned}
\hat{x}_{i_1 i_2 k} &\geq x_{i_1}(t_k) - x_{i_2}(t_k) \ \forall\ i_1, i_2 = 1, \ldots, M, i_1 < i_2, \\
&\qquad \forall\ k = 0, \ldots, \rho - 1 && (3–63) \\
\hat{x}_{i_1 i_2 k} &\geq -\big[x_{i_1}(t_k) - x_{i_2}(t_k)\big] \ \forall\ i_1, i_2 = 1, \ldots, M, i_1 < i_2, \\
&\qquad \forall\ k = 0, \ldots, \rho - 1 && (3–64) \\
\hat{y}_{i_1 i_2 k} &\geq y_{i_1}(t_k) - y_{i_2}(t_k) \ \forall\ i_1, i_2 = 1, \ldots, M, i_1 < i_2, \\
&\qquad \forall\ k = 0, \ldots, \rho - 1 && (3–65) \\
\hat{y}_{i_1 i_2 k} &\geq -\big[y_{i_1}(t_k) - y_{i_2}(t_k)\big] \ \forall\ i_1, i_2 = 1, \ldots, M, i_1 < i_2, \\
&\qquad \forall\ k = 0, \ldots, \rho - 1 && (3–66)
\end{aligned}$$

Next, we replace (3–57) with

$$\alpha_{ik} + \bar{\alpha}_{ik} \leq \Delta t \epsilon_i^{\max} \tag{3–67}$$

adding the constraints

$$\alpha_{ik} \geq x_i(t_k) - x_i(t_{k-1}) \; \forall \; i = 1, \ldots, M, \; \forall \; k = 0, \ldots, \rho - 1 \tag{3–68}$$

$$\alpha_{ik} \geq -\big[x_i(t_k) - x_i(t_{k-1})\big] \; \forall \; i = 1, \ldots, M, \; \forall \; k = 0, \ldots, \rho - 1 \tag{3–69}$$

$$\bar{\alpha}_{ik} \geq y_i(t_k) - y_i(t_{k-1}) \; \forall \; i = 1, \ldots, M, \; \forall \; k = 0, \ldots, \rho - 1 \tag{3–70}$$

$$\bar{\alpha}_{ik} \geq -\big[y_i(t_k) - y_i(t_{k-1})\big] \; \forall \; i = 1, \ldots, M, \; \forall \; k = 0, \ldots, \rho - 1 \tag{3–71}$$

Finally, we replace (3–58) with

$$\phi_{ijk} + \bar{\phi}_{ijk} = 0 \tag{3–72}$$

and add the constraints

$$\theta_{ijk} \geq x_i(t_k) - \bar{x}_j, \; \forall \; i, \; \forall \; j, \; \forall k \tag{3–73}$$

$$\theta_{ijk} \geq -\big[x_i(t_k) - \bar{x}_j\big], \; \forall \; i, \; \forall \; j, \; \forall \; k \tag{3–74}$$

$$\bar{\theta}_{ijk} \geq y_i(t_k) - \bar{y}_j, \; \forall \; i \; \forall \; j \; \forall k \tag{3–75}$$

$$\bar{\theta}_{ijk} \geq -\big[y_i(t_k) - \bar{y}_j\big], \; \forall \; i, \; \forall \; j, \; \forall \; k \tag{3–76}$$

$$\phi_{ijk} \leq \beta_{ijk}(x_{high} - x_{low}), \; \forall \; i, \; \forall \; j, \; \forall \; k \tag{3–77}$$

$$\phi_{ijk} \geq 0, \; \forall \; i, \; \forall \; j, \; \forall \; k \tag{3–78}$$

$$\phi_{ijk} \leq \theta_{ijk}, \; \forall \; i, \; \forall \; j, \; \forall \; k \tag{3–79}$$

$$\phi_{ijk} \geq \theta_{ijk} - \big[1 - \beta_{ijk}\big]\big[x_{high} - x_{low}\big], \; \forall \; i, \; \forall \; j, \; \forall \; k \tag{3–80}$$

$$\bar{\phi}_{ijk} \leq \beta_{ijk}(y_{high} - y_{low}), \; \forall \; i, \; \forall \; j, \; \forall \; k \tag{3–81}$$

$$\bar{\phi}_{ijk} \geq 0, \; \forall \; i, \; \forall \; j, \; \forall \; k \tag{3–82}$$

$$\bar{\phi}_{ijk} \leq \bar{\theta}_{ijk}, \; \forall \; i, \; \forall \; j, \; \forall \; k \tag{3–83}$$

$$\bar{\phi}_{ijk} \geq \bar{\theta}_{ijk} - \big[1 - \beta_{ijk}\big]\big[y_{high} - y_{low}\big], \; \forall \; i, \; \forall \; j, \; \forall \; k. \tag{3–84}$$

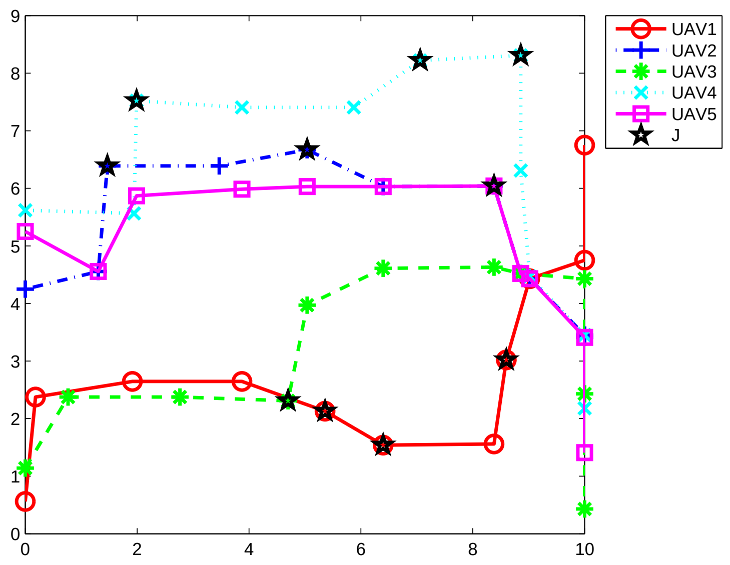


Figure 3-16. Example with 5 agents.

Thus the problem becomes one of minimizing (3–62) subject to the constraints (3–55), (3–56), (3–59)-(3–61), (3–63)-(3–84). The resulting formulation is a mixed integer linear program (MILP) and can be solved using a number of commercial software packages. In the following section, we present some preliminary results from one such package, as well as providing a discussion of the experiments.

3.2.4 Case Studies

We have implemented the MILP formulation of the CCPM using the CPLEX$^{\text{TM}}$ optimization suite from ILOG [63]. CPLEX contains an implementation of the simplex method [110], and uses a branch and bound algorithm [199] together with advanced cutting-plane techniques [118, 154].

The instances were tested on grids of size 10. The set of coordinates, J, to be visited were generated uniformly at random. Three sets of coordinates were generated and each visited by three different sets of UAVs, numbering 5, 7, and 10. The y-coordinates of the starting and ending positions were also randomly generated using a uniform distribution

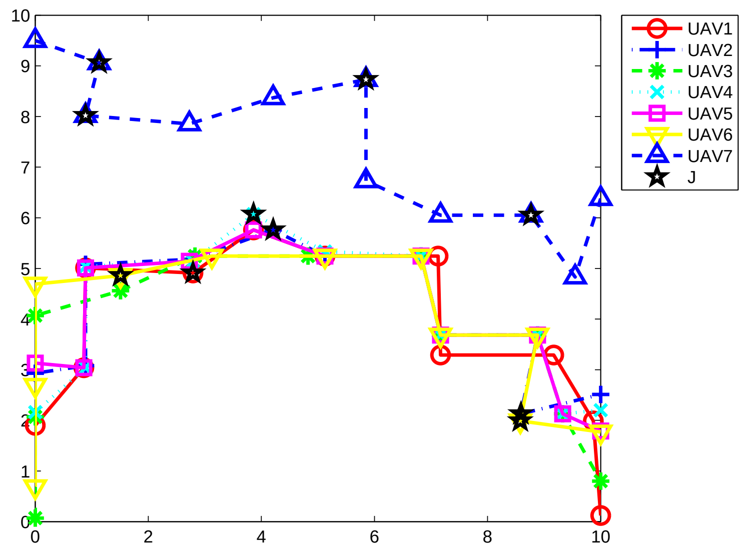


Figure 3-17. Example with 7 agents.

and the x-coordinates were assumed to be 0 and 10 respectively. The scenarios were solved making use of the MILP formulation derived above. The optimal solutions were obtained for the instances with 5 UAVs. The instances with 7 UAVs and 10 UAVs were stopped at optimality tolerances of 10% and 25% respectively. A time-frame of 10 units was provided as an input and the minimum and maximum speed of the UAVs were 0 and 2 units respectively.

We have provided three graphical representations of the trajectories of the agents from each problem set. Figure 3-16 shows the paths traversed in one of the scenarios containing 5 agents. The movements of the agents are from left to right in the figure. The points which must be visited are denoted as stars. We see that from their starting points, the agents tend to converge into a tight formation. Notice that UAV2 separates from the group around point $(7, 6)$ in order to visit three "must visit" points and arrive at its destination. The remaining agents travel together until they must diverge to reach their destinations.

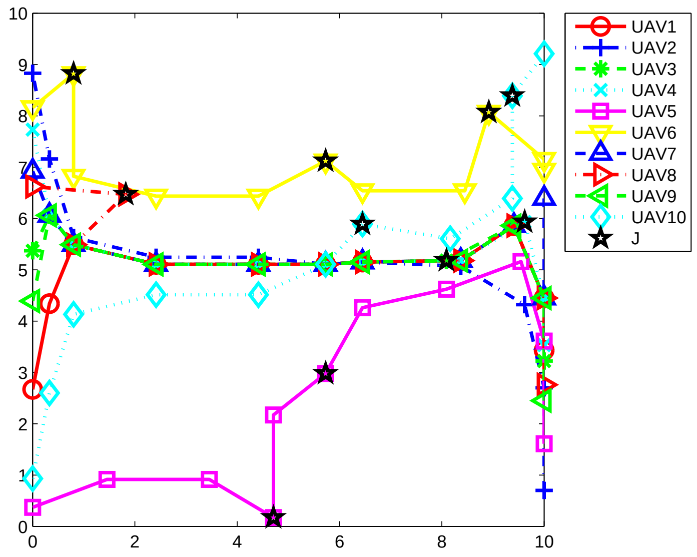


Figure 3-18. Example with 10 agents.

In Figure 3-17 we have an example scenario containing 7 agents. As before, the agents quickly converge to a tightly coupled formation with agents leaving the group only to visit the points in J or arrive at their destinations. A 10 agent scenario is depicted in Figure 3-18 and similar behaviors of the agents can be observed. Figures 3-19-3-24 show how the paths of the agents change (from the 10 agent scenario in Figure 3-18) as agents are randomly removed from the scenario. As expected, the remaining agents are forced to spread out in order to ensure visiting the J target points.

The results presented are very promising. Indeed, the agents exhibit the exact behavior we would expect to see given the nature of the CCPM. As communication strength is inversely proportional to distance, the convergence to a common path clearly indicates that the UAVs are attempting to maximize the communication amongst the group. Consider the scenario in Figure 3-16. Notice that midway through the mission a very clear clustering effect can be seen as two distinct groups of agents make their way towards the “must visit” points and ultimately, their destinations.

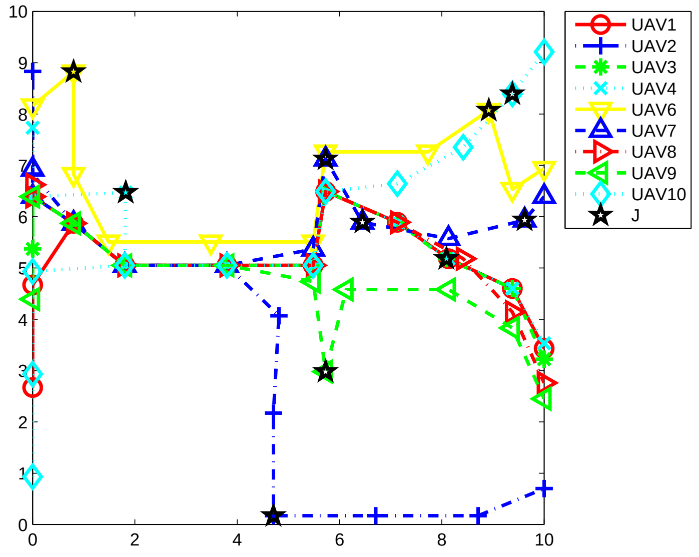


Figure 3-19. Example derived from 10 agent example, with one agent removed.

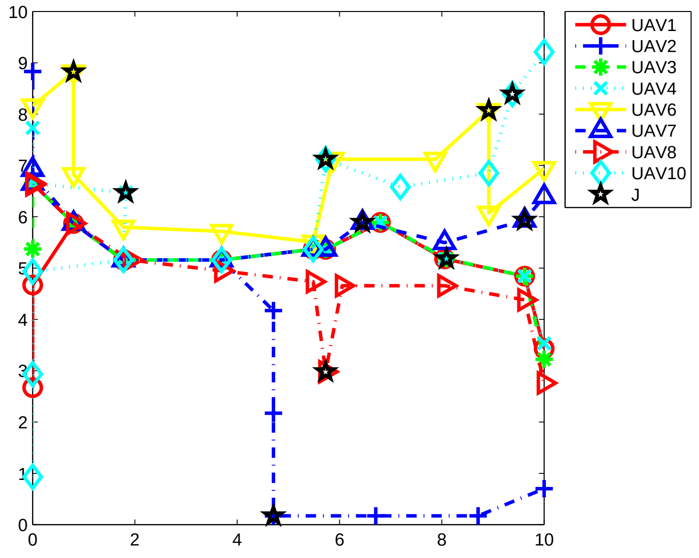


Figure 3-20. Example derived from 10 agent example, with two agents removed.

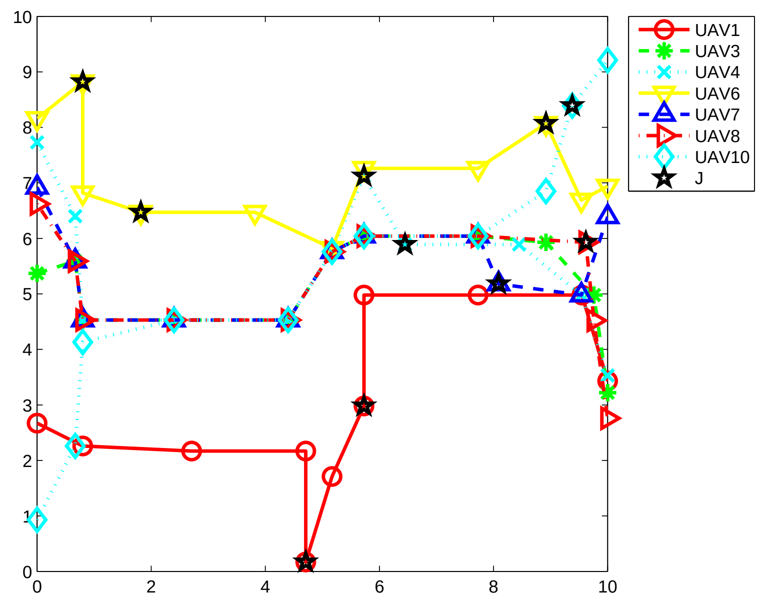


Figure 3-21. Example derived from 10 agent example, with three agents removed.

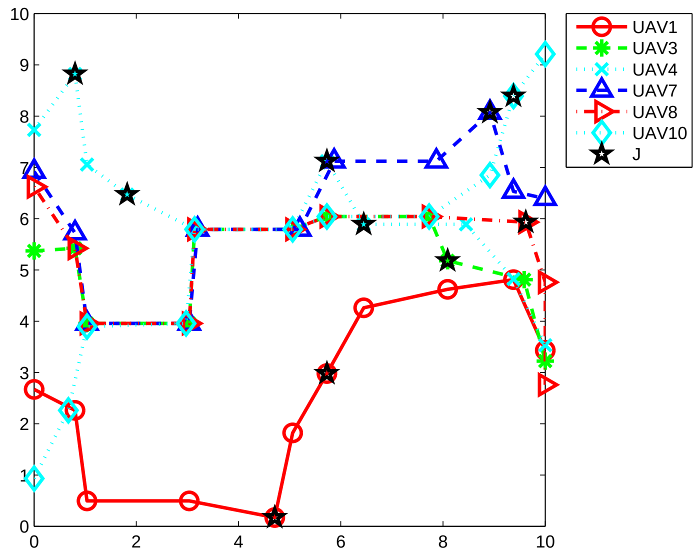


Figure 3-22. Example derived from 10 agent example, with four agents removed.

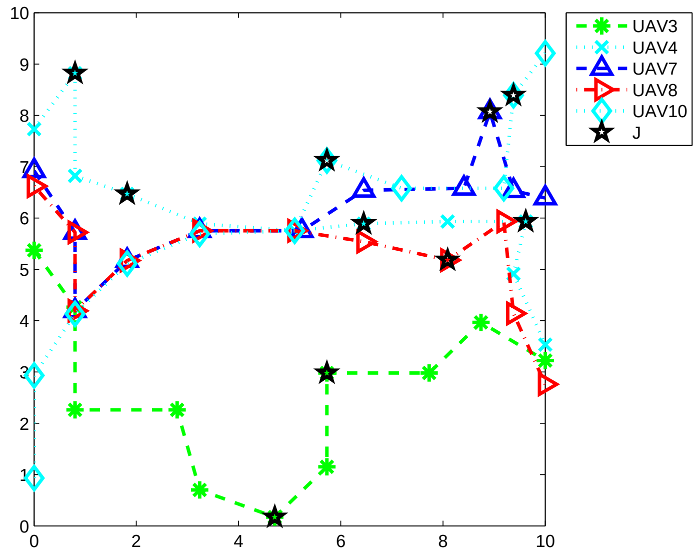


Figure 3-23. Example derived from 10 agent example, with five agents removed.

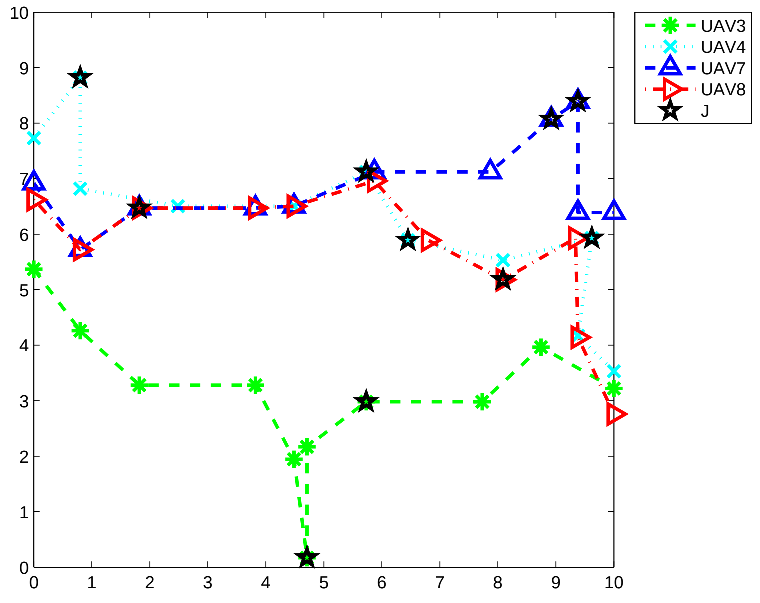


Figure 3-24. Example derived from 10 agent example, with six agents removed.

3.2.5 Conclusion

In this chapter, we studied the so-called TARGET VISITATION PROBLEM whose objective is to plan the sequence for an unmanned aerial vehicle to visit a set of targets which minimizes the total distance traveled and maximizes the utility of the sequence. Until now, the literature on this problem has been slight [100]. This chapter presents the first extensive computational analysis for the problem. First we provided a mathematical model for the TVP based on integer linear programming and proved that finding an optimal solution is NP-complete. To overcome the computational complexity, we described the implementation of a random keys based genetic algorithm for finding near optimal solutions [23]. The heuristic was then hybridized by the implementation of a local search procedure. The numerical results presented demonstrated the effectiveness of the proposed procedure. Out of 6250 experiments, the hybrid heuristic calculated optimal solutions for over 99.9% of the trials in a fraction of the time required by the commercial integer programming solver CPLEX.

Since the TVP is a relatively new problem in the literature, there are several directions for future research. Clearly other metaheuristics can be implemented and compared with GA approach. However, due to the computational complexity, decomposition techniques should most likely to be the focus in order to determine optimal solutions for large-scale instances of the problem. Extensions to the model proposed are also possible such as imposing a constraint on the total distance traveled and generalizing the model to include multiple vehicles. Visiting mobile targets would present other interesting challenges beyond the model presented in this chapter.

We also provided a review of recent work in the area of cooperative communication in mobile ad hoc networks. While inherently a problem of path planning, our formulations incorporated communication as a measure of the fitness of a given solution. We presented some discrete versions of the problem and derived two continuous formulations, the first time this has been considered. The advantage of the new models is that they ensure that a specified amount of the battlespace is explored by the agents. This addition is important in real-world applications particularly in the areas of surveillance, reconnaissance, and rescue operations. The preliminary numerical results demonstrate the effectiveness of the proposed models.

Due to the inherent complexity of the problem, future research will focus on continuous heuristic techniques for the newly proposed models, similar to those found in [111, 112]. Percentile risk constraints will be incorporated into the formulation. Commonly applied in financial applications, risk measures such as Value-at-Risk (VaR) and Conditional Value-at-Risk have proven to be effective tools for military applications as well [56, 59, 60]. We also plan to provide some theoretical results regarding feasibility of problem instances.

CHAPTER 4
REVIEW OF EVACUATION PROBLEMS

4.1 Introduction

Hurricanes, earthquakes, industrial accidents, nuclear accidents, terrorist attacks and other such emergency situations pose a great danger to the lives of the populace. Evacuation during these situations is one way to increase safety and avoid escalation of damages. Evacuation problems are being given increased attention over the last five years. The techniques that are currently employed could be broadly categorized into optimization or simulation methods. In either case, the evacuation problem is dealt over a network where arcs or edges are the roads linking two places or the nodes of the network. The typical factors usually taken into consideration by these models are origin-destination assignment, response time of the evacuees, modes of transportation, contra flows etc. Factors such as origin-destination assignment and arc capacities could be set as static or dynamic and these decisions heavily influence the evacuation efficiency. Most of the recent surveys and reviews in the field of evacuation were made for specific instances [104, 178]. The surveys cater to a specific technique and discuss in detail on their impact on emergency evacuation. This chapter is more across the board. The objective is to present a comprehensive report on the techniques that are available in the literature and broadly classify them. An initial classification was made depending on the approaches used in the evacuation models, namely optimization-based or simulation-based approaches. Further, a subclassification was made based on some vital features considered by the model. These are features that are expected to have a significant impact on evacuation efficiency. The solution methodologies, for every model under these classifications, were discussed in detail and were also assessed based on their computational performance, scalability, extensibility, realizability and the major components considered. Computational efficiency of the model is important in case of unforeseen events. The models needs to be executed quickly to generate alternative plans and prepare for the dynamic scenarios. The complexity of the evacuation problems makes the researchers resort to for heuristic procedures and simulation based approaches. Thus the solutions tested and provided needs to be scalable, or in other words, realizable on larger instances. Most of the models are customized to specific situations but when needed to accommodate additional features and handle more parameters they need to be extensible. The realizability of the model is achieved when they are tested on real time networks and most of the theoretical guarantees are accomplished. The chapter finally concludes with few words on the shortcomings of the available models that may require attention and lays down vital features that a new designer has to seek in a model.

4.2 Optimization Techniques

A large number of optimization models, sometimes referred to as analytical models in the literature, have been developed for evacuation studies.While some of these models are discrete, the rest are extensions from these base models. However, these models are essentially a simple network flow problem trying obtain a minimum cost flow from source to destination. A detailed discussion of the flow problems is carried out in the following section.

4.2.1 Maximum Dynamic Flow

An elementary evacuation flow problems could be formulated as a linear integer program using a variant of the maximum dynamic flow problem [76, 77]. The maximum dynamic flow problem is to determine the maximum amount flow from origin to destination within a specific time T. Ford and Fulkerson formulated this problem on a time expanded static network, where each node and edge is replaced by T copies corresponding to each time instance [76]. Given a digraph $G(V, E)$ and time interval T, let $c(u, v)$ and $t(u, v)$ be the capacity and traversal time of arc $(uv) \in E$. Let $x(u, v, \tau)$ be the amount of flow leaving node u along arc (u, v) at time τ. Let the origin node be s and destination node be t. Assuming holding over of flow over any node is allowed, we have the following integer linear programming problem that solves the maximum dynamic flow problem:

$$\text{Maximize} \quad \sum_{\tau=0}^{T}\sum_{(su)\in E} x(s,u,\tau) - \sum_{\tau=0}^{T}\sum_{(us)\in E} x(u,s,\tau - t(u,s)) \tag{4–1}$$

s.t.

$$\sum_{(uv)\in E} x(u,v,\tau) - \sum_{(vu)\in E} x(v,u,\tau - t(v,u)) = 0$$
$$, \ \forall\ u \neq s,t, u \in V, \tau = 0,1,\ldots,T \tag{4–2}$$

$$[\sum_{\tau=0}^{T}\sum_{(su)\in E} x(s,u,\tau) - \sum_{\tau=0}^{T}\sum_{(us)\in E} x(u,s,\tau - t(u,s))]$$
$$+[\sum_{\tau=0}^{T}\sum_{(tu)\in E} x(t,u,\tau) - \sum_{\tau=0}^{T}\sum_{(ut)\in E} x(u,t,\tau - t(u,t))] = 0 \tag{4–3}$$

$$0 \leq x(u,v,\tau) \leq c(u,v), \forall (uv) \in E, \tau = 1\ldots T \tag{4–4}$$

The objective function that has to be maximized gives the net amount of flow leaving the origin node s by time period T. The constraint set (4–2) ensures conservation of flow at every node, where the amount of flow that enters a node is exactly equal to the amount of flow that leaves the node at any time period. The constraint set (4–3) ensures that at the end of T time periods the amount of flow that leaves the origin s is equal to the amount of flow that enters the destination t. The above formulation solves a maximum flow problem over a time expanded graph. The problem becomes extremely difficult to solve for larger graphs with a bigger time frame. However, [76] suggested a strictly polynomial algorithm for the maximum dynamic flow. They solved the minimum cost flow problem on the original graph, without time expansion, and decomposed the flow into a set of paths. Then they obtained the maximum dynamic flow by temporally repeating the flow along the paths.

Most of the evacuation problems ramify from the maximum dynamic flow problem. The quickest flow problem, sometimes referred to as evacuation problem, is to determine the minimum time required to send a given amount of flow from the origin to the destination. This problem is a simple variation of the maximum dynamic flow problem and could be solved through a binary search. [34] proved this reduction from the maximum dynamic flow problem to the quickest flow problem and provided a strongly polynomial time algorithm through a parametric search on the quickest time by repeatedly solving the maximum dynamic flow problem. Many other work have been done in this line generalizing this concept. [74] worked on multicommodity shipment of flows. A multicommodity flow problem on a static graph without a time bound is: Given a graph with a arc travel time and capacity on each arc and a set of commodities $K = 1, \ldots, k$ with each commodity having specific origin s_i and destination t_i, it is required to send a specific amount of flow from s_i to t_i of the corresponding commodity in the minimum amount of time. The problem is formulated in the following manner.

$$\text{Minimize} \quad \sum_{i=1}^{k} \sum_{(uv)\in E} c(u,v,i)x(u,v,i) \tag{4–5}$$

s.t.

$$\sum_{(uv)\in E} x(u,v,i) - \sum_{(vw)\in E} x(v,w,i) = 0, \forall v \in V, i = 1, \ldots, k \tag{4–6}$$

$$\sum_{(ut_i)\in E} x(u,t_i,i) - \sum_{(t_iv)\in E} x(t_i,v,i) \geq d_i, \forall i = 1, \ldots, k \tag{4–7}$$

$$0 \leq x(u,v,i) \forall uv \in E, i = 1, \ldots, k \tag{4–8}$$

$$\sum_{i=1}^{k} x(u,v,i), \leq c(u,v) \forall (uv) \in E, i = 1, \ldots, k \tag{4–9}$$

$x(u,v,i)$ is the amount of flow on arc (uv) of commodity i and $c(u,v,i)$ is the cost of unit flow on arc (uv) of commodity i. The constraint set (4–6) implies the flow conservation at a node for a particular commodity and the constraint set (4–7) ensures that the sinks node t_i receives d_i amount of flow. Finally, (4–8) and (4–9) restraints the flow capacity on each arc. The above is a multicommodity flow problem for a static graph. In order to solve the quickest multicommodity problem we have to extend the formulation to a time expanded graph similar to the previous problem formulation. However, this results in numerous constraints and variables if the time horizon, T, considered in large. Each node and arc of the graph is replaced by T copies, each corresponding to the specific time instant. The multicommodity flow problem is known to be mathsfNP-hard even for a static graph [84]. To overcome the time expansion difficulty, [74] suggested a scaling algorithm where in each node and arc is replaced by T/δ copies instead T, thus reducing the problem size considerably and striking a balance between the precision of the result and the running time of the algorithm. [116] provided solutions for three variations of maximum dynamic flow problem. They provided a polynomial time approximation for the earliest arrival flow problem, which was studied by [196] and [143]. The earliest arrival flow problem requires the flow to be maximized at each time step of the given horizon, unlike the maximum dynamic flow problem. The proposed algorithm is based on successive shortest path algorithm, where the flow is augmented along the shortest path, quickest path in our case, and the chain decomposition is given by augmentations performed in a sequence of static graphs. However, successive shortest path is a pseudo-polynomial algorithm. This difficulty is usually handled by scaling. Unlike traditional scaling, the proposed algorithm performs an upward capacity scaling. A dynamic flow quickest path with a small capacity could be repeated temporally to obtain a maximum flow. We refer to [116] for a detailed account of the algorithm and proof of approximation. The second problem studied was lexicographic maximum dynamic flow. Given a set of sources and their priority of evacuation the lexicographic maximum dynamic flow maximizes the flows leaving the sources in the specified order. Finally, they studied the quickest flow problem with fixed number of sources and destinations with equal priorities having specific supply demands. [122]pointed out that the problem of determining quickest flow from any subset of nodes to any other subset of nodes is equivalent to a single source shortest path problem. Thus for a fixed number of sources and destinations this problem is polynomially solvable, if we consider all possible subsets of the sources and destinations. In the same fashion, for a fixed number of source and destination, the lexicographic maximum dynamic flow could be solved for all possible ordering of the sources to solve the above problem. A detailed and more efficient algorithm was given by [115] for this quickest transshipment problem with more than one origin and destination. These models based on maximum dynamic flow problem have their practical limitations as they are oblivious to factors such as congestion control, dynamic origin and destination demand and multimodal transportation, which are quite conceivable in a real time situation. The flow problems are rather a simplified versions of real time model. The models are computationally efficient with polynomial time solutions for exact or approximate solutions. The problems thus could help in providing quick solutions to large problem sizes under a simplified setting. The realizability of the model is limited but their computational efficiency could be taken advantage of in preprocessing stages of analytical techniques and heuristic developments.

4.2.2 Dynamic Traffic Assignment

[140] introduced the dynamic traffic assignment (DTA) problem and formulated it as a non-linear program with a non-convex mathematical program. The problem is to find an assignment of flows on the links optimally. Like before, let us assume the planning horizon to be T. And $G(V, E)$ be the digraph under consideration. Let $x(u, v, \tau)$ be the amount of flow on arc (u, v) at time τ. Let $F(u, \tau)$ be the external input in node u at time τ and $h_{uv,\tau}(x(u, v, \tau))$ be the cost function. Also, let $g_{uv}(x(u, v, \tau))$ be the exit function denoting the amount of flow that exits from arc u,v during period τ. Finally, $d(u, v, \tau)$ be the amount of flow entering arc (uv) at node u. The DTA problem is formulated as follows:

$$\text{Minimize} \quad \sum_{\tau=0}^{T} \sum_{(uv)\in E} h_{uv,\tau}(x(u,v,\tau)) \tag{4–10}$$

s.t.

$$x(u,v,\tau+1) - x(u,v,\tau) + g_{uv}(x(u,v,\tau))$$
$$-d(u,v,\tau) = 0, \forall (uv) \in E, \tau = 1,2,\ldots,T \tag{4–11}$$

$$\sum_{\forall uv \in E} d(u,v,\tau) - F(u,\tau)$$
$$-\sum_{\forall pu \in E} g_{pu}(x(p,u,\tau)) = 0, \forall u \in V \backslash t, \tau = 1,2,\ldots,T \tag{4–12}$$

$$0 \le x(u,v,\tau), \forall (uv) \in E, \tau = 0 \ldots T \tag{4–13}$$

$$0 \le d(u,v,\tau) \forall (uv) \in E, \tau = 0 \ldots T \tag{4–14}$$

This work by Merchant and Nemhauser was the opening to dynamic traffic assignment. They formulated discrete piecewise linearized version of the traffic assignment problem. The problem assumes that the demands are available and it has a single destination. In most of the emergency situations these assumptions are not preserved. For instance, of a multicommodity flow where the it is required to maintain the origin-destination pair a single destination model may not be suitable. Also, dynamic demand in place of a static demand estimate is more efficient and more precise in a real-time situation. [39] validated the model by proving that the constraints satisfy linear independence constraint qualification as the exit function is continuously differentiable. [113] linearized the exit and cost function and obtained the global optimum for the nonlinear, non-convex optimization problem by solving a series of T+1 optimization problems. [40] provided a link flow non-linear mixed integer programming formulation and a convergent dynamic algorithm to solve the dynamic user equilibrium problem. It explicitly seeks equilibrium in terms of path travel times unlike Merchant and Nemhauser's model. The model depends on static use equilibrium functions with additional constraints to ensure temporally continuous flow. The technique itself is not pertinent to evacuation studies and hence we refer to the work by [164] who made a detailed study on dynamic traffic assignment in their survey. We are more interested in its applicability in emergency situation, however the little background provided is required.

[65] introduced the cell based dynamic traffic assignment that segmented the highway links into equal sized cells, such that each cell could be traversed in an unit time. Each cell has a specific capacity and the congestion is explicitly handled by restraining the amount of flow from one cell to another. [206] relaxed Daganzo's formulation by holding of flow at nodes in the flow conservation equations. [50] implemented a dynamic traffic modeling technique based on the cell transmission model [65] for an optimal no-notice mass evacuation. They also proposed a network transformation to a single destination network and then a cell network, for the above implementation. Their objective in no-notice evacuation also includes identification of destinations. The modeling is done through a graph transformation. The model employs aggregation of zones and hence could be scaled appropriately to handle large graphs. The model assumes prior knowledge of originating demands and zonal information, which in most cases are available. Also, the optimization formulation was provided for a cell-based transmission for a time expanded graph. The model optimization model as such might be time consuming while applied over real time graphs. The authors pointed out that without user equilibrium constraints the model may not be of practical interest. The model might be useful during no-notice mass evacuation, assuming that there is an efficient technique to solve it, but the assumptions make it rigid to extend it to other emergency situations. Another work employing cell transmission model is by [133]. They discuss the staged evacuation procedure, where in a zonal classification of the nodes is done based on the severity of impacts they suffer and starting time for each evacuation zone is determined, taking the response time of evacuees into consideration. These models based on cell based transmission involves in a formulation for a time expanded static graph with each link replaced by a group of cells. This methodology may not yield quicker results for an evacuation problem considered

over a larger space and time and this may be a necessity in an emergency situation. The complexity of the models will further increase when they consider more complex features such as user equilibrium constraints, congestions management, dynamic demand etc. This computational difficulty could be overcome the aggregating the nodes of the network under consideration and solving the problem in a smaller graph. The efficiency could also be improved by increasing the coarseness of the discretization of time depending on the accuracy of the results desired. The models suggested an evacuation procedure with single destination or sink, which cannot handle multicommodity flows to identify optimal destinations. [88] suggested a heuristic to the cell-based transmission model with multiple destinations to overcome this implementation difficulty. However, the accuracy of the heuristic and its convergence when the problem size grows is a question of consideration. [42] detailed about the concerns in a time expanded evacuation problem. Time expanded networks, while having computational limitations in a large scale evacuations, might benefit the small scale networks such as the building evacuation as the network under consideration is quite small compared to large scale evacuation networks. [142] provided a pseudopolynomial algorithm for solving the maximum dynamic flow and quickest flow problems with a time-varying travel time, node and arc capacity. We refer to [126] for more details on small-scale evacuation, where they made a comprehensive review on the building evacuation models.

The variational inequality approach is another way of formulating the the DTA problem [25, 170, 164, 19]. Variational Inequality provides a convenient formulation technique for network equilibrium problems arising in economics, finance and transportation. It was put forth by [108] to study partial differential equations problems. [64] studied the equilibrium problems in transportation networks applying variational inequality for a static travel demand. A finite dimensional variational inequality problem could be stated as: Given a subset $k \subset R^n$ find a vector $x^* \in k$ such that

$$\langle F(x^*), x - x^* \rangle \geq 0, \forall x \in k \tag{4–15}$$

where $F : k \rightarrow R^n$ is a continuous function. In case of the transportation problem the F is the cost function and x is the flow on a arc and n being the number of arcs in the network. The equilibrium conditions could be viewed as Kuhn-Tucker conditions for a convex nonlinear problem. [150] gave a detailed account on variational inequalities and their applications to several equilibrium problems. A much wider range of problems including multimodal transportation and elastic demands were discussed along with a discussion of algorithms to network equilibrium problems formulated with variational inequalities. [79] provided a variational inequality formulation for a continuous time problem with dynamic route choices and departure time decisions, a dynamic version of the static Wardropian user equilibrium. It considers the path costs realized through travel time and penalty for early or late arrival time and the optimal flow pattern is simultaneous route departure equilibrium. As the model is dealt in a continuous time domain it might lack a good algorithm to efficiently solve complex integrals on the path costs presented in the model. Another continuous time model for transportation networks was provided by [167]. They considered a single commodity problem with elastic demand and a time-dependent equilibrium and expressed the corresponding equilibrium problem as quasi-variational inequality. The subset $k \subset R^n$ in equation (4–15) is replaced by a point-to-set map $K : E \rightarrow R^n$. [30] presented another variational inequality based equilibrium problem for a multicommodity flow problem where the user's desired origin-destination pair and departure and arrival time window is provided. [170] have a detailed section of variational inequality models in the transportation framework.

4.2.3 Non-deterministic methods

Significant work has been done on evacuation problems from non-deterministic perspective. Smith et. al have done substantial work on stochastic evacuation network [184, 185, 183]. They provide non-inferior solutions to bi-objective routing problem in queueing networks. The two objectives modeled were the distance traveled and the clearance time. The problem also considered the multicommodity flow problem. They provided a path based bi-objective formulation with flow conservation and capacity constraints for all the routing alternatives. The routes are generated and fed to the above formulation by an algorithm that iteratively generates candidate paths to assess congestion. [189] gave an overview of deterministic, stochastic and hybrid methods for evacuation problems and compared a analytical queueing network with a simulation model for a hospital evacuation considering evacuation time, congestion and optimal routes. [155] provided two solutions, one with optimal evacuation route and another with set of strategies from which the evacuee can choose the best arc at each step. They dealt the evacuation problem on network with stochastic, time-varying travel time. [194] considered a stochastic dynamic network in which the origin destination demand are random variables with known probability distributions. A linear programming formulation based on the system optimal dynamic traffic assignment propagating traffic by cell transmission was provided. The results are robust as they build a confidence level which requires the solutions to meet the expected demands. [157] incorporated capacity constraints in cell transmission model for system optimal dynamic traffic assignment. The capacity constraints are probabilistic in nature due to the impacts of the disaster. They also compared the model to one with a deterministic capacity. [18] studied the resource allocation problem in the emergency evacuation network. They employed queueing theory to model the varying capacities circulation spaces, such as finite sized corridors, staircases etc. and study their effects on throughput. The evacuees in this model experience a service at the evacuation nodes and this service rate decays with the increasing amount of traffic.

4.3 Significant Features in Optimization techniques

This section will discuss the factors that are widely regarded to influence the evacuation efficiency. While some of them are only germane to optimization techniques, the rest will be revisited when we discuss the simulation techniques.

4.3.1 Contra Flows

A contra flow or reversible flow is a strategy to improve the efficiency of evacuation by allowing traffic in the opposite direction of the roads during an emergency. Most of the recent evacuation plans proposed incorporates contra flow in them. The contraflows while being efficient in evacuation they come with a few pitfalls like increase in accidents and operation cost, which might be a consideration for their implementation in the evacuation plans [29].

Consider the following lane reversal problem: Given a directed graph $G(V, E)$ with non-negative arc capacities and travel times, a subset of nodes S as sources and planning period or horizon T, determine the quickest time to evacuate all the occupants from source to destination by allowing lane reversals. A single source single sink contraflow problem is polynomially solvable, but a multiple source and sink problem becomes strongly mathsfNP-hard [171]. Thus, even simple cases of the problem is hard to solve and researchers have to resort to heuristic. Simulation procedures could be performed for a given configurations of networks and hence may not be a suitable tool to identify optimal lane identification, but the analytical procedures are not computationally efficient even for simple models without complex features [200]. f

[121] proposed two heuristics to solve the evacuation problem with contra flow more than one source and destination. The first heuristic was proposed for a time expanded static graph $G_T(V_T, E_T)$. The heuristic runs on a time expanded graph, making it less attractive for large scale evacuation with large number of nodes and with evacuation planned over a bigger time frame. The second heuristic proposed was a simulated annealing procedure. The algorithm begins with solution and perturbs it a little to obtain a better solution and this iterative procedure is carried out until a stopping condition is satisfied. This is a standard technique employed to escape local minimum and obtain values close global minimum. A similar tabu-based heuristic algorithm to solve lane reversible evacuation problem was proposed by [193]. It is a local search procedure

attempting to get the best solution in a given neighborhood. The system optimal dynamic traffic assignment model for the cell based transmission which was described in section 2.2 is considered for the implementation of contraflows and it permits partial capacity reversal, unlike the former approach. In another work [192], they formulated the dynamic traffic assignment model with lane reversible capabilities. This information could be used as an efficient initial solution for the tabu based search. The models discussed are simple flow problems with lane reversal capability and is far from being practically realizable. The models could however serve as a good preprocessing technique to identify the lanes to be reversed and then a more complex model could be employed on a reconfigured network. The heuristics developed to solve them, though computationally efficient, does not have any theoretical guarantee in the quality of the solutions. Thus the issue of scalability is a concern, as the quality of the solution could be compromised with the problem size. Approximate solutions to the contraflow problems for such guarantees is still an open area of study.

4.3.2 Evacuee Behavior

Like contra flows, another factor that is incorporated in the evacuation models in the recent years is the the response time of the evacuees. The response time is the time taken for a decision by the evacuee to evacuate. This latency could be due to the inefficiency in the information dissemination, panic in evacuees or the emergency situation itself. Conventional models disregard this, assuming zero loading and unloading time. [179] formulated a Nuclear power plant evacuation through bilinear programming and incorporated the response time of the evacuees. They used the formulation by [170] for dynamic traffic assignment. They proposed a departure model with the objective of quickest evacuation time with one model departure time from origin and another model arrival time at the destination. [109] simulated the escape panic situation in which the clogging, jamming at widening, panic initiation and impatience are captured. This would help in validating an evacuation model developed without considering human factors.

However, the study was aimed to model the pedestrian behavior, injuries caused due to congestion, uncoordinated passing of bottlenecks, physical interactions among people. Thus careful scaling of parameters is required if we need to realize this in larger network with vehicular motions. The model would more serve as a tool for calibration than for establishing optimal routes of evacuation. The survey about human behavior in fire [31] gives a gist of the influence of evacuee behavior on models. The study caters specifically to the situation caused by fire and certain concerns raised, such as smoke effects, fire alarms, cannot be extended to a generic situation. This would help in a similar fashion as the previous model in setting standards for an evacuation model and features to be identified in a model. A model was developed by [81] to predict the response behavior of hurricane Andrew in Louisiana and they were found to be similar to hurricane Floyd in South Carolina. The response curve could be used to model response behavior in evacuation models. Since the study was performed and tested for behavior during hurricane, response behaviors during situations such as event management, nuclear accidents, fire hazards, and no-notice evacuations needs to be analyzed in order to assure its generic applicability. [47] provided the measures for efficiency of an evacuation plan. Among the various factors they considered the loading curve of the time taken for evacuees to decide to evacuate was considered as a major factor as this is crucial in determining the travel time of the individual evacuees. [148] suggested that the panic situation as result of a danger results in the evacuees to move at random before restoring order, especially when they want to find the family members before trying to reach safety. They proposed a two stage formulation establish household trip chain sequence. In the first phase the optimization model determines a meeting location for the family and in the second phase comprises of the trip assignment to evacuate people to safety. The strategies are later supplied to a simulator to study the effects of reassignment. It provides a different perspective to behavior modeling but not the only feature to be captured.

In case of a large scale evacuation system, where the travel time is much larger compared to the response time of the evacuees the error in the case of excluding the response time would be very minimal. However incorporating it might increase the complexity of the optimization model and hence decrease the computational efficiency. The implementation in two different phases, like that of contraflows, in order to overcome this problem could also be difficult as the behavior modeling dynamically changes the traffic flow. This is should be a another factor taken into consideration while developing a model. Sometimes behaviorial response is related closely to the situation and may be different at different times.

4.3.3 Dynamic origin-destination demands

[164] made a note that the most difficult obstacle before deploying DTA is estimating the time dependent demand. The demand between origin-and destination in an evacuation problem is typically subject to the environmental changes. For instance, an evacuation scenario which just requires the evacuees to reach safety might not route passengers to the initially designated destination, as this would compromise the evacuation time. This could be because of any unforeseen circumstances such as congestion, road blockage, lane reversals or unavailability of shelter etc. Most researchers have assumed to have origin destination data available a priori and they don't change over time. Later, the analysis was carried out on a static demand. [202] formulated an evacuation problem where the origin destination pairs are not fixed. Given a set of origin or destination, the objective is to ship all the commodities from the origin set to any subset of destination nodes. This is more intuitive in an evacuation process, where the safety of individuals is the priority. This could be achieved through a simple transformation in which a dummy super-sink node is added with zero cost and infinite capacity arcs from all other destinations. Given an directed graph $G(V, A)$ and set of origins I and destinations J with arc capacities $c(a), \forall a \in A$ and cost function $h(x(a))$. Let D_{ij} be the demand matrix between OD pair.

Now we have the following static demand evacuation problem.

$$\text{Minimize} \quad \sum_{\forall a \in A} h(x(a)) \tag{4–16}$$

s.t.

$$x(a) = \sum_{\forall r \in R} \delta_{ar} p_r, \forall a \in A \tag{4–17}$$

$$\sum_{\forall r \in R_{ij}} p_r = D_{ij}, \forall i \in I, j \in J \tag{4–18}$$

$$0 \le x(a) \le c(a), \forall a \in A \tag{4–19}$$

R is the set of all paths between all origins and destinations, p_r is the flow along the $r^t h$ path and δ_{ar} is indicator variable with value 1 is arc a is in path r and 0 otherwise. Constraint set (4–18) will be replaced by

$$\sum_{\forall r \in R_i} p_r = D_i, \forall i \in I \tag{4–20}$$

for the single destination.

This is another formulation for a time expanded static graph, having all the drawbacks, with regards to computational efficiency, mentioned earlier. The model when assuming independent evacuees as flow variables, assume that the different family members could reach different destinations. As a path based model has been presented the scalability of the model needs more examination as the analytical techniques using path or route based modeling needs to consider exponentially many paths available. [80] applied the sequential logit model to the model the dynamic traffic demand for evacuation problems with the underlying assumption that the demand varies with the evacuee behavior. The time horizon is discretized and the decision to evacuate at each time interval is modeled as a sequence of binary decisions. Later the probability that a household will evacuate at a specific time is calculated using random utility theory, which in turn is used to estimate time varying demand. The problem of updating origin-destination matrix based on the traffic link count has also been studied by

[187, 205]. A bi-level programming, where the OD matrix is updated at the upper level and the updated matrix is later used for DTA at the lower level. The formulation allows traffic flow according to the traffic conditions and they are brought under an entropy maximization framework. Another useful study in this line was by [203], where they compared two models of compliance and non-compliance to predetermine the route and destination assignments for evacuation using simulation software. [190] proposed a genetic algorithm to estimate the dynamic demand between the origin destination pair. The method is based on the fact that dynamic demand OD matrix depends on spatial and temporal variations of congestion or in other words the variation in demand based on the traffic count on a link. The congestion of link and hence a specific path from an origin to destination would cause the optimal solution to seek an alternate path causing an alteration in demand. The genetic algorithms tries to find the optimal OD demand matrix from the seed OD matrix provided as input.

4.3.4 Multimodal Transportation

Multimodal transportation are realistic in an emergency situation and it also considerably affects the evacuation efficiency. It is therefore essential to incorporate mode choices in the models in order to develop precise models. A formulation based on cell transmission was provided by [44] permits intermodal transportation with cars and buses for a single destination network. The formulation was a system optimal integer linear programming. Thus the corresponding flow problem is a bimodal transportation for a single commodity. The model assumes no pedestrian movements, but transit cells are considered where evacuees can move to another vehicle. The model is a very basic model and have several assumptions that makes it practically infeasible. For instance, the model assumes that there is an unlimited supply of buses and cars. Also, the models contains enormous amount integer variables and constraints. Addition of multicommodity flow constraints would make the model even more complex. [26] studied the influence of multimodal transportation on the evacuation efficiency in building complexes. They extended the cellular automata model, where space is divided into cells and time is discretized such that vehicles can move from one cell to another in one time step, to capture the interaction between in pedestrians and vehicle behavior. To overcome the computational complexity posed by the formulations in the optimization models investigators explored heuristic algorithms to solve the evacuation problem. [132] proposed a heuristic algorithm, HASTE, which provided a close approximation to the optimal solution that could be achieved through a linear program for the cell based dynamic traffic assignment problem. This overcomes the difficulty of the computational time spent on the overwhelming size of the problem at a price of approximating the optimal solution. The accommodation of multimodal transportation usually increases the precision of the model as it is more realistic and also the evacuation efficiency. However, it increases the complexity of the model as a whole and decreases the computational efficiency.

4.3.5 Miscellaneous factors

The key logistical issues during the aftermath of a disaster have been carefully analyzed by [114]. The response and recovery operations such as supply of food and medicine on reduced capacity network are discussed. The research was based on public accounts and interviews made during the field visits to Katrina impacted area. Another interesting study in the context of evacuation is to examine the vulnerability of the network and identify the insecure links in the network. It is important because it accounts for the connectivity between the origin and destination during evacuation. This may allow us to estimate the arc or link capacities of the network for evacuation. [191] measured the vulnerability of a network using a vulnerability index and aggregated them over all origin destination pairs. [177] provided a topological index to determine the depressiveness/concentration, which helps in identifying isolated districts and estimate the robustness of the road network in an emergency situation. The travel choice is tied to the vulnerability of the network. The change in the demands and flows owning to disruptions in the network was also studied by [45]. They formulated a travel demand model to derive a measure to assess the vulnerability of a transportation network. The measures are capable of valuating the changes in both demand and supply. [47] used agent based simulation technique, PARAMICS, to compare the staged and simultaneous evacuation. They concluded that for a general network topology there is no specific stratgey that results in better evacuation efficiency. However a high density population could be evacuated quicker over a grid structured network by staged evacuation.

4.4 Simulation Techniques

The large size and time over which the optimization models have to be implemented make them less suitable for immediate realizability. Simulation based approaches for evacuation strategies are widely adopted to overcome this practical difficulty. Also, simulation based models could be a tool to study the current plans without actually executing it. The evacuation plans generated by the analytical models discussed above could be simulated to identify inconsistencies in the model, thus serving as a good instrument for validation. The simulation models permits the designers and researchers to visualize the evacuation and hence it is more incisive compared to analytical models.

One level of breakdown of the simulation-based technique will be as macro, micro or meso traffic simulations. A microscopic traffic model captures the lineaments of individual vehicles, whereas a macroscopic traffic model provides a collective vehicle dynamics. Macroscopic simulations are similar to fluid flow study, where the estimates are based on a group of vehicles as a whole. Macroscopic simulations are not computationally expensive, but fails to adapt to random and rapid changes in the environment. Micro simulations is a diametric simulation technique, where the characteristics of individual vehicles are captured and thus able to predict and adapt to the changes in the model more efficiently than macroscopic simulations. The disadvantage of this technique is the computational expense encountered as the system size grows and more vehicles are added to the system. However, the micro-simulators are getting popular in the recent years with the increase in the computational capacity. A detailed note about macro and micro simulations and their relative advantages and disadvantages is discussed by [166].

4.4.1 Microscopic Simulation Techniques

Agent based simulations are also called as micro-simulators, as the agents or drivers involved in the system could be studied individually. In most traffic simulation studies there is a need for the simulation to be performed with more precision. For instance, vehicular interactions during congestion lane change or traffic signals at intersections could be dealt only in a microscopic model. [62] performed an agent based neighborhood evacuation study. The model has a nice property to study the disaggregate outputs such as the vehicle safety and travel times within zones rather than average travel time of entire system. However some of the assumptions are rigid such as departure times were assumed to be available and the destinations were preassigned. These assumptions may not be preserved in real time situation. The topology of the tested network was relatively simple compared to real time networks. It will be interesting to compare the computational efficiency when tested real time with features such queues, spill overs and congestion. The PARAMICS is a micro-traffic simulation software widely used in simulation. Many simulation based evacuation studies deployed PARAMICS for traffic simulation studies. [52] developed a simulation model using PARAMICS to simulate the evacuation of a high risk area in Santa Barbara called Mission Canyon Neighborhood. The model is used to analyze the possible evacuation scenarios changing traffic control, number of vehicles per household, opening of alternate exit and critical links in the network. Although these may not be the optimal ones, they are validated strongly based on empirical observations. The simulation provides the lower bound on the actual evacuation time as evacuee behavior is not considered and demand and arrival time estimates are not accurate. But the simulation results providing the best scenario in terms of the evacuation time might be indicative of the necessary factors to keep in mind in an emergency situation. In another study, [46] employed PARAMICS to compare the efficiencies of staged and simultaneous evacuation studied under free-flow and congestion situation. The study was performed for different network topologies. They made some simplistic assumptions like availability of route choices, destination information that prevents it from immediate realization. The findings, which include staged evacuation during congestion, would guide the design of new models. The aggregation of areas into zones is a technique employed to gain computational efficiency, but with a compromise in the precision. The default rules of PARAMICS for trip generation and destination and route choices were used for the simulation. They also provided a review of agent based simulation models and explained its advantage. They performed the simulation over three types of network namely ring, grid and an actual road network. The study indicated that staged evacuations based on zones are efficient than simultaneous evacuation. The models assume that the drivers are assumed to take the shortest path which leads to frequent congestion and hence the results may not reflect the best simultaneous evacuation time. CORSIM is another micro-simulation software used for traffic simulation. CORSIM is a combination of two other microsimulation models namely NETSIM and FRESIM, which are used for traffic simulation in surface streets and freeways respectively. [182] used CORSIM to create a transportation model for Birmingham. The generated model was then used test several emergency scenarios. They also made a brief review of micro-simulated evacuation models and discussed the vital features of CORSIM. Several response actions such as traffic diversions, altering signal timings, roadway clearance and access restriction were incorporated in the testing. The simulation model was then tested on different scenarios namely, a discrete traffic event, evacuation situation, and simulation of available response plans. The performance measures, which includes speed, queueing length and queueing time, were not justified to be the best set and they do not guarantee that the simulated results correspond to the most efficient solution corresponding to the assumptions made. The extensibility of the model is a concern for networks besides the networks examined under the same settings. The model will not have immediate realizability, but is useful in validating and calibrating models and the recommendations with respect the measures that were focussed in the study. CORSIM was also used to test the efficiency of contra flows or lane reversal under various evacuation scenarios [197]. The study was carried out in very plain settings, just to test the effectiveness of lane reversal of the link in a network. The study could serve only as a guide to evaluate the lane reversal efficiency rather than stand alone model to establish evacuation routes. The simplicity of the model makes it computationally efficient to obtain quick results. We discuss the demerits of establishing lane reversal strategies using simulation techniques in section 4.5.2. Another popular micro-traffic simulation software, VISSIM, is also used in evacuation studies. [201] used VISSIM for nuclear power plant accidents with dynamic traffic assignment and most desirable destinations. [188] provided a comparison of VISSIM and CORSIM with the demand estimates provided by Federal Emergency Management Agency (FEMA) to study the efficiency of lane reversals. Both these models indicated increased throughput and decreased queue with lane reversal implementation. Throughput at key nodes, queue lengths and average speeds were considered to be the efficiency measures of the evacuation plan. These efficiency measures were compared between a lane reversal plan and a do-nothing plan. The evacuation scenarios were considered essentially for a reversing only two vital links of the network. The lane reversals were considered for only the critical links of the network, but the rest of the model is heavily dependent on the simulators used by the model. As microscopic simulations keeps track of individual entities in the system it is computationally expensive and it was often used in small scale evacuation systems. Also, the details kept by these models makes them more complex. Their function logic gets complicated for operation and they often result in befuddling number of parameters that are tough to keep track. However, this limitation is being slowly subdued with the advent of faster computers.

4.4.2 Macroscopic Simulation Techniques

As mentioned earlier macroscopic models captures the vehicles and their activities more coarsely compared to its counterpart. Traffic is aggregated and the aggregated flow will be studied through model variables such as speed, density and flow rate. NETVAC [181] and MASSVAC [168] are two popular macro-traffic simulation software. NETVAC models radial evacuation from risk area, similar to evacuation during a nuclear accident. The travel demand is incidental in the model, since all the households within the risk area require evacuation. MASSVAC is a macroscopic simulation model developed for rural networks and was tested on a small rural network in Virginia for several loading curves. It comprises of a community module to define the boundary of the hazard, a population characteristics module to determine the population allocation spatially and an evacuation module that performs the actual traffic assignment. It has a simple input structure and trip distribution. Later versions of MASSVAC have been developed to incorporate complex features such as user equilibrium assignments. Oak Ridge Evacuation Modeling System (OREMS) is a popular macro-simulation evacuation simulator. Identifying bottlenecks and feasibility of evacuation, establishing evacuation routes and identifying alternate conrod strategies are some of the important features of OREMS. GIS interface, which is a useful add-on in the recent years is also available in OREMS. Moriarty et al., compared major macro-simulation softwares including OREMS, DYNEV and ETIS where the factors influencing the evacuation response were identified and several enhancements were suggested to improve the evacuation efficiency [146]. Most of these models were provided with the evacuation demand as an input to the model and the models are a coarse approximation of reality. The increase in computational capacity in the recent years are making microsimulators more attractive as they could also provide precision in performing localized studies and networks of reasonable sizes. However macrosimulators could be integrated with microsimulators or analytical models. They could serve as quick validators in a evaluating a developed model. Also, the very large scale network are still a little far fetched for analytical models and microsimulators and are mostly handled by macrosimulators. Macroscopic models are much simpler compared to the microscopic models, when it comes to to calibration and computation. This is only achieved through some shortcomings. They cannot be applied to instances which requires individual vehicular dynamics to be tracked. As the vehicle positions are not known in these models, the modern traffic control systems such as ramp metering, lane change maneuvers and a few other features of intelligent transportation systems cannot be captured by these models. Also, evacuation that requires modeling of human behavior cannot employ macroscopic simulation studies for the same reason. The accuracy is compromised by the aggregation of the traffic and network. They are more applicable in large scale evacuation with respect to time and space, whenever these shortcomings are less realized.

4.4.3 Meso-Simulation Techniques

A new generation of simulators called meso-simulators are popular in the recent days. They combine the pros and cons of the micro and macro simulators. The meso-simulators clubs the vehicles into packets or platoons which are then simulated as separate entities. [68] employed CEMPS, a meso-simulator, to develop a spacial decision support system. This is very practical model in the sense that the decision support system integrates the real time data obtained from geographical information system (GIS) to make traffic decisions and a object oriented simulation comprises of decision modeling and dynamic analysis components. CEMPS provides a convenient framework that permits this communication. The model does not sufficiently clarify the computational and economic concerns that may arise due to the increase in the size of evacuation network. [91] provided a meso-micro scale simulation study using SmartCAP that allows monitoring and studying of aggregate traffic flow behavior such as density, flow, and velocity and integrated it with a micro-simulator, SmartAHS, which is used record the individual details. The integrated simulator consists of "window" of the micro-simulator that communicates with the meso-simulator. Essentially the vehicles are micro-simulated and the meso-traffic flow characteristics such as velocity, flow rate, etc. are preserved. The claimed boost in the computational speed is pertinent to specific situations by the use of meso-simulators. More precisely, the gain will be linear in terms of the level of aggregation of vehicles into packets. This still cannot justify the phasing out of macrosimulators or microsimulators. However, the level of aggregation will help in achieving the capability of a microsimulator to the desired accuracy. DYNEMO [163] proposed another mesoscopic traffic flow model that was developed, where unit of a traffic flow is an individual vehicle unlike packets. The motion of the vehicles is determined by their link's traffic density. The function that gives the relation between traffic density and speed is provided as an input to the model.

4.4.4 Integrated Techniques

Simulation techniques could be used in conjunction with analytical models in order to gain the best out of the optimization and simulation techniques. The evacuation plans that are generated through the optimization techniques could be a good lower bound on actual evacuation time, thus these plans could serve as a candidate for simulators to identify the discrepancies in the model, foresee the requirements and capture the features that were not incorporated in the optimization technique. [180] provided a bi-level simulation based approach to solve the evacuation problem. At one level the time dependent route assignments are determined and at another level a dynamic loading problem is solved and the output is later aggregated. The time dependent route assignment is solved using the method of successive averages and the traffic demand simulation is done using DYNASMART-P to estimate the vehicle trip times and link travel times, thus achieving a system optimal schedule. The dynamic traffic assignment is implicity handled by DYNASMART-P, relieving the optimization model. The model is computationally efficient but the gain in speed could be softened by the degree of accuracy of the heuristic with the increase in problem size and the complexity due to new features when extended to other situations. [207] presented an evacuation system for ocean city integrating optimization and simulation techniques. An evacuation plan is generated by the optimization module and the plan is revised by the results of simulation evaluation. The pros and cons of the plans established were discussed. The model provides route choice options that permits users to deviate from the available or pretested plans, but the model assumes the availability of preestimated or forecasted demands. The formulation is based on cell transmission model proposed by [65] with necessary flow conservation and demand constraints. The result of this technique is a set of candidate plans, which could later be finalized using calibrated simulators. The results were then evaluated with a microscopic simulation program. This cell-based formulation makes the model more complicated in terms of the size and complexity when dealt over a larger system and so as the number of plans that have to be stored and tested in the next phase. The method is comprehensive compared to an integrated technique that produces one single solution, but makes it computationally unattractive. The candidate set could be a limited set to overcome this undesired effect. A similar two level optimization method for evacuation of the ocean city was proposed by [134]. At a broad level, they maximize the throughput at the higher level in terms of the number of vehicles being evacuated and minimize the travel time at the lower level. The cell-based formulation was made by accommodating cells of varying sizes in order to decrease the complexity of the optimization model. [135] proposed another integrated technique in which the cell based optimization model was used to formulate the demand constraints and the flow and storage capacity constraints. The result of the optimization module is then fed as the input to the microscopic simulator, CORSIM, which models real time operational constraints and driver behavior that were not captured in the optimization model. The evacuation systems proposed were modeled with static demands. These integration techniques are very useful for a number of reasons. The evacuation plan could be validated through simulation and hence can be reliable. The computationally expensive task could be simulated and other operations could be optimized and thus they attempt to seek an optimal balance between precision and speed.

4.5 Significant Features in Simulation techniques

4.5.1 Reviewed Features

[104] identified 22 evacuation models and assessed them by grouping them based on four different perspectives, namely enclosure i.e the fineness or coarseness of a network, population perspective where actions are taken based on individuals or by groups, behavioral perspective based on how the occupants react to the environment and the nature of model applications. [178] provided a survey of simulation models for emergency evacuation. They made a classification based on modeling approach, namely flow based, agent-based and cellular automata and discussed their relative advantages from the perspective of evacuee behavior. A flow based modeling comprises of nodes representing the structures or places to and from the evacuees have to be moved and arcs mapping to the hallways, roads, etc., that links these nodes. Since the individual characteristics are not emphasized in this modeling the practical realization of these models are impeded. However, if these models were used in conjunction with analytical models could be very powerful tool in terms of both speed and precision. In cellular automata the evacuation space is discretized into cells or grids with specific capacity and the entities flow from one cell to another. The models based on cell based transmission could be very accurate. [8] analyzed the 32 different micro-simulation models and compared the features available in the models. Scale of application, i.e the size of the network that the model can handle was one of the features that was discussed in the papers. A detailed statistics of the objects and phenomenon that the models included was provided. Queue spill backs, weaving, incidents and commercial vehicles were by and large modeled. Indicators of objectives showed that the speed, travel time, congestion and queue length were widely adopted by the models and indicators such as comfort and performance were rarely used. Analysis of telematic functions and interface were also discussed. A comprehensive survey

on simulation studies was provided [169] analyzed various simulation software that are available and presented all the components considered for evacuation by these software. The categories of evaluation of the models were modeling, behavior, operations and hazards. Most of these features have already been discussed. This analysis provided a generalized framework for simulation studies of emergency evacuation.

4.5.2 Lane reversals and Traffic Control

Contra flows or lane reversals are being considered in the simulation models just like in optimization models. The recent trend and studies have indicated that there is a significant decrease in evacuation time with the implementation of lane reversals. A more efficient way of controlling the traffic is by employing traffic personnel. [197] studied traffic control by capturing these scenarios and simulating them. The travel time was significantly less for the instances with lane reversal. [190] undertook a research to study the effect of contraflow implementations in evacuation in New Orleans. They employed CORSIM to simulate the freeway configuration and two other scenarios with contraflow implementation. The study carried out various loading configurations of major highway permitting contraflows. The experiments were used to demonstrate the benefits of contraflow. A significant improvement in the measures of effectiveness, namely travel time and average speed was achieved for the configurations permitting contraflow. In most cases the contra flows are assume complete lane reversals, that is, the entire capacity of the road is switched towards the destinations. There are a couple of reasons for holding a capacity in either directions. In case of a large network, a link of the network in either direction could be used in a route to reach the destination. In case of a link failure, where alternate paths are required, the reversed links can no longer be part of an alternate path to the destination. If this was not the case and even if a link cannot be a part of route from origin to destination, these arc capacities could be used in a routing problem where empty buses have to be routed back to pick up points. Most of the papers currently assume that there will be enough number of buses to support the evacuation by making just one trip. Thus studies that identify the amount of capacities that has to be reversed within highway rather than complete reversals are limited. Simulation studies are not the best way to study the effects of contraflow as they need a input plan which could be calibrated or validated, but cannot be a tool to identify the links or the amount of capacities that has to be reversed. Analytical tools could be employed for this purpose and the plans could later be tested and verified in simulations.

4.5.3 Dynamic Demand Estimation

In a large scale evacuation the evacuation distance is large and assuming static traffic conditions is not precise. DTA allows different traffic conditions to be included in the simulation. It captures the complex dynamic demand pattern that arises due to congestion, queues, spill back and delays. There will be a significant difference in the evacuation efficiency between static and dynamic demand. The factors that results in the difference has been discussed before. The dynamic demand estimation through a simulator may not be as difficult as the contraflows. [10] provided a simulation model with a demand predictive capabilities implemented as a part of DynaMIT simulator. The model generates a pre-trip demand based on the historical demand and two systematic deviations based on daily demand fluctuations and driver response behavior. The pre-trip demand is used to establish a disaggregate pre-trip travel decisions for the drivers. Then a behavioral model is employed to exploit the available real time data and alter route decisions. Then a disaggregate origin destination matrix is generated based on the traffic count on the links of the network, which is then aggregated to estimate the new demand. The model takes into account various factors including behavior pattern, daily demand fluctuations and a random error component, thus giving more reliability to the model. The model employs a time discretization for determining intervals between which the distance updates are made. This makes the model computationally expensive based on the length of the interval. The question of consideration will be the capability of the model to handle other features in addition to the dynamic demand estimation. This model was just an extension of the work by [17]. The demand estimation through update of historical demand based on driver behavior and was further enhanced by systematic deviations in the newer models. A similar method, QUEENSOD, [3] involves a seed matrix similar to the pre-trip demand matrix based on historical demand. The seed matrix establishes traffic counts and a micro-simulator is used to establish the traffic routes based on the estimate and then the seed matrix is altered to reduce the error between the estimated and observed traffic counts.

4.5.4 Miscellaneous Factors

The number vehicles per household in micro-traffic simulation may be important as this increases the demand. The use of critical links is a major factor in evacuation as it is important to ensure that the critical links are not congested, which would result in heavy traffic disruption. Incident management is a minor feature that models focuses on the need for alternate routes in case of accidents and estimation of traffic personnel for diversion and lane control. [145] studied the impact of staged evacuation efficiency, where they considered six different scenarios and and they carried out simulation on a representative traffic network that resulted in successful staged evacuations. The scenarios comprised of combinations of shifting the departure times of evacuations. This resulted in increase in the total number of trips, but prevented congestion and queues. They concluded that the departure time shifting and total number of trips made had a positive impact on the clearance time.

4.6 Conclusions

The models currently available in the literature are usually customized for the evacuation of specific regions in geography tailored to its needs. The models have their relative advantages and disadvantages, but precise to their needs. It is rather tedious to generate a unified model that could be used in all situations. Inclusion of several features impacts the complexity of the model and hence the computational speed. On the other hand, simplifying a model would compromise the precision of the model. However, the models have some common overlapping features that we highlighted in this report. We saw that the hybrid models could be reasonably accurate and precise by exploiting the relative advantages of simulation and optimization techniques. This chapter provided broad classification of evacuation model based on simulation or optimization methods. Further, it identified the factors considered by these optimization models and commented on the approaches. We highlighted the features that will have a significant effect on the travel time namely intermodal transportation, dynamic traffic demand estimation and contra flows or lane reversals in case of a wide area evacuation. Models incorporating intermodal transportation and contra flows demonstrated the improvement in evacuation efficiency compared to the traditional models. Also, static demand model have become obsolete and dynamic demand is necessary for practical realization of the models. Some areas that needs attention are optimization problems to establish alternate evacuation paths for incident managements. Critical node detection and traffic management on critical links are studies that might improve the efficiency of the evacuation and also might give an indication of the necessary links that we could focus on contraflows. Also heuristic exploration of optimization techniques could significantly reduce the computational speed. Research in the field of clustering of nodes and zonal division of network is very limited. This might shed light in performing evacuation over a smaller aggregated network helping in computational efficiency.

CHAPTER 5
NETWORK FLOW PROBLEMS WITH LANE REVERSALS

5.1 Introduction

We study contraflow network problems, wherein we try to maximize flow in a graph while permitting direction reversals of an arc, resulting in a capacity increase in the direction of switch. The applications are realized in an emergency situation, where people have to be 'evacuated' from a specific area; *i.e.* a football stadium after a game, a city expecting a flood or hurricane, a zone where an unexploded ordnance device has been found, or a region which has been attacked by terrorists. In most of these cases, the evacuees are expected to leave the area of risk, the source(s), towards a safer place, the sink(s). A flow towards the source is undesired during most of these scenarios and we do not expect the evacuees to go in this direction. As a direct consequence, all the arcs that are not a part of any path from the source node(s) to the sink(s) might be left unused. One can even encounter idle arcs during certain scenarios, such as managing a football event, wherein we do have some amount flow towards the source. These idle arcs could be used to increase the efficiency of evacuation by reversing their directions. The scenarios involving in partial lane reversal capability could be captured with appropriate graph transformation. We discuss several scenarios that may arise during the reconfiguration, which includes permitting only a subset of arcs to be reversed, imposing a switching cost to the arcs involved in the reversals.

There are very few optimization techniques in the literature handling arc reversals. Kim and Shekar [121] proposed a simulated annealing procedure for this problem and provided empirical results. They also provide a sketch of the proof that the problem is NP-complete. A tabu-based heuristic was proposed by Tuydes and Ziliaskopolos [193] for the problem. They focus their study on a specialized version, where they permit lane reversals with partial capacities. Hamza-Lup et al. [106] proposed a heuristic for this contraflow problem. These techniques and their pitfalls were discussed in [121]. A few other studies in the literature that are not analytical in nature were also proposed. They rely on simulation-based methods and decision support tools [190, 197].

In this chapter, we provide a detailed study of the arc reversal (or contraflow) problems with respect to their computational complexity. The motivation is to introduce the problems formally to provide a basis for further research in this area. As the applications are mainly realized during emergency situations, the dynamic flow problems are of principal interest, but we study static cases as presuppositions and also for the sake of completeness of the study. In section 5.2, we provide a brief background of the network flow problems and explain the terminology used in the rest of the chapter. We then provide a discussion of static flow problems in Sec. 5.3. A polynomial time algorithm through a graph transformation is introduced for the static maximum flow problem with arc reversal capability. The result is evident and it is useful in Sec. 5.4.1 in showing that the dynamic maximum contraflow problem, with single source and single sink, is polynomially solvable. We show in Sec. 5.4.2 that the the decision version of the multiple sources and multiple sinks version of the problem is NP-complete through a reduction from 3-SATISFIABILITY (3SAT). In Sec. 5.4.2.2, we show that the problem becomes NP-complete by having just two sources or sinks. In addition, we discuss the inability of the graph transformation that was employed earlier to provide feasible solutions. We finally show in Sec. 5.5 that the problem of finding the minimum total cost, incurred due to an arc switching cost, to identify the arcs to be reversed is NP-hard, even in the static case.

5.2 Background

The basic terminologies and definitions that are predominantly used in the network flows literature and that are essential for the rest of the chapter are explained in this section.

Definition 5.1 (Static feasible flow).

Given is a graph $G = (V, A)$ with capacities $c_e \in \mathbb{Z}^+$ for all arcs $e \in A$. A static flow, characterized by the function $f : A \to \mathbb{R}^+$, with value v, from $s \in V$ to $t \in V$ is feasible, if

$$f_e \leq c_e, \quad \forall e \in A \tag{5–1}$$

$$\sum_{(i,j)\in A} f_{i,j} - \sum_{(j,k)\in A} f_{j,k} = \begin{cases} v, & j = s \\ 0, & \forall j \in V \backslash \{s \cup t\} \\ -v, & j = t \end{cases} \tag{5–2}$$

We call node s as the 'source', node t as the 'sink' and rest of the nodes as 'intermediate' or 'transhipment' nodes.

Equation 5–1 ensures that the flow f_e along each arc $e \in A$ meets the capacity constraints; as we assume all lower bounds on the flow to be 0. In equation 5–2, the net flow out of s is v and t is $-v$. For all intermediate nodes it is 0 and is also referred to as flow conservation. The definition of a feasible flow generalizes in a natural way for the case of multiple sources and multiple sinks.

A sequence of distinct nodes $x_1, x_2, \ldots, x_n$ of a graph $G = (V, A)$ is called a chain if $(x_i, x_{i+1}) \in A, \forall i = 1, \ldots, n$. A chain is also referred to as a directed path. Let P be the set of all chains from s to t. We define another flow function, $h : P \to \mathbb{R}^+$, in terms of the flow along the chains from s to t. A feasible flow f with value v could be decomposed into a set of chains, P, from s to t, such that

$$v = \sum_{i=1}^{|P|} h_i \quad .$$

The process of obtaining flow along the chains this way is called as 'chain decomposition'. A more detailed account of these terminologies could be found in [4, 76].

In a dynamic graph or network $G = (V, A)$ each arc is associated with a travel time, $t : A \to \mathbb{R}^+$, besides the capacity function. The graph expanded over T time periods, $G^T = (V^T, A^T)$, is obtained by replacing each node by T copies and having nodes v_i^l and $v_j^{l+t_{i,j}}$ connected in $G^T = (V^T, A^T)$ if v_i and v_j are connected in G, for all $l = 0, \ldots, T - t_{i,j}$.

This concept of a feasible flow can be directly adopted to the dynamic case by ensuring that both equations 5–1 and 5–2 are satisfied for all discrete time steps. Hence, a feasible dynamic flow is a feasible flow in the time expanded graph with the value equal to the sum of the net flows out of all the T copies of s. For more details about time expanded graphs refer, for instance, to [4, Chapter 19.6].

5.3 Maximum Static Contraflow Problems

In this section, we provide a polynomial time algorithm solving the maximum contraflow problem in a static graph. The results presented in this section are very basic and straight forward. Nevertheless, we discuss them in detail as this helps us in developing the main results in Section 5.4.

Now, let us define define the maximum flow problem with arc reversal capability.

Definition 5.2 (Maximum Contraflow (MCF))**.**

Instance: *Given a directed graph $G = (V, A)$ with source $s^+ \in V$, sink $s^- \in V$ and capacity $c_e \in \mathbb{Z}^+$ on each arc $e \in A$.*

Question: *What is the maximum flow from node s^+ to node s^- if the direction of the arcs can be reversed?*

This problem is also called *maximum flow problem with arc reversal.* Consider now procedure P-MCF. In the first step, an auxiliary graph $\widetilde{G} = (V, \widetilde{A})$ is constructed. The transformation from the original graph G is obtained by summing the capacities of arcs (i, j) and (j, i). This allows us to reduce the MCF problem to the maximum flow problem on the transformed graph in step 2. Step 3 removes cycle flows in the transformed graph. This ensures that the constructed solution of the MCF problem in step 4 is well defined.

We have the prior knowledge that there exists an optimal flow to the maximum flow problem that does not have cycles. Thus, arcs on either direction will never be used in this flow for the maximum flow problem. This is the basic idea of procedure P-MCF that motivates the graph transformation given. This result is straightforward but we can realize its impact in Sec. 5.4.1.

Procedure Maximum Contraflow (P-MCF)

1. Construct the transformed graph $\widetilde{G} = (V, \widetilde{A})$ where the arc set is defined as

$$(i,j) \in \widetilde{A}, \text{ if } (i,j) \in A \text{ or } (j,i) \in A \quad ,$$

The arc capacity function $\widetilde{c}$ is given by

$$\widetilde{c}_{i,j} := c_{i,j} + c_{j,i} \quad ,$$

for all arcs $(i,j) \in \widetilde{A}$.

2. Solve the maximum flow problem on graph $\widetilde{G}$ with capacity $\widetilde{c}$.
3. Perform flow decomposition into path and cycle flows of the maximum flow resulting from step 2. Remove the cycle flows.
4. Arc $(j,i) \in A$ is reversed, if and only if the flow along arc (i,j) is greater than $c_{i,j}$, or if there is a non-negative flow along arc $(i,j) \notin A$ and the resulting flow is the maximum flow with arc reversal for graph $G = (V, A)$.

End procedure

Theorem 5.1 (Proof of correctness). *Procedure P-MCF solves the maximum flow problem with arc reversal for graph $G = (V, A)$ optimally.*

Proof. The proof consists out of two steps. First, we show that any solution of the procedure P-MCF is feasible for $G = (V, A)$. Second, we show its optimality.

For feasibility, we only have to show that step 4 in the algorithm is well defined; *i.e.* not both arcs (i,j) and (j,i) have to be switched. However, this is ensured by step 3. The optimal solution after the flow decomposition results in a set of paths from source to sink and a set of cycles with positive flows. After the flow decomposition we could cancel the positive flows along all cycles and ensure that there is no flow along any cycle. This ensures that there is either a flow along arc (i,j) or (j,i), but never on both arcs. Hence, the resulting flow from step 4 is a feasible flow with arc reversal for graph $G = (V, A)$.

Now, we prove that the resulting flow is also optimal. Note that any optimal solution to the maximum flow problem with arc reversal on graph $G = (V, A)$ is also a feasible solution to the maximum flow problem on the transformed graph $\widetilde{G} = (V, \widetilde{A})$. As the amount of flow send from s to t is not changed in steps 3 and 4, the resulting flow is an optimal solution to the maximum flow problem with arc reversal on graph $G = (V, A)$. $\square$

The running time of procedure P-MCF is dominated by solving a maximum flow problem in step 2 and by the flow decomposition in step 3; as steps 1 and 4 can be done in $O(|A|)$. Let us denote the running time for solving the maximum flow problem by $S_1(|V|, |A|)$ and for the flow decomposition problem by $S_2(|V|, |A|)$. Then, the running time of procedure P-MCF is given by $O(S_1(|V|, |A|) + S_2(|V|, |A|))$. Using the highest-label preflow-push algorithm leads to $S_1(|V|, |A|) = O(|V|^2 \cdot \sqrt{E})$, [48]. The flow decomposition can be done, for instance, in $O(|V| \cdot |E|)$, [4]. This proves the following theorem.

Theorem 5.2 (Running time). *Procedure P-MCF solves the maximum contraflow problem in strongly polynomial time.*

We are now able to extend the result above to the case of multiple sources and multiple sinks. This problem is also called *maximum transshipment contraflow (MTCF) problem.*

Corollary 5.1. *The static version of the maximum contraflow problem with multiple sources and multiple sinks is polynomially solvable.*

Corollary 5.1 can be realized through a simple reduction. Let S^+ and S^- be the set of sources and the set of sinks, respectively. Then, add a 'super-source' u^+ and a 'super-sink' v^- together with the arcs (u^+, s^+), for all $s^+ \in S$, with arc capacities equal their respective surplus and (s^-, v^-) for all $s^- \in S^-$ with their arc capacities equal their respective deficits. For more details, refer to [76].

Recognize that we basically show in this section that the maximum contraflow problem is equivalent to a maximum flow problem on an undirected (modified) graph. This could be seen in the graph transformation provided in step 1 of procedure P-MCF with arcs having same capacities in either directions.

5.4 Maximum Dynamic Contraflow Problems

In this section, we discuss the *maximum dynamic contraflow (MDCF) problem*. The maximum dynamic flow problem was studied by Ford and Fulkerson [76], where they try to maximize the flow sent from source to sink, within a given time horizon T. Unlike the static case, in the dynamic network flow problem the flow over an arc can be repeated over time. Ford and Fulkerson proved that this problem is equivalent to solving a minimum cost flow problem with the arc costs as travel times on the arcs. Then the optimal flow on the arcs from source to sink is decomposed into a set of paths or chains. These chains are then *temporally repeated* over time to obtain the required dynamic flow. In other words, there is always a temporally repeated chain flow that is equivalent to the maximum dynamic flow. Let us assume there are P paths obtained from the chain decomposition of the optimal minimum cost flow. Then the maximum dynamic flow is given by

$$\sum_{i \in P} (T + 1 - t_i) h_i \quad ,$$

where h_i is the flow along the i^{th} path and t_i is the time taken to travel the i^{th} path. In this section, we first study the single source and single sink dynamic flow problem having arc reversal capability. We provide an algorithm employing a similar kind of graph transformation as procedure P-MCF and discuss its proof of correctness together with its worst case running time analysis. This implies that the *quickest contraflow (QCF) problem* is also polynomially solvable. In the *quickest flow problem*, the time to send a given flow from source to sink is minimized. Burkard et al. [33] gave a strongly polynomial time algorithm for this problem.

Hoppe [117] studied the multiple sources and multiple sinks version of this problem, also called the *quickest transshipment problem*, where they minimize the time taken to send the supply at the sources to the sinks satisfying their demands. In static network flows, the multiple sources and multiple sinks are handled by adding a 'super-source' and a 'super-sink'. Then they are connected to the sources and sinks respectively, see

Corollary 5.1. However, this solution procedure is not applicable in a dynamic case anymore. For the same reason, the dynamic contraflow problem with multiple sources and multiple sinks is NP-complete. We provide an example illustrating this together with a proof of its NP-completeness.

5.4.1 Single Source and Single Sink

Let us extend the MCF problem of Sec. 5.3 to the dynamic case.

Definition 5.3 (Maximum Dynamic Contraflow (MDCF)).

INSTANCE: *Given a directed graph $G = (V, A)$ with source node $s^+ \in V$, sink node $s^- \in V$, capacity $c_e \in \mathbb{Z}^+$ and transmission time $t_e \in \mathbb{Z}^+$ on each arc $e \in A$ with $t_{i,j} = t_{j,i}$ if $(i, j), (j, i) \in A$, and an overall time horizon $T \in \mathbb{Z}^+$.*

QUESTION: *Determine the maximum amount of flow that can be send in T units of time from source s^+ to sink s^-, if the direction of the arcs can be reversed at time 0. Note: In this case, if we choose to switch an arc, it remains switched from time 0 to T. The case where we allow switching of arcs back and forth in time is trivial as the quickest transhipment contraflow problem, with this assumption, reduces to the quickest transhipment problem through the graph transformation suggested in procedure P-MDCF and hence is polynomially solvable.*

Definition 5.3 states that in a MDCF problem, the graph is allowed to be asymmetric with respect to the arc capacities. However, whenever both directions of an arc are included in the graph, then the traveling time of these two arcs must be the same. This assumption implies that the switching of an arc only changes the capacities of the arcs but does not alter their traveling time.

The concept of temporally repeated flows is very fundamental for the maximum flow problem with single source and single sink. Our algorithm for solving the MDCF problem is mainly based on this concept. Hence, let us repeat the definition given by Ford and Fulkerson, [76, page 147].

Definition 5.4 (temporally repeated).

A dynamic flow which can be generated by repeating chain flows of a static flow in graph G is called temporally repeated flow.

The following theorem reveals the usefulness of temporally repeated flows in the context of single source and single sink network flow problems, [76, Theorem 9.1].

Theorem 5.3. *There is a temporally repeated dynamic flow that is maximal over all dynamic flows for T periods.*

The flow to be temporally repeated could then be determined by just solving a minimum cost flow problem. Let us denote its running time by $S_3(|V|, |A|)$. Using, for instance, the minimum mean cycle-canceling algorithm leads to a strongly polynomial running time of $O(|V|^2 \cdot |E|^3 \cdot \log(|V|))$, [89].

Before we proceed to the next lemma, we need to know that utilizing the concept of time expanded graphs in a solution algorithm leads to a pseudo-polynomial running time. In this case, the running time depends on $|T|$, rather than $\log(|T|)$ which would then lead to a weakly polynomial running time. Nevertheless, we use the concept of time expanded graphs in Theorem 5.4.

Consider now procedure P-MDCF. We show in Theorem 5.4 that it solves the MDCF problem correctly. The main differences of procedure P-MCF and P-MDCF is given in step 2. For the dynamic problem, we need temporally repeated flows. This ensures that only one of the arcs (i, j) or (j, i) is used in the flow. This enables us to use the same flipping rule for the arcs as in procedure P-MCF.

In order to show the correctness of procedure P-MDCF, we need the following lemma.

Lemma 5.1. *The maximum amount of flow in the single source and single sink maximum dynamic contraflow problem for graph $G = (V, A)$ is less than the optimal flow in the maximum contraflow problem for the corresponding time expanded graph $G^T = (V^T, A^T)$.*

Proof. The result follows directly from the observation that every feasible flow to the maximum dynamic contraflow problem has an equivalent feasible flow to the maximum contraflow problem of the time expanded graph. □

Procedure Maximum Dynamic Contraflow (P-MDCF)

1. Construct the transformed graph $\widetilde{G} = (V, \widetilde{A})$ where the arc set is defined as
$$(i,j) \in \widetilde{A}, \text{ if } (i,j) \in A \text{ or } (j,i) \in A \quad .$$
The arc capacity function $\widetilde{c}$ is given by
$$\widetilde{c}_{i,j} := c_{i,j} + c_{j,i}$$
and the traveling time is
$$\widetilde{t}_{i,j}\ (= \widetilde{t}_{j,i}) := \begin{cases} t_{i,j}, & \text{if } (i,j) \in A \\ t_{j,i}, & \text{otherwise} \end{cases} \quad ,$$
for all arcs $(i,j) \in \widetilde{A}$.
2. Generate a dynamic, temporally repeated flow on graph $\widetilde{G}$ with capacity $\widetilde{c}$ and traveling time $\widetilde{t}$.
3. Perform flow decomposition into path and cycle flows of the flow resulting from step 2. Remove the cycle flows.
4. Arc $(j,i) \in A$ is reversed, if and only if the flow along arc (i,j) is greater than $c_{i,j}$, or if there is a non-negative flow along arc $(i,j) \notin A$ and the resulting flow is the maximum flow with arc reversal for graph $G = (V, A)$.

End procedure

Please note that Lemma 5.1 holds good for more than one source and one sink. However, in general, equality holds only for the case of a single source and a single sink, as we will see in the following theorem. We are now ready to prove the correctness of procedure P-MDCF.

Theorem 5.4 (Proof of correctness). *Procedure P-MDCF solves the maximum dynamic contraflow problem for graph $G = (V, A)$ optimally.*

Proof. The concept of this proof is similar to the proof of Theorem 5.1. First, we prove that all the steps in procedure P-MDCF are well defined and result in a feasible solution. Second, we show optimality.

For feasibility, the proof follows directly from the fact that the constructed flows are temporally repeated and hence, there is only a flow in one direction of two nodes, and never in both directions at the same time as well as at different time periods. After canceling the flows along the cycles, we have flows either on arc (i, j) or on (j, i) but not on both. This ensures that the flow is less than the reversed capacities on all the arcs at all time units. This also ensures the feasibility. In other words, we now have established the fact

$$[G = (V, A)]_{MDCFopt} \geq [\widetilde{G} = (V, \widetilde{A})]_{MDFopt} \quad ,$$

by the argument that every feasible flow of the dynamic flow problem in the transformed graph $\widetilde{G} = (V, \widetilde{A})$ is feasible to the maximum dynamic contraflow problem in the graph $G = (V, A)$. Our proof is complete if we show that

$$[G = (V, A)]_{MDCFopt} \leq [\widetilde{G} = (V, \widetilde{A})]_{MDFopt} \quad .$$

To see this, first note that the maximum contraflow in graph $G^T = (V^T, A^T) \geq$ maximum dynamic contraflow in graph $G = (V, A)$, from Lemma 5.1. Hence we have,

$$[G = (V, A)]_{MDCFopt} \leq [G^T = (V^T, A^T)]_{MCFopt} \quad .$$

By Theorem 5.1 we have that the maximum contraflow problem in graph $G^T = (V^T, A^T)$ is equivalent to the maximum flow problem in the graph $\widetilde{G}^T = (V^T, \widetilde{A}^T)$, where the arc set $\widetilde{A}^T$ is defined as

$$(i, j) \in \widetilde{A}^T, \text{ if } (i, j) \in A^T \text{ or } (j, i) \in A^T \quad ,$$

and the arc capacity function $\widetilde{c}$ is given by

$$\widetilde{c}^t_{i,j} := c^t_{i,j} + c^t_{j,i} \quad .$$

Thus,

$$[G^T = (V^T, A^T)]_{MCFopt} = [\widetilde{G}^T = (V^T, \widetilde{A}^T)]_{MFopt} \quad .$$

By Theorem 5.3, the maximum flow in the time expanded graph $\widetilde{G}^T = (V^T, \widetilde{A}^T)$ can be obtained by a temporally repeating a chain flow of a static graph $\widetilde{G} = (V, \widetilde{A})$. Hence we have the fact,

$$[\widetilde{G}^T = (V^T, \widetilde{A}^T)]_{MFopt} = [\widetilde{G} = (V, \widetilde{A})]_{MDFopt} \quad .$$

□

Just like procedure P-MCF, running time dominating are steps are 2 and 3 for procedure P-MDCF. This results in a worst case running time of $O(S_2(|V|, |A|) + S_3(|V|, |A|))$; which is strongly polynomial.

Theorem 5.5 (Running time)**.** *Procedure P-MDCF solves the maximum flow problem in strongly polynomial time.*

For given excess b, the *quickest contraflow problem* determines the minimum time horizon T needed by any feasible flow.

Corollary 5.2. *The quickest contraflow problem can be solved in a strongly polynomial time.*

One way to realize Corollary 5.2 is through the work by Burkard et al. for the quickest flow problem, [33]. First, obtain an upper bound on the quickest time and second, perform a binary search by repeatedly solving the minimum dynamic contraflow problem. Such a bound can be obtained in polynomial time, for instance, by computing a path from source to sink and temporally repeating flow along the path until all supply at the source is sent to the sink. However, this leads to a weakly polynomial algorithm. A strongly polynomial algorithm could be obtained through a parametric search suggested by Megiddo [138, 33].

5.4.2 Multiple Sources and Multiple Sinks

Let us start with the definition of the multiple sources and multiple sinks version of the MDCF problem.

Definition 5.5 (Dynamic Transsphipment Contraflow (DTCF))**.**

INSTANCE: *A directed graph $G = (V, A)$, a set of sources $S^+ \subset V$, a set of sinks $S^- \subset V$, arc capacities $c_e \in \mathbb{Z}^+$ and transmission time $t_e \in \mathbb{Z}^+$ for each arc $e \in A$ with $t_{i,j} = t_{j,i}$ if $(i, j), (j, i) \in A$, and an overall positive integer time bound T.*

QUESTION: *Is there a feasible dynamic flow within time horizon T, allowing each arc to be revered once at time 0?*

Note that the DTCF problem is a decision problem corresponding to the maximum dynamic contraflow problem with multiple sources and multiple sinks.

5.4.2.1 DTCF is NP-complete in the strong sense

In this section, we proof that the DTCF problem is NP-complete. A sketch of the proof outline was given in [121]. However, we provide a rigorous proof. Also, the proof has some differences though we provide the reduction from the same problem, *3SAT*, [82, page 46]:

Definition 5.6 (3SAT)**.**

INSTANCE: *Collection $C = \{c_1, c_2, \ldots, c_m\}$ of clauses on a finite set U of variables such that $|c_1| = 3$ for $1 \leq i \leq m$.*

QUESTION: *Is there a truth assignment for U that satisfies all the clauses in C?*

3SAT is known to be NP-complete in the strong sense, see [82, Theorem 3.1].

For an instance of 3SAT, construct a graph $G_{3SAT} = (V, A)$ for DTCF as follows. For each clause c_i we have one source node c_i^+ with a surplus of 1. Each variable $u_j \in U$, is presented by six nodes in the graph: two for each literal, named u_j^1, u_j^2, $\overline{u}_j^1$ and $\overline{u}_j^1$ respectively, one source node with surplus 1, d_j^+, and one sink node with deficit -1, d_j^-. Finally, there is one node with deficit $-|C|$, named s^-. This sums up to $|V| = |C| + 6|U| + 1$ nodes. Each clause node c_i^+ is connected to the nodes with superscript 1 representing its literals, taking 3 time units. For each j, the node u_j^1 is connected to its copy, u_j^2, with transshipment time of 1. Nodes d_j^+ are connected to u_j^2 and $\overline{u}_j^2$ with transshipment time 1, while nodes d_j^- are connected to u_j^1 and $\overline{u}_j^1$ having a transshipment time of 1. Finally, each second copy (superscript 2) of the literals is connected to the sink s^- taking a time of 1. All arcs have a capacity of $|C|$. This leads to $|A| = 3|C| + 8|U|$ arcs in graph G_{3SAT}. One such graph transformation is shown in Fig. 5-1.

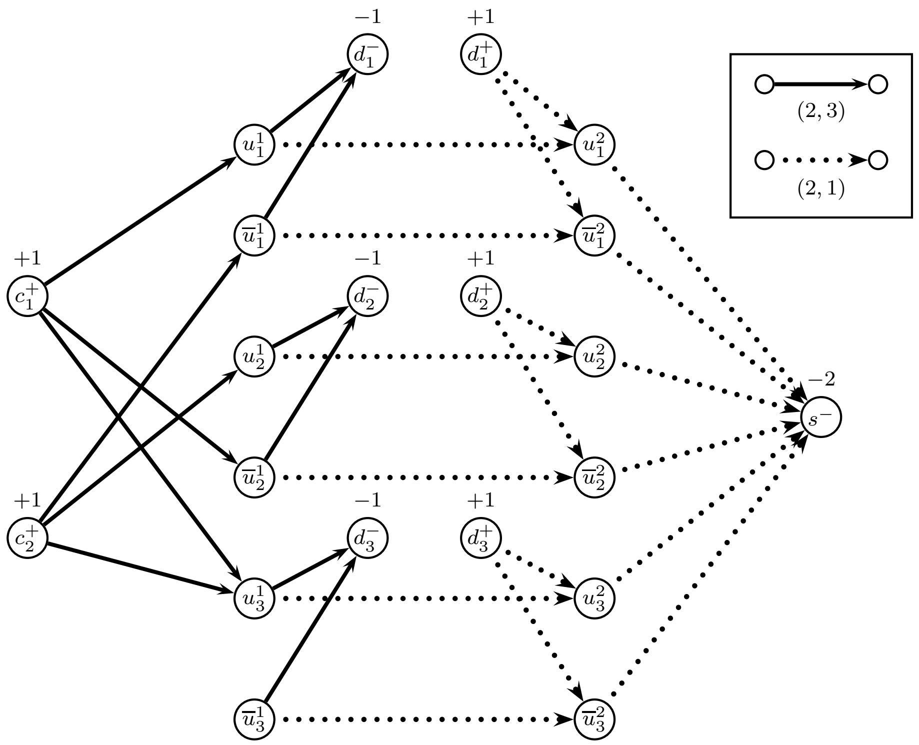


Figure 5-1. Transformed graph G_{3SAT} corresponding to 3SAT instance $C = \{\{u_1, \overline{u}_2, u_3\}, \{\overline{u}_1, u_2, u_3\}\}$

The proof of the validity of the transformation is based on the following key observation.

Lemma 5.2. *In any feasible flow f in the graph G_{3SAT} within time $T = 5$, there is a flow of value 1 from node d_j^+ to node d_j^-, for all j.*

Proof. Let us fix index j and assume that the flow to node d_j^+ is integral. If the flow to node d_j^- does not come from node d_j^+, then it can only come from exactly one of the nodes c_i or d_k^+ with $k \neq j$. However, in both cases, the flow arrives at node d_j^- earliest at time 6, or 7 respectively. This proofs the lemma for the case of integer flows. The case of fractional flow is similar: If some fraction of the flow to node d_j^- comes from a different node then d_j^+, then the flow arrives after time $T = 5$. □

Lemma 5.2 implies that for a feasible flow, at least one of the arcs (u_j^1, u_j^2) or $(\overline{u}_j^1, \overline{u}_j^2)$ has been switched for all j – with other words, at most one of the two arcs (u_j^1, u_j^2) and

$(\overline{u}_j^1, \overline{u}_j^2)$ keep their direction in any feasible flow with time bound $T = 5$. Now, we are able to proof the following lemma.

Lemma 5.3. *An instance of 3SAT is a 'YES' instance, if only if the transformed graph G_{3SAT} is a 'YES' instance for DTCF with overall time bound $T = 5$.*

Proof. "⇒" Let 3SAT have the feasible assignment $u_j = a_j$ for all variables, with $a_j \in \{0, 1\}$. Then, reverse the arcs (u_j^1, u_j^2) if $a_j = 0$, and reverse arc $(\overline{u}_j^1, \overline{u}_j^2)$ otherwise. Now, for all j, send one unit of flow from d_j^+ to d_j^- along the reversed arc. As only one of the arcs (u_j^1, u_j^2) or $(\overline{u}_j^1, \overline{u}_j^2)$ has been switched, we can send flow from any of the nodes c_i^+ through any non-switched arc, dependent on the assignment of the literals. This leads to a feasible flow for DTCF within time $T = 5$.

"⇐" We have to show, that any feasible flow f for DTCF needing (at most) 5 units of time leads to a 'YES' instance of the 3SAT. We assign the following value to each variable $u_j \in U$ as

$$u_j := \begin{cases} 0, & \text{if arc } (u_j^1, u_j^2) \text{ is reversed in flow } f \\ 1, & \text{otherwise} \end{cases} . \tag{5–3}$$

We have to show that this is a satisfying truth assignment for the 3SAT instance. Now, assume that clause c_i is not a truth assignment. One unit of flow is send from node c_i^+ to node s^- through one of the nodes u_j^1 or $\overline{u}_j^1$ with $u_j \in c_i$ or $\overline{u}_j \in c_i$. Notice that this flow cannot go through any other node c_k^+ with $1 \leq k \leq m$ and $k \neq i$. Lemma 5.2 implies that the corresponding value of variable u_j has been set; *i.e.* $u_j = 1$ if the flow passes node u_j^1, or $\overline{u}_j = 1$ if it passes through node $\overline{u}_j = 1$. This leads to a contradiction. □

The second part of the proof of Lemma 5.3 together with Lemma 5.2 give the idea of the transformation from *3SAT*. First, we have to send one unit of flow from each of the nodes d_j^+ to d_j^-. This ensures that (at least) one of the arcs between the copies of the literals has to be reversed. The arc which has not been switched can then be used for the flow of the nodes c_j^+, allowing the clauses to have a truth assignment. Hence, the value of the literals is reflected by the switching of the arcs.

Theorem 5.6. *DTCF is* NP*-complete in the strong sense.*

Proof. DTCF $\in$ NP, as a non-deterministic algorithm needs only guess the set of arcs to be reversed together with a flow f and check if the flow is feasible with time bound $T = 5$; which can all be done in polynomial time. Lemma 5.3 states that the given transformation G_{3SAT} from *3SAT* to DTCF is valid. As the cardinality of the node set and the arc set of the constructed graph is $O(|C|)$, the transformation is polynomial in the input size of *3SAT*. □

We want to mention that the transformation from *3SAT* can easily be changed to the general SAT by only changing the appropriate arcs from the clause nodes to the nodes representing the literals.

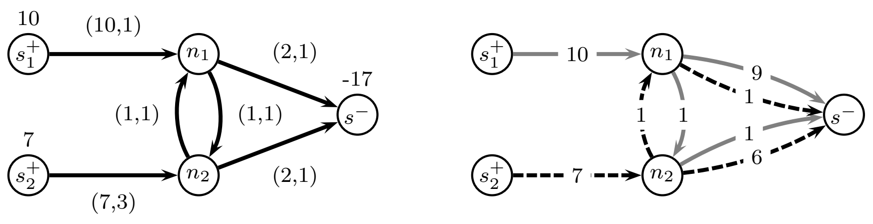
5.4.2.2 What Makes DTCF so Tough to Solve?

Ford and Fulkerson introduced the idea of temporally repeated chain flows of a static flow. This enabled them to solve the maximum dynamic flow problem with one source and one sink. The fundamental principle is that there is always an optimal dynamic flow which uses only one direction of an arc, but never both. They call this a *standard chain decomposition*. This property allows us to solve the maximum dynamic flow problem in strongly polynomial time. We exploit this property in Section 5.4.1 to solve the MDCF problem.

The concept of standard chain decomposition is not sufficient for some well known dynamic flow problems [117, 105, 156]. An example is given in Fig. 5-2. Graph $G = (V, A)$ shown in Fig. 5-2 (a) has a feasible flow with time $T = 6$ as illustrated in Fig. 5-2 (b). The dashed and gray lines show the two flows from nodes s_1^+ and s_2^+ to node s^-, respectively. Analyzing the graph reveals that there is no feasible flow within time horizon $T = 6$ using only one of the arcs (n_1, n_2) or (n_2, n_1); this can be seen, for instance, by considering the flow trough the cut separating s^- from the rest of the graph.

However, it was still possible to solve the maximum dynamic flow problem with multiple sources and multiple sinks in the time-expanded graph; resulting in a pseudo-polynomial running time algorithm. However, Hoppe was able to provide a polynomial time algorithm for the dynamic transshipment problem [117]. He introduced the concept of *non-standard chain decomposition*, allowing flow in either directions of an arc at different time steps – if both directions of an arc are present in the graph.

Loosely speaking, the procedures P-MCF and P-MDCF reverse the arcs on the fly and they are blind whether they reverse an arc or not. This does not cause any problems in the context of static flows or single source and single sink dynamic flows, as in a standard chain decomposition, one can always derive an optimal solution using only one the arcs during the whole time horizon. However, in the case of multiple sources and multiple sinks, the potential of using both arcs leads to the problem that we have to know if an arc has been reversed or not. But exactly this memory and the tradeoff of reversing the arc now or at a later time, makes the problem NP-complete. Consider Fig. 5-2 again. Applying the idea of procedures P-MDCF to this problem leads to the following result: At time 1, we would switch arc (n_2, n_1) in order to increase the capacity and at time point 3, we would switch it back again; resulting in a flow needing only $T = 5$ time steps.



A Graph $G = (V, A)$

B Feasible flow with $T = 6$ using both arcs (n_1, n_2) and (n_2, n_1)

Figure 5-2. A tough instance of DTCF

In Sec. 5.4.2.1, we showed that DTCF is NP-complete. The reduction from *3SAT* involves $|C| + |U|$ source nodes and $|U| + 1$ sink nodes. In the following, we show that there is no polynomial time algorithm for the DTCF problem having only two sources and one sink (or one source and two sinks), unless $\mathsf{P} = \mathsf{NP}$. In other words, allowing only one more source or sink to DTCF makes the problem NP-complete.

We do not go into full detail here, but rather provide the idea of a reduction from PARTITION, which is motivated by the key observation of Lemma 5.2 and the NP-completeness proof by MELKONIAN, [139]. Given is a finite set A and a size $a_i \in \mathbb{Z}^+$ for each $i \in A$. The PARTITION problem decides whether there is a subset $\bar{A} \subseteq A$ such that $\sum_{i \in \bar{A}} a_i = \sum_{j \in \bar{A} \setminus A} a_j$, or not. PARTITION is known to be NP-complete (in the weak sense), see [82, Theorem 3.5, Chapter 4.2]. Let $\sum_{i \in A} a_i = 2L$ with $L \in \mathbb{Z}^+$. We construct an instance of the DTCF with two source nodes s_1^+, s_2^+ and one sink node s^-, as shown in Fig. 5-3. The idea of this transformation is that the flow at node s_2^+ has to pass through node v_0^1 to reach node s^-, and one unit of flow from node s_1^+ has to travel though node v_n^1 to node s^-. This is indeed true as otherwise the total time bound of $T = 2L + 2$ would be exceeded. The flow through the nodes v_0^1 to v_n^1 and back gives the assignment to set $\bar{A}$; *i.e.* $i \in \bar{A}$ if and only if arc $(v_{i_1}^1, v_i^1)$ is not reversed in the graph.

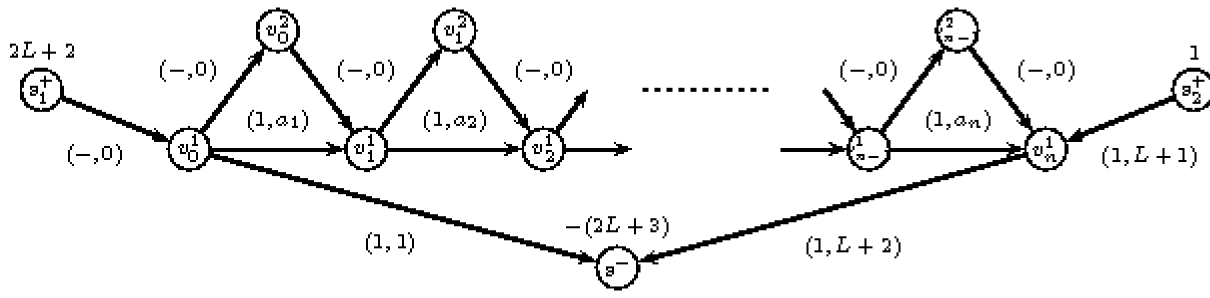


Figure 5-3. Instance for DTCF with time bound $T = 2L + 2$ resulting from PARTITION

5.5 Contraflow Problems with Arc Switching Cost

To allow the switching of an arc in order to increase the capacity in one direction results from the application in evacuation scenarios. However, in practice, you might not be able to switch certain arcs. For instance, in evacuation scenarios, certain streets are reserved for emergency vehicles but can also be used by (limited number of) other travelers; *i.e.* this can be modeled by reducing the capacity of this arc and blocking it from being reversed. In addition, the switching of an arc is highly costly; *i.e.* in order to switch the direction of a highway, we have to set up police blocks on each entry to the highway. Hence, it is natural to ask what are the minimum cost incurred in switching the arcs allowing a certain (minimum) amount of flow. This leads to the following problem.

Definition 5.7 (Fixed Switching Cost Contraflow (FSCF))**.**

Instance*: A directed graph $G = (V, A)$ with a set of sources S^+, a set of sinks S^-, excess $b \in \mathbb{Z}^{|V|}$, arc capacities c_e and arc-switching cost b_e^f for each arc $e \in A$.*

Question*: Find a feasible flow f in G with minimal total cost, if the direction of the arcs can be reversed with (fixed) cost b^f.*

Note that FSCF is a static problem with multiple sources and multiple sinks. The fixed cost b_e^f occur, whenever arc e is reversed. This definition allows to model the situation described above: Whenever an arc cannot be reversed, then its cost can be assigned a high value; *i.e.* Big M. As the cost of switching can differ for each arc, we can distinguish between the effort of reversing an arc; *i.e.* reversing a highway or an alleyway involves different cost or resources.

The fixed switching-cost contraflow problem has the following interesting value. One can solve the MTCF problem and determine the optimal flow in the graph, see Corollary 5.1. Later, one can apply the FSCF problem to determine the minimal cost implied by the switching of arcs, while still pushing the optimal amount of flow trough the graph.

Notice that the FSCF problem has a similar structure as the *minimum concave-cost network flow* problems. These problems ask to find a feasible flow while minimizing the total cost which are in this case the sum of concave-costs induced by using of the arcs. For an exact definition and an overview about this problem, please see the survey by Guisewite and Pardalos, [102]. We can basically assume the concave-cost per arc to consist of fixed cost, occurring whenever this particular arc is used, and a variable cost, depending on how much flow is send trough this arc, see [120]. Fixing the variable cost to zero leads to a special problem called *minimum cost fixed flow* (MCFF) problem. Krukme et al. prove that this problem is NP-hard in the strong sense even on series-parallel graphs, [125, Theorem 14]. Series-parallel graphs have a very special structure and are defined recursively, see [97, 24]. Furthermore, Krukme et al. show that the minimum cost fixed flow problem is equivalent to the following problem, [125, Theorem 8]:

Definition 5.8 (0/1-Minimum Improvement Flow (MIF))**.**

Instance: *A graph $G = (V, A)$ with sink node s^+, source node s^-, excess $b \in \mathbb{Z}^{|V|}$, arc capacities $c_e \in \mathbb{Z}^+$, maximum capacities $C_e \in \mathbb{Z}^+, C_e \geq c_e$ and capacity improvement cost $\bar{b}_e \in \mathbb{Z}$.*

Question*: Determine an improvement strategy $d : A \to \{0, C_e - c_e\}$ with minimum cost $\sum_{e \in A} d_e \bar{b}_e$, such that the graph with the improved capacity $u_e + d_e, \forall e \in A$, allows a feasible flow f from s^+ to s^-.*

The definition given here is slightly different then the one in the paper by Krukme et al., [125, Definition 7]. Basically, we assume all data to be positive integral. The improvement strategy function d is a 0-1 decision if additional capacity is used or not; independent of how much additional capacity is used. The cost for this additional capacity for arc e is fix at value $(C_e - c_e)\bar{b}_e$. In order to prove that FSCF is strongly NP-hard, we show that it is equivalent to MIF.

Theorem 5.7. *Fixed switching-cost contraflow is equivalent to 0/1-minimum improvement flow.*

Proof. Without loss of generality, we can assume the FSCF problem to have single source and single sink. Recognize that the graph transformation provided for Corollary 5.1 works here.

"$\Rightarrow$" Given an instance of FSCF for graph $G = (V, A)$ with arc capacity c_e and arc-switching cost b_e^f. Construct an instance of MIF for graph $\mathbb{G} = (\mathbb{V}, \mathbb{A})$ as follows. If there is an arc $(i, j) \in A$ and $(j, i) \notin A$, then $(i, j), (j, i) \in \mathbb{A}$ with $_{i,j} = \mathbb{C}_{i,j} = \mathbb{C}_{j,i} := c_{i,j}$, $\bar{}_{i,j} = {}_{j,i} := 0$, and $\bar{}_{j,i} := b^f_{i,j}/c_{i,j}$ respectively. For the case that $(i, j), (j, i) \in A$, we define $(i, j), (j, i) \in \mathbb{A}$ with $_{i,j} := c_{i,j}$, $\mathbb{C}_{i,j} = \mathbb{C}_{j,i} := c_{i,j} + c_{j,i}$, $\bar{}_{i,j} := b^f_{j,i}/(c_{i,j} + c_{j,i})$, $_{j,i} := c_{j,i}$, and $\bar{}_{j,i} := b^f_{i,j}/(c_{i,j} + c_{j,i})$ respectively. By applying the cycle reduction principle used in Sec. 5.3 and 5.4.1, we can see that this transformation is indeed valid.

"$\Leftarrow$" Given an instance of MIF for graph $G = (V, A)$ with c_e, C_e and $\bar{b}_e$, construct an instance of FSCF for graph $\mathbb{G} = (\mathbb{V}, \mathbb{A})$ as follows. For any arc $(i, j) \in A$, we have the three arcs $(i, j), (i, \bar{i}), (j, \bar{i}) \in \mathbb{A}$. Define $_{i,j} := c_{i,j}$, $^f_{i,j} = {}^f_{i,\bar{i}} := M$, $_{i,\bar{i}} = {}_{j,\bar{i}} = C_{i,j} - c_{i,j}$ and $^f_{j,\bar{i}} := \bar{b}_{i,j}(C_{i,j} - c_{i,j})$, where M is a big number preventing to switch the corresponding arc in an optimal solution. □

Recognize that having fixed cost for arc reversals makes the problem NP-hard, even in the static case. One reason is, for instance, the previously mentioned observation, that the procedure P-MCF is 'blind' for the arc reversal decisions. Adding a time component to FSCF makes it practically even more difficult to solve. The time component reveals also the differences between the (dynamic) fixed switching-cost contraflow problem and the (dynamic) 0/1-minimum improvement flow problem: MIF affects only a particular arc (i, j), while in FSCF also the reverse arc (j, i) is affected, if both arcs are contained in the graph.

5.6 Conclusions

This chapter formally introduces the contraflow problem that has applications in emergency transportation management. Several classic network flow problems are studied, including static and dynamic networks. A polynomial time algorithm for the dynamic contraflow problem with single source and single sink is given, together with an NP-completeness proof for the dynamic transhipment contraflow problem. The hardness of the contraflow problem with arc reversal cost is also indicated.

CHAPTER 6
MULTIMODAL SOLUTIONS FOR EVACUATION PROBLEMS

6.1 Introduction

The survey on evacuation problems [16] indicates that there is a shortage of analytical techniques in multimodal evacuation studies. This chapter focuses on establishing efficient evacuation routes with bimodal transportation. We consider emergency management or event management situations such as football game, which assumes the absence of panic situations but still captures the several aspects of an evacuation settings. These include high demands during the event, need to satisfy demands quickly and congestion due to high demands. We assume private cars and buses as the modes of transportation. The cars are to take a path from source to destination, while the buses are routed. We assume that the routes of the buses are known. We need to establish efficient paths for the cars and determine the frequency of the buses along the routes. The problem is comparable to the line planning problem [96, 2, 28], where multiple lines or modes of transportation are available and demands of people are available at specific time windows. The lines are predefined paths and the frequency of a line needs to be determined. Columns generation procedures are quite popular for the line planning problem and much research have been done in this area [28, 165]. We employ the branch and price approach to solve the problem.

In the next section, we formally introduce the problem and discuss the integer linear programming formulation. In section (6.3) we discuss the multimodal flow problem and integer programming formulation. In section (6.4), we discuss the branch and price procedure for the multimodal flow problem. Finally in section (6.5), we provide some numerical results.

6.2 Multimodal Problem

Multimodal flow problems are known to be NP-hard [169]. Thus, the problem, tailored to the needs of evacuation studies, requires efficient approaches to solve them either approximately or exactly. We provided a path based formulation that would enable us to employ a branch and price procedure to solve the problem. We formally define the problem and state the assumptions.

6.2.1 Problem Definition

In this problem, we have two sets of people depending on their modes of transport. We recognize the modes of transportation as private cars and buses. We know the demands of cars and people traveling by bus for every pair of node. We also have a set of bus tours that has already been established. The arcs of the network under consideration is shared by both cars and buses. Each link have a capacity and a cost. We need to determine the most efficient path for the cars between the origins and destination and the optimal buses routes required without exceeding the capacity of the arcs. Networks with travel times on their links could be expanded over time with each link having a cost depending on the time instance of the originating node and its distance from its nearest source node. The optimal bus routes then would provide us with the frequency with which the buses have to be routed.

We formally define the problem as: Given a graph $G(V, E)$ with c_{ij}, u_{ij} as the cost and capacity on each arc $ij \in E$, a set of T bus tours, and two sets of origin destination pairs $(OD)_1$ and $(OD)_2$ corresponding to demands of origin and destinations of cars and people traveling by bus respectively, determine the minimum cost path of the cars and the optimal subset of bus routes satisfying demands and capacity.

6.2.2 Assumptions and realization

We make some simplistic assumptions, which does not hinder the realization of the model. We assume that the demands between the origin-destination pairs are known and remain static. We assume that the bus routes are established and we only need to determine their frequency. We also assume zero loading and unloading time for the buses. The last assumption could however be overcome in the current model by appropriately changing the bus routes to accommodate fixed loading times. We now provide a path and route based formulation that will enable us to implement a branch and price mechanism.

6.3 Formulation and discussion

A bimodal evacuation problem is considered in which two modes of evacuation, namely private cars and buses, are used for evacuating people from the origin nodes. The people have their destination preferences from the respective origins. In an evacuation setting, the buses are routed to pick up people from their origins and drop them at their destinations. The private cars are taken by people directly from the origin to the respective destinations. The demands are known in terms of number of cars and number of people for the respective origin-destination pair. The buses and the cars have to share the capacity of the arcs. We aim to reach the destinations with the least possible cost. The objective function is a little loosely defined, but we will elaborate it shortly. We provide a branch and price framework to solve the problem. We have two subproblems, one to generate the paths of the private cars and the other to generate paths of people. For the time expanded formulation, we need to determine an upper bound on the value of T. A loose bound on this value could be obtained by individually bounding the times for buses and cars separately and adding them together. The implication is we serially route them one after the other and this is still a feasible solution to the problem. This will also help us obtain an initial feasible solution to our branch and price procedure. We discuss the procedure to obtain individual time bounds later.

f_p (x_p) is a binary variable indicating whether path p is used to satify the demand of the corresponding origin destination pair. b_t is binary variable with value 1 if a bus tour t is used and 0 otherwise. $\alpha_1(ij,p)(\alpha_2(ij,p))$ is an indicator variable with value 1 if arc ij is in bus path(car path) p and 0 otherwise. $\beta(ij,t)$ is an indicator variable with value 1 if arc ij is in tour t and 0 otherwise. $\gamma_1(st,p)(\gamma_2(st,p))$ is an indicator variable with value 1 if a bus path(car path) p has origin and destination as s and t respectively and 0 otherwise. Let P_1 and P_2 be the sets of all bus paths and car paths respectively and T be the set of all bus tours. B is the capacity of a bus. OD_1 and OD_2 are the sets of origins-destination pairs corresponding to buses and cars respectively. d^1_{st} and d^2_{st} is the demand of people using buses and cars respectively from origin s to destination t. Finally, b_t is a binary variable with value 1 if a tour is picked and 0 otherwise.

$$\text{Minimize} \quad \sum_{p \in P_1} c_p x_p + \sum_{p \in P_1} c_p f_p \tag{6–1}$$

s.t.

$$\sum_{p \in P_1} \gamma_1(st, p) x_p = 1, \forall st \in OD_1 \tag{6–2}$$

$$\sum_{p \in P_2} \gamma_2(st, p) f_p = 1, \forall st \in OD_2 \tag{6–3}$$

$$\sum_{p \in P_1} d^1_{st} \alpha_1(ij, p) x_p - \sum_{t \in \mathrm{T}} \beta(ij, t) B b_t \leq 0, \forall ij \in E \tag{6–4}$$

$$\sum_{p \in P_2} d^2_{st} \alpha_2(ij, p) f_p + \sum_{t \in \mathrm{T}} \beta(ij, t) b_t \leq u_{ij}, \forall ij \in E \tag{6–5}$$

$$x_p \in \{0, 1\}, \forall p \in P_1 \tag{6–6}$$

$$f_p \in \{0, 1\}, \forall p \in P_2 \tag{6–7}$$

$$b_t \in \{0, 1\}, \forall t \in \mathrm{T} \tag{6–8}$$

6.4 Branch and Price Mechanism

We note that the problem has exponentially many paths in terms of the input size of the graph and we will be generating these variables in the subproblem. This is a standard approach in most of the routing and scheduling problems.

6.4.1 Restricted Master Problem (RMP)

The restricted master problem is obtained by relaxing constraints (6–6) - (6–7) as continuous variables and replacing the sets P_1 and P_2 by the restricted sets $\bar{P}_1 \subseteq P_1$ and $\bar{P}_2 \subseteq P_2$ respectively. This leads to the restricted master problem.

$$\text{Minimize} \quad \sum_{p \in \bar{P}_1} c_p x_p + \sum_{p \in \bar{P}_1} c_p f_p \tag{6–9}$$

s.t.

$$\sum_{p \in \bar{P}_1} \gamma_1(st, p) x_p = 1, \forall st \in OD_1 \tag{6–10}$$

$$\sum_{p \in \bar{P}_2} \gamma_2(st, p) f_p = 1, \forall st \in OD_2 \tag{6–11}$$

$$\sum_{p \in \bar{P}_1} d^1_{st} \alpha_1(ij, p) x_p - \sum_{t \in \mathrm{T}} \beta(ij, t) B b_t \leq 0, \forall ij \in E \tag{6–12}$$

$$\sum_{p \in \bar{P}_2} d^2_{st} \alpha_2(ij, p) f_p + \sum_{t \in \mathrm{T}} \beta(ij, t) b_t \leq u_{ij}, \forall ij \in E \tag{6–13}$$

$$0 \leq x_p \leq 1, \forall p \in \bar{P}_1 \tag{6–14}$$

$$0 \leq f_p \leq 1, \forall p \in \bar{P}_2 \tag{6–15}$$

$$0 \leq b_t \leq 1, \forall t \in \mathrm{T} \tag{6–16}$$

Let $\pi \in \mathbb{R}^{OD_1}$ be the unrestricted dual variable corresponding to constraint set (6–10), $\mu \in \mathbb{R}^{OD_2}$ be the unrestricted dual variable corresponding to constraint set (6–11), $\eta \in \mathbb{R}^E_-$ be the non-positive dual variable corresponding to the constraint set (6–12) and finally $\psi \in \mathbb{R}^E_-$ be the non-positive dual variable corresponding to the constraint set (6–13). In the pricing problem, we are interested in the reduced cost of the variable x_p and f_p. We determine the minimum reduced cost of the path flow variables in the pricing subproblems. If the minimum reduced cost corresponding to a origin destination pair is negative we add it to the restricted set (corresponding to the cars or buses) and the restricted master problem is solved again.

6.4.2 People-Path Subproblem

In the people-path subproblem, we determine the minimum negative reduced cost, $\bar{x}_p$, of a path flow variable, x_p, for people taking buses between a given origin-destination pair $st \in OD_1$. This is given by

$$\bar{x}_p = c_p - (\pi_{st} + \sum_{\forall(ij)\in E} \alpha_1(ij,p)\eta_{ij}) = -\pi_{st} + \sum_{\forall(ij)\in E} (\alpha_1(ij,p)c_{ij} - \alpha_1(ij,p)\eta_{ij}) \tag{6–17}$$

The cost of the path is given by the sum of the cost on arcs in the above equation. Now, the shortest path problem for all pairs of $st \in OD_1$ with the above arc costs $c_{ij} - \eta_{ij}$ is solved. If π_{st} is more than the length of a path, then it is added to the restricted path set P_1 and the RMP is solved again. We observed that the dual variable μ is negative and hence the cost on each arc is positive. Thus the resulting shortest path problem is solvable in polynomial time.

6.4.3 Car-Path Subproblem

In the car-path subproblem, we are concerned with the negative reduced cost, $\bar{f}_p$, of car flow variables, f_p, for the pairs of origin-destination $st \in OD_2$. This given by

$$\bar{f}_p = c_p - (\mu_{st} + \sum_{\forall(ij)\in E} \alpha_2(ij,p)\psi_{ij}) = -\mu_{st} + \sum_{\forall(ij)\in E} (\alpha_2(ij,p)c_{ij} - \alpha_2(ij,p)\psi_{ij}) \tag{6–18}$$

We solve the shortest path, just as in people-path subproblem, but with arc cost $c_{ij} - \psi_{ij}$ and if the cost of a path from s to t is less than μ_{st}, we add it to the restricted set P_2 and the RMP is solved again.

6.4.4 Branching Strategy

The branching rules is important as this determines the complexity of the pricing problem. The branching also induces some practical difficulties that needs to be explicitly handled. We address the two important problems encountered while branching.

The decision to branch occurs at a node of the branch and bound tree, when we cannot enter any more columns to the restricted sets from either subproblems and the relaxed LP solution at the current node is infeasible to the integer program. At this juncture, if any of the b_t variables are fractional, we decide to branch on them. It is easy to see that this branching will not cause any difficulty to the subproblems as the arc costs corresponding to the shortest path problems still remains positive. If none of the b_t variables are fractional and if a flow variable is fractional, a branching on the fractional path flow variable would restrict the subproblems to generate paths other than the path that was fractional. For instance, let f_p be the fractional path with value $\bar{f}_p$. We branch by adding

$$f_p \leq \lfloor \bar{f}_p \rfloor$$

to one branch and

$$f_p \geq \lceil \bar{f}_p \rceil$$

to the other branch. The difficulty now in the subproblem is that a candidate path generated for that origin-destination pair should not be the branched variable. We cannot guarantee that the shortest path problem could generate such a path. In fact after k branchings, we might have to solve the k^{th} shortest path subproblem. This is a common difficulty that arises in branch and price approaches for multicommodity flow problems. There are a few techniques in the literature that handles this issue [41, 9, 162, 176]. One technique is to make an arc or a set of arcs of the path that was branched as forbidden arcs in the branches. Thus the subproblem will not regenerate the path [41]. Another technique in the literature [9] to solve the problem is to branch on arc flow variable instead of path flow variable. For instance, the amount of cars on an arc ij is given by $\sum_{p \in P_2} d^2_{st} \alpha_2(ij,p) f_p$ and let $\bar{x}_{ij}$ be the fractional flow on the arc. So we can add the constraint

$$\sum_{p \in P_2} d^2_{st} \alpha_2(ij,p) f_p \leq \lfloor \bar{x}_{ij} \rfloor$$

to one branch and

$$\sum_{p \in P_2} d^2_{st} \alpha_2(ij,p) f_p \geq \lceil \bar{x}_{ij} \rceil$$

to other branch. This however does not guarantee positive arc costs anymore in the subproblems and they become NP-hard. This problem was overcome by adding separate variables for flows along cycles in the RMP. Thus the subproblem has to return a shortest path if available or a negative cost cycle. This could be solved in polynomial time. In this problem, we revisit the technique employed by [162] for a bandwidth packing problem, which is very similar to the multicommodity flow problem. A fractional path variable is dealt by creating a number of branches. Each branch corresponds to an arc of the fractional path, where it is forbidden and there is one additional branching node in which the path is fixed to the solution.

The next problem to be addressed occurs when we arrive at a branch and bound node with the LP relaxation resulting in an infeasible solution. In an elementary branch and bound procedure we prune the search in this situation. However, this is not possible in branch and price mechanism as we have not yet considered the entire set of columns and hence there might exist a path that has not been entered but could provide a feasible solution in the future. We take care of this issue, by adding dummy paths in the initial solution with high cost that will provide us with the feasible solution.

6.5 Computational Results

We tested grid graphs of size 25 to 400 nodes. Table 6-1 enumerates the instances we tested. The largest instance tested was grid graph with 400 nodes and 1520 edges with 100 cars and 40 buses.

Table 6-1. Results of the Branch & Price model tested on grid graphs

#	Nodes	Arcs	Cars	Buses	Tours	CPU Time
1	25	80	6	6	3	0.013
2	25	80	10	10	4	0.064
3	25	80	20	10	4	0.174
4	25	80	15	15	4	0.089
5	100	360	5	5	4	0.138
6	100	360	10	10	5	1.161
7	100	360	20	10	5	1.063
8	100	360	20	20	7	0.8
9	225	840	10	10	7	5.637
10	225	840	20	20	10	1.205
11	400	1520	100	40	14	52.503

Additionally, in order to test the roubustness of the code we tested a few online benchmark instances for multicommodity flow problems [128] for four planar graphs and results are provided in table 6-2. We generated one dummy bus tour and one bus commodity for each of the instances.

Table 6-2. Results of the Branch & Price model tested on planar graphs with one bus commodity

#	Nodes	Arcs	Cars	CPU Time
1	30	150	92	0.08
2	50	250	267	0.177
3	80	440	543	0.509
4	100	532	1085	1.605

6.6 Conclusion

This chapter provides a branch and price framework to solve a bimodal multicommodity flow problem. We consider cars and buses as two modes of transportation and we obtained optimal paths for cars and identified the bus routes needed for transportation. We tested the model on grid graphs of sizes upto 400 nodes. As a future work, we need to employ heuristic methods for the subproblems, develop procedures that would provide good lower bound for the branch and bound. We are currently in the process of implementing a box stabilization technique in order to accelerate the convergence of the branch and price procedure that would enable us to test much larger instances.

CHAPTER 7
CONCLUSIONS

In this dissertation, we focused on network models having applications in disaster management. We examined three kinds of applications, in which we studied different problems. The first application dealt with the identification of critical nodes in a graph that are crucial for its connectivity. We analysed and provided solution techniques to two variations of the problem. We then studied two path planning problems that involves in routing of agents or vehicles in a network in order to visit targets along their routes. We finally studied evacuation problems in order to establish evacuation routes and contraflow plans. In all these problems, we examined the complexity of the problem, provided integer programming formulations and heuristic or exact solution techniques to solve them.

In chapter 2, we studied the variations of the CRITICAL NODE DETECTION PROBLEM. The problem identifies the nodes whose deletion results in a subgraph that has maximum fragmentation. We provided complexity analysis for the problems, integer programming formulation and heuristic solutions. We would like to study a weighted version of the problem and explore approximation algorithms that would guarantee a theoretical bound on the solution.

In chapter 3, we studied two path planning problems. In the first problem, theTARGET VISITATION PROBLEM, we formulated it as an integer program and provided a genetic algorithm for the problem. We then compared the solutions of the heuristic with the CPLEX solutions. We also studied COMMUNICATION MODELS FOR COOPERATIVE NETWORK, where we have to route multiple agents to visit targets in a cooperative network in order to maximize their communication. We provided an integer programming formulation and a GRASP based heuristic procedure to solve the problem. As a future work, we would like to parallelize the heuristic and enhance it with more sophisticated search procedures.

In chapter 5, we studied various network flow problem with arc reversal capabilities. We performed a detailed complexity analysis for these problems. Future research in these problems involves in developing efficient solution techniques for the NP-hard problems. In chapter 6, we studied the evacuation problem for which we provided a branch and price approach to solve a bimodal multicommodity flow problem. We are currently working on stabilization techniques to accelerate the convergence of the algorithm in order to test large scale instances.

REFERENCES

[1] E. Aarts and J.K. Lenstra, editors. *Local Search in Combinatorial Optimization*. Wiley, 1997.

[2] A. Abbas-Turki, R. Bouyekhf, O. Grunder, and A. El Moudni. On the line planning problems of the hub public-transportation networks. *International Journal of Systems Science*, 35(12):693–706, 2004.

[3] M. Van Aerde, H. Rakha, and H. Paramahamsan. Estimation of o-d matrices: The relationship between practical and theoretical considerations. In *In the proceedings of 82nd TRB Annual Meeting*, pages 122–130, 2003.

[4] R. K. Ahuja, T. L. Magnati, and J. B Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, Englewood Cliffs, New Jersey, 1993.

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

[6] R.M. Aiex, M.G.C. Resende, and C.C. Ribeiro. Probability distribution of soltuion time in GRASP: An experimental investigation. *Journal of Heuristics*, 8:343–373, 2002.

[7] R.M. Aiex, M.G.C. Resende, and C.C. Ribeiro. TTTPLOTS: A perl program to create time-to-target plots. *Optimization Letters*, published online(DOI: 10.1007/s11590-006-0031-4), 2006.

[8] S. Algers, E. Bernauer, M. Boero, L. Breheret, C. Di Taranto, M. Dougherty, and K. Fox. Review of micro-simulation models. Technical report, SMARTEST Project Deliverable D3, Institute for Transport Studies, University of Leeds, 1997.

[9] F. Alvelos. *Branch-and-Price and Multicommodity Flows*. PhD thesis, Universidade do Minho, 2005. http://repositorium.sdum.uminho.pt/dspace/bitstream/1822/2736/1/falvelos_phd.pdf.

[10] C. Antoniou, M. Ben-Akiva, M. Bierlaire, and R. Mishalani. Demand simulation for dynamic traffic assignment. In *Presented at the 8th IFAC Symposium on Transportation Systems*, pages 114–118, 1997.

[11] A. Arulselvan, C.W. Commander, L. Elefteriadou, and P.M. Pardalos. Detecting critical nodes in social networks. *Social Networks*, submitted, 2007.

[12] A. Arulselvan, C.W. Commander, M.J. Hirsch, and P.M. Pardalos. *Communication models for a cooperative network of autonomous agents*, page in press. Springer, 2007.

[13] A. Arulselvan, C.W. Commander, and P.M. Pardalos. A hybrid genetic algorithm for the target visitation problem. *Computers and Operations Research*, submitted, 2007.

[14] A. Arulselvan, C.W. Commander, and P.M. Pardalos. A random keys based genetic algorithm for the target visitation problem. *Lecture notes in control and information sciences*, 369:389–397, 2007.

[15] A. Arulselvan, C.W. Commander, P.M. Pardalos, and O. Shylo. Managing network risk via critical node identification. In N. Gulpinar and B. Rustem, editors, *Risk Management in Telecommunication Networks*, page submitted. Springer, 2007.

[16] A. Arulselvan, L. Elefteriadou, and P.M. Pardalos. A survey on evacuation problems. Technical report, Center for Multimodal Transportation and Congestion MItigation, University of Florida, 2004.

[17] K. Ashok and M. Ben-Akiva. Dynamic o-d matrix estimation and prediction for real-time traffic management systems. In C.F. Daganzo, editor, *International Symposium on Transportation and Traffic Theory*, volume 2337 of *Transporation and Traffic Theory*, pages 36–53. Elsevier Science Publishing Company Inc., 1993.

[18] D.L. Bakuli and J.M. Smith. Resource allocation in state-dependent emergency evacuation networks. *European Journal of Operational Research*, 89(3):543–555, 1996.

[19] J.X. Ban, H.X. Liu, and B. Ran. A link based quasi-variational inequality model for dynamic user equilibria, towards real time traffic operations. In *Proceedings of the 8th International IEEE Conference on Intelligent Transportation Systems*, pages 458– 463, September 2005.

[20] F. Barahona, M. Grötschel, M. Jünger, and G. Reinelt. An application of combinatorial optimization to statisical physics and circuit layout design. *Operations Research*, 36:493–513, 1998.

[21] R. Battiti and G. Tecchiolli. Parallel biased search for combinatorial optimization: Genetic algorithms and TABU. *Microprocessors and Microsystems*, 16:351–367, 1992.

[22] A. Bavelas. A mathematical model for group structure. *Human Organizations*, 7:16–30, 1948.

[23] J.C. Bean. Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal on Computing*, 6(2):154–160, 1994.

[24] M. W. Bern, E. L. Lawler, and A. L. Wong. Linear-time computation of optimal subgraph of decomposable graphs. *Journal on Algorithms*, 8:216–235, 1987.

[25] M.C.J. Bliemer and P.H.L. Bovy. Quasi-variational inequality formulation of the multiclass dynamic traffic assignment problem. *Transportation Research B*, 37:501–519, 2003.

[26] J.J. Blum and A. Eskandarian. The impact of multi-modal transportation on the evacuation efficiency of building complexes. In *Proceedings of the 7th International IEEE Conference on Intelligent Transportation Systems*, pages 702–707, Oct 2004.

[27] S.P. Borgatti. Identifying sets of key players in a network. *Computational, Mathematical and Organizational Theory*, 12(1):21–34, 2006.

[28] R. Borndörfer, M. Grötschel, and M.E. Pfetsch. A column-generation approach to line planning in public transport. *Transportation Science*, 41(1):123–132, 2007.

[29] W.M. Bretherton and M. Elhaj. Is a reversible lane system safe. In *Compendium of technical papers for 66th ITE Annual Meeting*, pages 277–281, 1996.

[30] Luce Brotcorne, Daniel De Wolf, Michel Gendreau, and Martine Labb. A dynamic user equilibrium model for traffic assignment in urban areas. In M. Gendreau and P. Marcotte, editors, *Transportation and Network Analysis: Current Trends*, pages 49–70. Kluwer Academic Publishers, 2002.

[31] J.L. Bryan. Human behaviour in fire: the development and maturity of a scholarly study area. *Fire and Materials*, 23(6):249–253, Mar 2000.

[32] L.S. Buriol, M.G.C. Resende, C.C. Ribeiro, and M. Thorup. A hybrid genetic algorithm for the weight setting problem in OSPF/IS-IS routing. *Networks*, 46:36–56, 2005.

[33] R. Burkard, K. Dlaska, and B. Klinz. The quickest flow problem. *Mathematical Methods of Operations Research*, 37(1):31–58, 1993.

[34] R.E. Burkard, K. Dlaska, and B. Klinz. The quickest flow problem. *Mathematical Methods of Operations Research*, 37(1):31–58, February 1993.

[35] S.I. Butenko. *Maximum Independent Set and Related Problems, with Applications.* PhD thesis, University of Florida, 2003.

[36] S.I. Butenko, X. Cheng, C.A.S. Oliveira, and P.M. Pardalos. A new algorithm for connected dominating sets in ad hoc networks. In S. Butenko, R. Murphey, and P. Pardalos, editors, *Recent Developments in Cooperative Control and Optimization*, pages 61–73. Kluwer Academic Publishers, 2003.

[37] S.I. Butenko, R.A. Murphey, and P.M. Pardalos, editors. *Cooperative Control: Models, Applications, and Algorithms*. Springer, 2003.

[38] S.I. Butenko, R.A. Murphey, and P.M. Pardalos, editors. *Recent Developments in Cooperative Control and Optimization*. Springer, 2004.

[39] M. Carey. A constraint qualification for a dynamic traffic assignment. *Transportation Science*, 20(1):55–58, 1986.

[40] M. Carey. Optimal time-varying flows on congested networks. *Operations Research*, 14(1):55–58, 1986.

[41] C.Barnhart, C.A. Hane, and P.H. Vance. Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems. *Oper. Res.*, 48(2):318–326, 2000.

[42] L.G. Chalmet, R.L. Francis, and P.B. Saunders. Network models for building evacuation. *Management Science*, 28:86–105, 1982.

[43] S. Chanas and P. Kobylanski. A new heuristic algorithm solving the linear ordering problem. *Computational Optimization and Applications*, 6:191–206, 1996.

[44] E. Chang and A. Ziliaskopoulos. Formulation of analytical time varying intermodal person trip assignment model. *Transportation Research Record*, 1882:1–9, 2004.

[45] A. Chen, S. Kongsomsaksakul, and Z. Zhou. Assessing network vulnerability using combined travel demand model. In *86th Transportation Research Board Annual Meeting, Washington, D.C.*, pages 07–2667, 2007.

[46] X. Chen. Agent based simulation of evacuation strategies under different road network structures. Technical report, University Consortium of Geographic Information Science, September 2006. http://www.ucgis.org/summer03/studentpapers/xuweichen.pdf.

[47] X. Chen and B. Zhan. Agent-based modeling and simulation of urban evacuation relative effectiveness of simultaneous and staged evacuation strategies. *Journal of Operations Research Society*, 59(1):25–33, 2006.

[48] J. Cheriyan and S. N. Maheshwari. Analysis of preflow push algorithm for maximum network flow. *SIAM Journal on Computing*, 18:1057–1086, 1989.

[49] B.H. Chiarini, W. Chaovalitwongse, and P.M. Pardalos. A new algorithm for the triangulation of input-output tables in eEconomics. In P. Pardalos, A. Migdalas, and G. Baourakis, editors, *Supply Chain and Finance*, pages 253–272. World Scientific, 2004.

[50] Y. Chiu, J. Villalobos, B. Gautam, and H. Zheng. Modeling no-notice mass evacuation using a dynamic traffic flow optimization model. *IIE Transactions*, 39(1):83–94, January 2007.

[51] N. Christofides. Worst-case analysis of a new heuristic for the traveling salesman problem. Technical report, Carnegie Mellon Univeristy, 1976.

[52] R.L. Church and R. Sexton. Modeling small area evacuation: Can existing transportation infrastructure impede public safety? Technical report, Caltrans Testbed Center for Interoperability Task Order 3021, April 2002.

[53] B.N. Clark, C.J. Colbourn, and D.S. Johnson. Unit disk graphs. *Discrete Mathematics*, 86:165–177, 1990.

[54] R. Cohen, K. Erez, D. ben Avraham, and S. Havlin. Efficient immunization strategies for computer networks and populations. *Physical Review Letters*, 85:4626, 2000.

[55] D.A. Coley. *An introduction to genetic algorithms for scientists and engineers*. World Scientific, 1999.

[56] C.W. Commander. *Optimization Problems in Telecommunications with Military Applications*. Ph.D. dissertation, University of Florida, August 2007.

[57] C.W. Commander, C.A.S. Oliveira, P.M. Pardalos, and M.G.C. Resende. A greedy randomized algorithm for the cooperative communication problem on ad hoc networks. In *8th INFORMS Telecommunications Conference*, 2006.

[58] C.W. Commander, C.A.S. Oliveira, P.M. Pardalos, and M.G.C. Resende. A one-pass heuristic for cooperative communication in mobile ad hoc networks. In D.A. Grundel, R.A. Murphey, P.M. Pardalos, and O.A. Prokopyev, editors, *Cooperative Systems: Control and Optimization*, pages 285–296. Springer, 2007.

[59] C.W. Commander, P.M. Pardalos, V. Ryabchenko, and S. Uryasev. The wireless network jamming problem. *Journal of Combinatorial Optimization*, published online, DOI 10.1007/s10878-007-9071-7, 2007.

[60] C.W. Commander, P.M. Pardalos, O. Shylo, and S. Uryasev. Recent advances in eavesdropping and jamming communication networks. In D.A. Grundel, R.A. Murphey, P.M. Pardalos, and O.A. Prokopyev, editors, *6th International Conference on Cooperative Control and Optimization*, pages 101–112. World Scientific, 2006.

[61] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 2001.

[62] T.J. Cova and J.P. Johnson. Microsimulation of neighborhood evacuation in urban wildland interface. *International Journal of Geographical information System*, 11(8):763–784, 2002.

[63] ILOG CPLEX. http://www.ilog.com/products/cplex, Accessed October 2006.

[64] S. Dafermos. Traffic equilibria and variational inequalities. *Transportation Science*, 14:42–54, 1980.

[65] F.C. Daganzo. The cell transmission model: A dynamic representation of highway traffic consistent with the hydrodynamic theory. *Transportation Research Part B*, 28B(4):269–287, 1994.

[66] G. Dantzig, R. Fulkerson, and S. Johnson. Solution of a large-scale traveling-salesman problem. *Operations Research*, 2, 1954.

[67] C. Darwin. *The Origin of Species*. Murray, sixth edition, 1872.

[68] F.N. de Silva, R.W. Eglese, and M. Pidd. *Evacuation planning and spatial decision making: designing effective spatial decision support systems through integration of technologies*, pages 358–373. IGI Publishing, Hershey, PA, USA, 2003.

[69] M. Desrochers and G. Laporte. Improvements and extensions to the miller-tucker-zemlin subtour elimination constraints. *Operations Research Letters*, 10:27–36, 1991.

[70] D. Dreier. Barabasi graph generator v1.4. http://www.cs.ucr.edu/ ddreier, Accessed November 2006.

[71] L. Elefteriadou. Highway capacity. In M. Kutz, editor, *Handbook of Transportation Engineering*, chapter 8, pages 8–1 – 8–17. McGraw-Hill, 2004.

[72] P. Festa, P.M. Pardalos, L. Pitsoulis, and M.G.C. Resende. GRASP with path-relinking for the weighted MAXSAT problem. *ACM Journal of Experimental Algorithmics*, accepted, 2006.

[73] P. Festa and M.G.C. Resende. GRASP: An annotated bibliography. In C. Ribeiro and P.Hansen, editors, *Essays and surveys in metaheuristics*, pages 325–367. Kluwer Academic Publishers, 2002.

[74] L. Fleischer and M. Skutella. *The quickest multicommodity flow problem*, pages 36–53. Lecture Notes in Computer Science. Springer, 2002.

[75] R.W. Floyd. Algorithm 97 (shortest path). *Communications of the ACM*, 5(6):345, 1962.

[76] FL. R. Ford and D. R. Fulkerson. *Flows in Networks*. Princeton University Press, Princeton, New Jersey, 1962.

[77] L.R. Ford and D.R. Fulkerson. Constructing maximal dynamic flows from static slows. *Operations Research*, 6:419–433, 1958.

[78] L.C. Freeman. Centrality in social networks I: Conceptual Clarification. *Social Networks*, 1:215–239, 1979.

[79] T.L. Friesz, D. Bernstein, T.E. Smith, R.L. Tobin, and B.W. Wie. A variational inequality formulation of the dynamic network user equilibrium problem. *Operations Research*, 41(1):179–191, 1993.

[80] H. Fu and C.G. Wilmot. A sequential logit dynamic travel demand model for hurricane evacuation. *Transportation Research Record*, 1882:19–26, 2004.

[81] H. Fu and C.G. Wilmot. Modeling the hurricane evacuation response curve. *TRANSPORTATION RESEARCH RECORD*, 2022:94–102, 2007.

[82] M. R. Garey and D. S. Johnson. *Computers and Intractability - A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, 1979.

[83] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, 1979.

[84] M.R. Gary and D.S. Johnson. *A guide to Theory of NP-Completeness*. W.H. Freeman and Company, 1979.

[85] F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 5:533–549, 1986.

[86] F. Glover, T. Klastorin, and D. Klingman. Optimal weighted ancestry relationships. *Management Science*, 20:B1190–B1193, 1974.

[87] F. Glover, M. Laguna, and R. Martí. Fundamentals of scatter search and path-relinking. *Control Cybernetics*, 39:653–684, 2000.

[88] H. Golani and S.T. Waller. Combinatorial approach multiple destination user optimal dynamic traffic assignment. *Transportation Research Record*, 1882:70–78, 2004.

[89] A. V. Goldberg and R. E. Tarjan. Finding minimum-cost circulations by cancelling negative cycles. *Journal of ACM*, 36:873–886, 1989.

[90] D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Kluwer Academic Publishers, 1989.

[91] G. Gomes, L. Munoz, C.J. Yi, Toy, S. Cinnamon, R. Horowitz, and L. Alvarez. Meso-microscale traffic simulation of an ahs control architecture. In *Proceedings of the American Control Conference*, pages 1806–1811, 2001.

[92] R.E. Gomory and T.C. Hu. Multi-terminal network flows. *Journal of SIAM*, 9(4):551–570, 1961.

[93] J.F. Gončalves and J.R. Almeida. A hybrid genetic algorithm for assemply line balancing. *Journal of Heuristics*, 8:629–642, 2002.

[94] J.F. Gonçalves, J.J.M. Mendes, and M.G.C. Resende. A hybrid genetic algorithm for the job shop scheduling problem. *European Journal of Operations Research*, 167:77–95, 2005.

[95] J.F. Gonçalves, J.J.M. Mendes, and M.G.C. Resende. A random keys based genetic algorithm for the resource constrained project scheduling problem. *Computers and Operations Research*, accepted, 2006.

[96] J.H.M. Goossens, C.P.M. Hoesel, and L.G. Kroon. On solving multi-type line planning problems. METEOR Research Memorandum RM/02/009, University of Maastricht, Maastricht, The Netherlands, 2002.

[97] J. L. Gross and J. Yellen. *Handbook of Graph Theory (Discrete Mathematics and Its Applications)*. CRC, New York, 2003.

[98] M. Grötschel, M. Jünger, and G. Reinelt. A cutting plane algorithm for the linear ordering problem. *Operations Research*, 32:1195–1220, 1984.

[99] M. Grötschel, M. Jünger, and G. Reinelt. On the acyclic subgraph polytope. *Mathematical Programming*, 33:28–42, 1985.

[100] D.A. Grundel and D.E. Jeffcoat. Formulation and solution of the target visitation problem. In *Proceedings of the AIAA 1st Intelligent Systems Technical Conference*, 2004.

[101] D.A. Grundel, R.A. Murphey, and P.M. Pardalos, editors. *Theory and Algorithms for Cooperative Systems*. World Scientific, 2004.

[102] G. M. Guisewite and P. M. Pardalos. Minimum Concave-Cost Network Flow Problems: Applications, Complexity, and Algorithms. *Annals of Operations Research*, 25:75–100, 1990.

[103] G. Gutin and A. Punnen. *The Traveling Salesman Problem and Its Variations*. Kluwer Academic Publishers, Dordrecht, 2002.

[104] S. Gwynne, E. R. Galea, M. Owen, P.J. Lawrence, and L. Filippidis. A review of the methodologies used in the computer simulation of evacuation from the built environment. *Building and Environment*, 34(6):741–749, 1999.

[105] B. Hajek and R. G. Ogier. Optimal dynamic routing in communication networks with continuous traffic. *Networks*, 14:457–487, 1984.

[106] G. L. Hamza-Lup, K.A. Hua, M. Lee, and R. Peng. Enhancing intelligent transportation systems to improve and support homeland security. In *Proceedings of the 7th International IEEE Conference on Intelligent Transportation Systems*, pages 250–255, 2004.

[107] P.R. Harper, V. de Senna, I.T. Vieira, and A.K. Shahani. A genetic algorithm for the project assignment problem. *Computers and Operations Research*, 32:1255–1265, 2005.

[108] P. Hartman and G. Stampacchia. On some nonlinear elliptic diferential functional equations. *Acta Mathematica*, 115:271–310, 1966.

[109] D. Helbing, I. Farkas, and T. Vicsek. Simulating dynamical features of escape panic. *Nature*, 407:487, 2000.

[110] F.S. Hillier and G.J. Lieberman. *Introduction to Operations Research*. McGraw Hill, 2001.

[111] M.J. Hirsch. *GRASP-based heuristics for continuous global optimization problems*. Ph.D. dissertation, University of Florida, December 2006.

[112] M.J. Hirsch, C.N. Meneses, P.M. Pardalos, and M.G.C. Resende. Global optimization by Continuous GRASP. *Optimization Letters*, 1(2):201–212, 2007.

[113] J.K. Ho. A successive linear optimization approach to dynamic traffic assignment. *Transportation Science*, 14(4):295–305, 1980.

[114] E.J. Holguin-Veras, N. Perez, S.V. Ukkusuri, T. Wachtendorf, and B. Brown. Emergency logistics issues affecting response to hurricane katrina: Synthesis and preliminary suggestions for improvement. *Transportation research record*, 2022:76–82, 2007.

[115] B. Hoppe and E. Tardos. Polynomial time algorithms for some evacuation problems. In *SODA '94: Proceedings of the fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 433–441. Society for Industrial and Applied Mathematics, 1994.

[116] B. Hoppe and E. Tardos. The quickest transshipment problem. In *SODA '95: Proceedings of the sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 512–521. Society for Industrial and Applied Mathematics, 1995.

[117] B. E. Hoppe. *Efficient Dynamic Network Flow Algorithms*. PhD thesis, Cornell University, 1995. http://www.math.tu-berlin.de/~skutella/hoppe_thesis.ps.gz.

[118] R. Horst, P.M. Pardalos, and N.V. Thoai. *Introduction to Global Optimization*, volume 3 of *Nonconvex Optimization and its Applications*. Kluwer Academic Publishers, 1995.

[119] R.M. Karp. Reducability among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[120] D. Kim and P. M. Pardalos. Dynamic Slope Scaling and Trust Interval Techniques for Solving Concave Piecewise Linear Network Flow Problems. *Networks*, 35(3):216–222, 2000.

[121] S. Kim and S. Shekhar. Contraflow Network Reconfiguration for Evaluation Planning: A Summary of Results. In *Proceedings of the 13th annual ACM international workshop on Geographic information systems*, pages 250–259, 2005.

[122] B. Klinz and G.J. Woeginger. Minimum-cost dynamic flows: The series-parallel case. *Networks*, 43:153–162, 2004.

[123] V. Krebs. Uncloaking terrorist networks. *First Monday*, 7(4), April 2002.

[124] M.S. Krishnamoorthhy and N. Deo. Node-deletion NP-complete problems. *SIAM Journal of Computing*, 8(4):619–625, 1979.

[125] S. Krumke, H. Noltemeier, S. Schwarz, H. Wirth, and R. Ravi. Flow Improvement and Network Flows with Fixed Costs. In *In Proceedings of the International Conference of Operations Research Züirich (0R'98)*, pages 158–167, 1998.

[126] E.D. Kuligowski and R.D. Peacock. Review of building evacuation models. Technical report, National Institute of Standards and Technology, July 2005.

[127] A. Langevin, F. Soumis, and J. Desrosiers. Classification of traveling salesman problem formulations. *Operations Research Letters*, 9:127–132, 1990.

[128] T. Larsson and D. Yuan. An augmented lagrangean algorithm for large scale multicommodity routing. *Computational Optimization and Applications*, 27(2):187–215, 2004.

[129] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys, editors. *The Traveling Salesman Problem*. Wiley, 1985.

[130] H.W. Lenstra. The acyclic subgraph problem. Technical Report Report BW26, Mathematisch Centrum, Amsterdam, 1973.

[131] J. Lewis and M. Yannakakis. The node-deletion problem for hereditary properties is np-complete. *J. Comput. System Sci.*, 20(2):219–230, 1980.

[132] H.X. Liu, X. He, and X. Ban. Cell-based many-to-one dynamic system optimal model and its heuristic solution method for emergency evacuation. In *86th Transportation Research Board Annual Meeting, Washington, D.C.*, pages Paper 07–2261, 20 pages, 2007.

[133] Y. Liu, X. Lai, , and G-L. Chang. A cell-based network optimization model for staged evacuation planning under emergencies. *Transportation Research Record*, 1964:127–135, 2006.

[134] Y. Liu, X. Lai, and G-L. Chang. Two-level integrated optimization system for planning of emergency evacuation. *Journal of transportation Engineering*, 10(10):800–807, october 2006.

[135] Y. Liu, N. Zou, and G-L. Chang. An integrated emergency evacuation system for real-time operations - a case study of ocean city, maryland under hurricane attacks. In *In the proceedings of Intelligent Transportation Systems, IEEE*, pages 464– 469, 2005.

[136] C. Lund and M. Yannakakis. The approximation of maximum subgraph problems. In *ICALP '93: Proceedings of the 20th International Colloquium on Automata, Languages and Programming*, pages 40–51, London, UK, 1993. Springer-Verlag.

[137] T.C. Matisziw and A.T. Murray. Modeling s-t path availability to support disaster vulnerability assessment of network infrastructure. *Computers and Operations Research*, in press 2008. doi:10.1016/j.cor.2007.09.004.

[138] N. Megiddo. Combinatorial optimization with rational objective functions. *Mathematics of Operations Research*, 4:414–424, 1979.

[139] V. Melkonian. Flows in dynamic networks with aggregate arc capacities. *Information Processing Letters*, 101(1):30–35, 2007.

[140] D.K. Merchant and G.L. Nemhauser. A model and an algorithm for the dynamic traffic assignment problems. *Transportation science*, 12(3):183–199, August 1978.

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

[142] E. Miller-Hooks and S.S. Patterson. On solving quickest time problems in time-dependent, dynamic networks. *Journal of Mathematical Modelling and Algorithms*, 3(1):39–71, 2004.

[143] E. Minieka. Maximal, lexicographic, and dynamic network flows. operations research. *Transportation Research Record*, 21:517–527, 1973.

[144] M. Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, 1996.

[145] S.W. Mitchell and A.E. Radwan. Heuristic priority ranking of emergency evacuation staging to reduce clearance time. *Transportation Research Record*, 1964:219–228, 2006.

[146] K.D. Moriarty, D. Ni, and J. Collura. Modeling traffic flow under emergency evacuation situations: Current practice and future directions. In *86th Transportation Research Board Annual Meeting, Washington, D.C.*, pages Paper 07–0745, 15 pages, 2007.

[147] R.A. Murphey and P.M. Pardalos, editors. *Cooperative Control and Optimization*. Springer, 2002.

[148] P.M. Murray and H.S. Mahmassani. Model for household trip chain sequencing in an emergency evacuation. In *In the proceedings of 82nd TRB Annual Meeting*, pages 21–29, 2004.

[149] Y.S. Myung and H.J. Kim. A cutting plane algorithm for computing k-edge survivability of a network. *European Journal of Operational Research*, 156:579–589, 2004.

[150] A. Nagurney. *Network Economics: A Variational Inequality Approach*. Springer, 1999.

[151] H. Narayanan, S. Roy, and S. Patkar. Approximation algorithms for min-k-overlap problems using the principal lattice of partitions approach. *Journal of Algorithms*, 21(2):306–330, 1996.

[152] C.A.S. Oliveira and P.M. Pardalos. An optimization approach for cooperative communication in ad hoc networks. Technical report, School of Industrial Engineering and Management, Oklahoma State University, 2005.

[153] C.A.S. Oliveira, P.M. Pardalos, and T.M. Querido. Integer formulations for the message scheduling problem on controller area networks. In D. Grundel, R. Murphey, and P. Pardalos, editors, *Theory and Algorithms for Cooperative Systems*, pages 353–365. World Scientific, 2004.

[154] C.A.S. Oliveira, P.M. Pardalos, and T.M. Querido. A combinatorial algorithm for message scheduling on controller area networks. *International Journal of Operations Research*, 1(1/2):160–171, 2005.

[155] S. Opasanon. *On finding paths and flows in multicriteria, stochastic, and time-varying networks*. PhD thesis, Department of Civil Engineering and Environmental Engineering, University of Maryland, College Park, 2004.

[156] J. B. Orlin. Maximum-throughput dynamic network flows. *Mathematical Programming*, 27:214–231, 1983.

[157] K. Ozbay and E.E. Ozguven. A stochastic humanitarian inventory control model for disaster planning. In *86th Transportation Research Board Annual Meeting, Washington, D.C.*, pages 63–75, 2007.

[158] M. Padberg and T. Sung. An analytical comparison of different formulations of the traveling salesman problem. *Mathematical Programming*, 52:315–357, 1991.

[159] C.H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[160] C.H. Papadimitriou and S. Vempala. On the approximability of the traveling salesman problem. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computation*, pages 126–133, 2000.

[161] V. Pareto. *Cours d'economie politique*. Geneva: Libraire Drott, 1964.

[162] M.J. Parker and A. Ryan. A column generation algorithm for bandwidth packing. *Telecommunications Systems*, 2:185–195, 1993.

[163] H.J. Payne. Models of freeway traffic and control. In *Math. Models Publ. Sys., Simul. Council Proc.*, volume 28, pages 51–61, 1971.

[164] S. Peeta and A. Ziliaskopoulos. Foundations of dynamic traffic assignment: Past, present and the future. *Networks and Spatial Economics*, 1(3-4):233–265, 2001.

[165] M.E. Pfetsch and R.Borndrfer. Routing in line planning for public transport. In Herbert Kopfer Hans-Dietrich Haasis and Jrn Schnberger, editors, *Operations Research Proceedings*, pages 405–410. Springer, 2005.

[166] M. Pidd, F.N. de Silva, and R.W. Eglese. Cemps: A configurable evacuation management and planning system. In *Proceedings of the Winter Simulation Conference*, pages 1319–1323, Dec 1993.

[167] F. Raciti and L.Scrimali. Time-dependent variational inequalities and applications to equilibrium problems. *J. of Global Optimization*, 28(3-4):387–400, 2004.

[168] A.E. Radwan, A.G. Hoebeika, and D. Sivasailam. A computer simulation model for rural network evacuation under natural disaster. *Institute of Transportation Engineers Journal*, 55(9):25–30, September 1985.

[169] E. Radwan, S. Mitchell, and G. Yildirim. Framework for modeling emergency evacuation. Technical report, Center for Advanced Transportation Systems Simulation. University of Central Florida, April 2005.

[170] B. Ran and D. Boyce. *Modelling Dynamic Transportation networks*. Springer, 2 edition, 1996.

[171] S. Rebennack, A. Arulselvan, L. Elefteriadou, and P.M. Pardalos. Complexity analysis for maximum flow problems with arc reversals. Submitted, 2008.

[172] M.G.C. Resende and P.M. Pardalos. *Handbook of Optimization in Telecommunications*. Springer, 2006.

[173] M.G.C. Resende and C.C. Ribeiro. Greedy randomized adaptive search procedures. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, pages 219–249. Kluwer Academic Publishers, 2003.

[174] M.G.C. Resende and R.F. Werneck. A hybrid multistart heuristic for the uncapacitated facility location problem. *European Journal of Operations Research*, 174:54–68, 2006.

[175] M.G.C. Resende and R.F. Werneck. A fast swap-based local search procedure for location problems. *Annals of Operations Research*, published online, DOI: 10.1007/s10479-006-0154-0, 2007.

[176] D.M. Ryan and B. A. Foster. An integer programming approach to scheduling. In A. Wren, editor, *Computer Scheduling of Public Transport: Urban Passenger Vehicle and Crew Scheduling*, pages 269–280. Elsevier Science Ltd., 1981.

[177] H. Sakakibara, Y. Kajitani, and N. Okada. Road network robustness for avoiding functional isolation in disasters. *Journal of transportation Engineering*, 130(5):560–567, 2004.

[178] G. Santos and B.E. Aguirre. A critical review of emergency evacuation simulation models. In *Proceedings of Building Occupant Movement during Fire*, pages 27–52. National Institute of Standards and Technology, 2004.

[179] P. Sattayhatewa and B. Ran. Developing a dynamic traffic management model for nuclear power plant evacuation. In *83rd Transportation Research Board Annual Meeting, Washington, D.C.*, page CDROM, 2004.

[180] H. Sbayti and H.S. Mahmassani. Optimal scheduling of evacuation operations. *Transportation Research Record*, 1964:238–246, 2006.

[181] Y. Sheffi, H. Mahmassani, and W. Powell. *NETVAC: A transportation Network Evacuation Model*. Center for Transportaion Studies, MIT, 1980.

[182] V.P. Sisiopiku, S.L. Jones, A.J. Sullivan, S.S. Patharkar, and X. Tang. Regional traffic simulation for emergency preparedness. Technical report, University Transportation Center for Alabama, The University of Alabama, The University of Alabama in Birmingham, and The University of Alabama in Huntsville, April 2004.

[183] J.M. Smith. Qnet-c: An interactive graphics computer program for evacuation planning. In *Proceedings of the conference on Emergency plannning, SCS multiconference, 14-16*, pages 19–24, Jan 1987.

[184] J.M. Smith. Multiobjective routing in stochastic evacuation network. In P.M. Pardalos D-Z. Du, editor, *Network Optmization Problem*, volume 2, pages 263–281. World Scientific, 1993.

[185] J.M. Smith and D. Towsley. The use of queueing networks in the evacuation of egress from buildings. *Environment and Planning*, B-8:125–139, 1981.

[186] W.M. Spears and K.A. DeJong. On the virtues of parameterized uniform crossover. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 230–236, 1991.

[187] A. Stathopoulos and T. Tsekeris. Enhanced dynamic origin-destination matrix updating with long term flow information. *Transportation Research Record*, 1882:159–166, 2004.

[188] A. P. Tagliaferri. *Use and Comparison of Traffic Simulation Models in the Analysis of Emergency Evacuation Conditions*. PhD thesis, Department of Civil Engineering, North Carolina State University, 2005.

[189] K. Talebi and J.M. Smith. Stochastic network evacuation models. *Computers and Operations Research*, 12(6):559–577, 1985.

[190] G. Theodoulou and B. Wolshon. Alternative methods to increase the effectiveness of freeway contraflow evacuation. *The Journal of Transportation Research Board*, 1865:48–56, 2004.

[191] P.M. Tuite and H.S. Mahmassani. Methodology for determining vulnerable links in transportation network. *Transportation Research Record*, 1882:88–96, 2004.

[192] H. Tuydes and A. Ziliaskopoulos. Network re-design to optimize evacuation contraflow. *Transportation Research Record: Journal of the Transportation Research Board*, 1964:157–168, 2006.

[193] H. Tuydes and A. Ziliaskopoulos. Tabu-based heuristic approach for optimization of network evacuation contraflow. *Transportation Research Record*, 1964:157–168, 2006.

[194] S.T. Waller and A.K. Ziliaskopoulos. Stochastic dynamic network design problem. *Transportation Research Records*, 1771:106–113, 2001.

[195] S. Warshall. A theorem on boolean matrices. *Journal of the ACM*, 9(1):11–12, 1962.

[196] W. L. Wilkinson. An algorithm for universal maximal dynamic flows in a network. *OPerations Research*, 19:1602–1612, 1971.

[197] B. Williams, A. Tagliaferri, S. Meinhold, J. Hummer, and N. Rouphail. Simulation and analysis of freeway lane reversal for coastal hurricane evacuation. *J. Urban Plng. and Devel.*, 133(1):61–72, 2007.

[198] J.M. Wislon. A genetic algorithm for the generalised assignment problem. *Journal of the Operational Reswearch Society*, 48:804–809, 1997.

[199] L. Wolsey. *Integer Programming*. Wiley, 1998.

[200] B. Wolshon, E. Urbina, and M. Levitan. National review of hurricane evacuation plans and policies. louisiana. Technical report, State University Hurricane Center, 2001.

[201] F. Yuan and L.D. Han. Evacuation modeling and operations using dynamic traffic assignment and most-desirable destination approaches. In *Proceedings of TRB 84th Annual Meeting, Transportation Research Board*, page CDROM, 2005.

[202] F. Yuan, L.D. Han, S-M. Chin, and H. Hwang. Proposed framework for simultaneous optimization of evacuation traffic destination and route assignment. *Transportation Research Record*, 1964:50–58, 2006.

[203] F. Yuan, L.D. Han, S-M. Chin, and H. Hwang. Does noncompliance with route and destination assignment compromise evacuation efficiency? In *86th Transportation Research Board Annual Meeting, Washington, D.C.*, pages paper 07–2396, 25 pages, 2007.

[204] T. Zhou, Z-Q. Fu, and B-H. Wang, editors. *Epidemic dynamics on complex networks*. 452-457, Progress in Natural Science, 2006 16.

[205] X. Zhou, S. Erdogan, and H.S. Mahmassani. Dynamic origin-destination trip demand estimation for subarea analysis. In *85th Transportation Research Board Annual Meeting, Washington, D.C.*, pages 176–184, 2006.

[206] A.K. Ziliaskopoulos. A linear programming model for the single destination system optimum dynamic traffic assignment problem. *Transportation science*, 34(1):37–49, February 2000.

[207] N. Zou, S-T. Yeh, G-L. Chang, A. Marquess, and M. Zezeski. Simulation-based emergency evacuation system for ocean city, maryland, during hurricanes. *Transportation Research Record*, 1922:138148, 2005.

BIOGRAPHICAL SKETCH

The author of this dissertation, Mr. Ashwin Arulselvan, is a graduate student at the University of Florida in industrial and systems engineering.