5-2019

Mixed Integer Programming Approaches to Novel Vehicle Routing Problems

Tony Kent Rodriguez
University of Tennessee, trodrig5@vols.utk.edu

To the Graduate Council:

I am submitting herewith a dissertation written by Tony Kent Rodriguez entitled "Mixed Integer Programming Approaches to Novel Vehicle Routing Problems." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Industrial Engineering.

James Ostrowski, Major Professor

We have read this dissertation and recommend its acceptance:

John Kobza, Anahita Khojandi, Michael Langston

Accepted for the Council:

Dixie L. Thompson

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

Mixed Integer Programming Approaches to Novel Vehicle Routing Problems

A Dissertation Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Tony Kent Rodriguez

May 2019

Acknowledgments

I would like to thank my advisor, Professor Jim Ostrowski, for his guidance during my time as a graduate student. His guidance has played an important role in this thesis. I would also like to thank the members of my committee, Professors Anahita Khojandi, John Kobza, and Michael Langston. Your questions, encouragements, and advice have helped make this thesis what it is.

I would like to express gratitude to the University of Tennessee for the funding I received as a graduate student.

I would like to thank the friends I met while here, whose words of encouragement helped make this possible. And lastly, I would like to thank Whitney for moving to Texas, which finally convinced me to hurry up.

Abstract

This thesis explores two main topics. The first is how to incorporate data on meteorological forecasts, traffic patterns, and road network topology to utilize deicing resources more efficiently. Many municipalities throughout the United States find themselves unable to treat their road networks fully during winter snow events. Further, as the global climate continues to change, it is expected that both the number and severity of extreme winter weather events will increase for large portions of the US.

We propose to use network flows, resource allocation, and vehicle routing mixed integer programming approaches to be able to incorporate all of these data in a winter road maintenance framework. We also show that solution approaches which have proved useful in network flows and vehicle routing problems can be adapted to construct high-quality solutions to this new problem quickly. These approaches are validated on both random and real-world instances using data from Knoxville, TN.

In addition to showing that these approaches can be used to allocate resources effectively given a certain deicing budget, we also show that these same approaches can be used to help determine a resource budget given some allocation utility score. As before, we validate these approaches using random and real-world instances in Knoxville, TN.

The second topic considered is formulating mixed integer programming models which can be used to route automated electric shuttles. We show that these models can also be used to determine fleet composition and optimal vehicle characteristics to accommodate various demand scenarios. We adapt popular vehicle routing solution techniques to these models, showing that these strategies continue to be relevant and robust. Lastly, we validate these techniques by looking at a case study in Greenville, SC, which recently received a grant from

the Federal Highway Administration to deploy a fleet of automated electric shuttles in three neighborhoods.

Table of Contents

List of Figures

Chapter 1

Introduction

This chapter introduces some fundamental concepts that will be further explored through this thesis. We begin by presenting a brief overview of integer programming and some of the reasons that it has become popular as a modeling paradigm in recent years. We then turn our attention to a particular problem which has benefited from this paradigm, which is the vehicle routing problem.

1.1 Integer Programming

An integer programming (IP) problem is a mathematical optimization problem in which some of the variables are restricted to be integers. This restriction allows for a diverse set of scenarios to be modeled, ranging from how many discrete widgets a company should produce so as to maximize profits to which road segments a delivery vehicle should traverse in order to deliver said widgets to a set of customers in the most efficient manner.

In this work our focus will be on integer linear programs (ILP), which are integer programs in which the objective function and all constraints can be expressed as linear functions. With the added assumption of linearity, an ILP is an optimization problem that can be written in the following matrix-vector form:

$$\min c^T x + d^T y \tag{1.1}$$

subject to:

$$Ax + Gy \leq b \tag{1.2}$$

$$x \in \mathbb{R}^n_+, y \in \mathbb{Z}^p_+, \tag{1.3}$$

where $c \in \mathbb{Q}^n$, $d \in \mathbb{Q}^p$, $A \in \mathbb{Q}^{m \times n}$, $G \in \mathbb{Q}^{m \times p}$, and $b \in \mathbb{Q}^m$. We note the slight abuse of notation in (1.2). In this context, $u \leq v$ for two vectors u and v is taken to mean each component of u is less than or equal to the corresponding component of v. A *pure integer program* is when $n = 0$ (there are no continuous variables), and a *mixed integer program* (MILP) is when $n > 0$ and $p > 0$ (there are both continuous and integer variables). The set $\{(x, y) \mid Ax + Gy \leq b, x \in \mathbb{R}^n_+, y \in \mathbb{Z}^p_+\}$ is called the *feasible region*.

We note that, even in the restricted case of the y variables being binary (as opposed to general integers), determining if the feasible region of a MILP is non-empty is one of Karp's classical 21 NP-complete problems [28]. Clearly, then, solving a MILP is NP-hard. However, if y is fixed (or if $p = 0$), then (1.1) - (1.3) is a linear program, which can be solved in polynomial time [27]. A more careful consideration of linear programming can be found in Bertsimas and Tsitsiklis [8].

Even though many aspects of practical problems are inherently non-linear [39], MILP still represents a powerful tool to address these problems. In addition to the obvious decisions that can be modeled with integer variables, such as the workers that should be on a shift in order to satisfy demand, decisions regarding yes/no questions can also be modeled under this paradigm. Modeling yes/no decisions through binary variables allows a host of operational decisions to be captured and addressed in MILP models.

An example of a problem which MILP has had great success with is the traveling salesman problem (TSP). The classical TSP asks the following question: given a set of cities, what is the shortest route to visit every city and return to the city of origin? We present the formulation given by Dantzig, Fulkerson, and Johnson in their groundbreaking work [14]. Given a set of cities $\{1, 2, ..., n\}$ and the distance for each pair of cities i, j given as c_{ij}, the MILP formulation for the TSP is as follows:

$$\min \sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} c_{ij} x_{ij} \tag{1.4}$$

subject to:

$$\sum_{i=1, i \neq j}^{n} x_{ij} = 1 \qquad \forall j = 1, 2, ..., n \tag{1.5}$$

$$\sum_{j=1, j \neq i}^{n} x_{ij} = 1 \qquad \forall i = 1, 2, ..., n \tag{1.6}$$

$$\sum_{i \in \mathcal{C}} \sum_{j \in \mathcal{C}} x_{ij} \leq |C| - 1 \qquad \forall \mathcal{C} \subseteq \{2, 3, ..., n\} \tag{1.7}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j = 1, 2, ..., n \tag{1.8}$$

In (1.4) - (1.8), x_{ij} encodes if the path goes from city i directly to city j. We also note that the primary technique put forward in [14] to solve a continental United States-spanning 49 city TSP instance, namely the use of cutting planes, is still in use today.

The encoding of yes/no questions in binary variables also allows many separate questions to be addressed in the same MILP model. This can be important as many operational decisions in a practical setting can impact other decisions that must be made, sometimes even in other systems. This aspect of MILP is important for Chapters 2, 3, and 4, as each of the problems considered in this work have multiple systems which require operational decisions to be made.

1.2 The Vehicle Routing Problem

The vehicle routing problem (VRP) is that of determining an optimal set of routes a fleet of vehicles must traverse in order to make deliveries to a set of customers. First introduced by Dantzig and Ramser in [15], where it was called the truck dispatching problem, the vehicle routing problem has become an important and well-studied problem within the fields of combinatorial optimization and graph theory. The problem of efficiently routing vehicles is a problem faced throughout several industries on a daily basis. Because of this, the economic and ecological impacts are significant. Recently, the prestigious 2016 INFORMS Edelman Award was given to UPS for their work on integrating optimization approaches into their daily operations [25]. With their work on the VRP and optimizing other business activities, UPS estimates their new approaches will save up to \$400 million annually. Further, by routing trucks more efficiently, they expect their CO2 emissions to drop by 100,000 tons annually.

One of the most common ILP formulations for the VRP is the traveling salesman generalization presented in [15]. In this formulation, it is assumed the input graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ has as its vertex set $\mathcal{V}$ the set of customers and the vehicle depot D. It is assumed that this graph is complete, with $(i, j) \in \mathcal{A}$ representing the path necessarily to go from the location corresponding to node i to the location corresponding to node j. We assume c_{ij} is the cost associated with traversing this path from i to j. Thus, given $\mathcal{G}$ and the corresponding c_{ij} values, together with the number of vehicles (or routes) available K, the VRP is

$$\min \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{V}} c_{ij} x_{ij} \tag{1.9}$$

subject to:

$$\sum_{i \in \mathcal{V}} x_{ij} = 1 \qquad \forall j \in \mathcal{V} \setminus \{D\} \tag{1.10}$$

$$\sum_{j \in \mathcal{V}} x_{ij} = 1 \qquad \forall i \in \mathcal{V} \setminus \{D\} \tag{1.11}$$

$$\sum_{i \in \mathcal{V}} x_{iD} = K \tag{1.12}$$

$$\sum_{i \in \mathcal{V}} x_{Di} = K \tag{1.13}$$

$$\sum_{i \notin C} \sum_{j \in C} x_{ij} \geq r(C) \qquad \forall C \subseteq V \setminus \{D\}, C \neq \emptyset \tag{1.14}$$

$$x_{ij} \in 0, 1 \qquad \forall i, j \in \mathcal{V} \tag{1.15}$$

In this formulation x_{ij} represents whether some route contains the path from node i to node j. Constraints 1.10 and 1.11 ensure that only a single vehicle arrives and departs from each customer. Constraints 1.12 and 1.13 force exactly K vehicles to leave and arrive at the depot. The constraints in 1.14 are especially important in VRP formulations. These ensure that the routes are connected, which, together with 1.12 and 1.13, makes sure that each route also includes the depot. As they eliminate *subtours*, they are sometimes referred to as *subtour elimination constraints*. We note that there are exponentially many of these constraints. These constraints, and others which have been designed to cut off solutions which contain subtours, have received much attention in the past [32, 11, 37]. In general, the VRP is an NP-hard problem [13].

Since the VRP is a generalization of the traveling salesman problem, it is tempting to think that the same solution approaches that have proved useful for that problem would also be useful for solving VRP instances. This is often done with a combination of heuristics and enforcing the constraints in 1.14 in a lazy fashion (i.e. only add the constraints which are violated by candidate solutions). With these approaches, TSP instances with thousands of nodes can often be solved on modern hardware, with the largest instance solved to date having 85,900 nodes [7, 45, 5]. While these approaches have seen some success with VRP, the sizes of instances which can be reliably solved is often limited to hundreds of nodes [33]. Because of the applicability of VRP problems, it is often necessary to produce high-quality solutions to VRP instances with thousands of nodes in short periods of time. This being the case, many heuristic solution approaches have seen wide use over the years.

One of the first heuristics developed for VRP instances remains one of the most popular to date. This is the savings heuristic, developed by Clarke and Wright [12]. This likely remains a popular approach because the idea is simple, implementations tend to be straight-forward, and it often yields good results, despite the fact that it lacks a refinement phase. Roughly speaking, this heuristic first produces $|\mathcal{V}| - 1$ routes of the form (D, i, D) for $i \in \mathcal{V} \setminus \{D\}$. That is, each route has the vehicle leave the depot, visit customer i, then immediately return to the depot. After this initialization, at each iteration, two routes are chosen and merged so as to maximize the reduction in cost (or savings) of the merge. This is repeated until no more feasible merges are possible, or until the pre-defined number of routes has been achieved.

The savings algorithm presented by Clarke and Wright is a greedy heuristic. As such, while it tends to be fast, it also can perform poorly. In a sense, as this algorithm takes pairs of routes and merges them, one pair at a time, this is similar to trying to solve a maximum weight graph matching problem by iteratively selecting edges of maximum weight and adding them to the matching. Viewing the savings algorithm in this way, one obvious modification is to compute the pairwise savings of each route merge then find the maximum weight matching in the corresponding graph. For more details on this approach, see [16].

In addition to various savings heuristics, another important class of VRP algorithms are cluster first, route second algorithms. Like the savings algorithms, most of these are purely constructive, having no refinement phase apart from potentially rerunning the same algorithm but with slightly different parameters. As the name would suggest, the main idea behind these approaches are to first cluster the customers according to some approach, and then to form the individual routes in the second phase. For standard VRP instances, given a partition of the customers, the routes can be constructed by solving the corresponding TSP instance. The main difference between these approaches then is in how the customers are clustered. Fisher and Jaikumar form the clusters by solving a generalized assignment problem [21], while Ryan, Hjorring, and Glover determine the clusters by considering a set partitioning problem [48]. When the nodes of the graph have an obvious and meaningful embedding in the plane, Renaud and Boctor cluster the customers according to their angles from the depot [46].

In addition to constructing routes, many successful heuristic approaches refine the routes after construction. These refinements take two forms: interroute refinements, and intraroute refinements. Interroute refinements attempt to find better clustings of the customers, while intraroute refinements attempting to find better routes under the current clustering. It is common for a refinement strategy to alternate between interroute and intraroute refinements. In the event that the routes can be proven to be optimal under a given clustering, obviously intraroute refinements are unnecessary. While this can often be done in VRP instances such as the one presented in 1.9, modern variations of the VRP often have several additional complicating aspects.

Whether performing interroute or intraroute refinements, some of the most popular approaches are metaheuristics rather than VRP-specific searches. Local searches have been especially popular. This is likely due to the same reasons the savings heuristic remains popular: they are often simple to understand, simple to implement, and perform well. One such solution method is the Tabu search put forward by Glover [24]. Tabu search is a type of iterative local search that is often quite good at escaping locally optimal points which are not globally optimal. It does this by allowing the next solution at the next iteration to have a non-improving (or even strictly worse) objective value. Because of this, cycling could easily become a problem. In order to prevent cycling, some characteristic of the current solution is stored (this characteristic being tabu), and for some number of iterations, solutions with the given characteristic will not be considered.

In Chapters 2 and 3, we show that heuristic approaches which are similar to the savings heuristic can be useful in routing deicing vehicles to cover a set of arcs within a graph. Furthermore, in Chapter 4, we compare simple on-demand constructive heuristics with a Tabu search in the context of routing automated shuttles to pick up and deliver customers in an on-demand fashion.

Chapter 2

Allocating Limited Deicing Resources in Winter Snow Events

This chapter is based on a paper submitted by Tony K. Rodriguez, Olufemi Omitaomu, and James A. Ostrowski:

> Tony K. Rodriguez, Olufemi Omitaomu, James A. Ostrowski. Allocating Limited Deicing Resources in Winter Snow Events. *Submitted to Journal on Vehicle Routing Algorithms.*

Authors Omitaomu and Ostrowski posed the question. Authors Rodriguez and Ostrowski developed the algorithmic framework. Author Rodriguez developed the software, ran the computational experiments, wrote the manuscript, and created all tables and figures. Authors Omitaomu and Ostrowski edited the manuscript.

In this chapter we develop a novel method for allocating resources in the context of reinforcing a network to make it more robust. This approach is based off of network flows with difficult complicating subproblems, in this case vehicle routing. In addition to a new mixed integer programming formulation for the problem, we develop a constructive heuristic for the problem. We implement these approaches and show the strategies are better than the currently used methods.

2.1 Introduction

Every winter, many cities face significant costs due to winter storms and road maintenance. Snow and ice removal from roadways constitute a large portion of these costs, with the direct costs totaling about $1.5 billion annually in the United States of America [17]. In addition to these extensive direct costs, several indirect costs also exist. Improperly managed snow and ice events can result in obstructed and unsafe roadways, which leads to prolonged business, school, and local government closures, in addition to an increase in traffic accidents. Furthermore, the use of salt and other deicing chemicals causes damage to roads and corrosion to infrastructure and vehicles [18]. As a result, intelligently managing winter road maintenance fleets and decisions about which roads to treat by utilizing traffic data and environmental factors can result in substantial increases to the well-being of a community.

The problem considered in this work is a variation of the Snow Plow Routing Problem. This is a generalization of the Chinese Postman Problem [36] and the Capacitated Arc Routing with Intermediate Facilities problem [23]. Often it is assumed that enough resources are present to treat the road network under consideration completely. As a result, many of the works that consider these problems have as their objective to minimize the monetary cost of treating the road network, the time cost of treating the road network, the cost of maintaining the fleet required to treat the road network, or some metric meant to capture inefficiencies in treatment schedules [41, 42, 43, 44, 29]. While much research has been directed towards problems of this kind, many works do not consider resource limiting constraints [29]. Since many municipalities throughout the United States lack the resources to treat their roads completely during a snowfall event, these models have limited utility in practice.

The focus of this work is to provide the City of Knoxville, TN with a rigorous framework and methodology to manage its snow removal and deicing planning. Like many municipalities in the southeastern United States, the City of Knoxville lacks the resources to treat the entire road network during each snowfall event. The aim of this research is to determine which roads to treat and the degree to which they should be treated to utilize the limited resources available more efficiently. This paper lays the groundwork by defining the problem and introducing a mixed integer programming (MIP) formulation of the problem, as well as a constructive heuristic that performs well on the instances considered. The remainder of the paper is structured as follows. In Section 2.2, the problem is formally defined. The MIP formulation of the problem is contained in Section 2.3. Section 2.4 describes a heuristic approach to solve the problem. Section 2.5 outlines the experimental procedure and results, and Section 2.6 draws some conclusions.

2.2 Deicing Vehicle Routing Problem

Deicing chemicals, such as brine solutions and common road salt, together with plowing are effective tools for snow and ice removal. Because many municipalities lack the resources to treat every road within their boundaries completely, there are two significant limitations with many snow removal systems: (i) the same amount of deicing material is applied to every road, and (ii) the treated roads are preselected based only on traffic counts. In practice, this results in streets with a high traffic volume being treated, while feeder streets, trouble spots, and neighborhood roads often go untreated. Consequently, many residents are unable to make it safely to these treated roads, lowering the overall utility gained from the treatment. However, simply placing roads with trouble spots higher in priority to be treated could also be unfavorable, as these roads may have little traffic. Further, some high-traffic streets require little to no treatment due to environmental conditions at that location. These factors together lead to some road segments being over treated, while entire swaths of the road network go untreated. In a sense, while many approaches consider traffic data as well as driving and weather conditions, none of them utilize them together in a cohesive manner. As a result, these approaches can fail to identify the bottlenecks present in the road network that citizens will encounter during snowfall events.

To address these issues, the following approaches are proposed. Data on solar radiation of road segments and local weather forecasts, together with slope data about a road network, can be used to quantify the fact that not all roads have equally bad driving conditions during snowfall events. The solar radiation data and weather forecasts can be used in an energy-balance framework [38] to estimate the amount of snow throughout the road network. This information, together with slope data, is used to compute the *Road Vulnerability Index*

(RVI) of a road segment to a given snow event. RVI is meant to be a measure of how a snow event impedes travel along a road segment, with higher RVI values corresponding to more dangerous driving conditions. While we have access to these data sets for our Knoxville, TN case study, we acknowledge that many municipalities will have access to different data, as well as having different metrics by which they measure driving conditions. As a result, our focus in this work is not on the specifics of computing the RVI of road segments, but rather the mixed integer programming model that incorporates these RVI values. RVI values should account for the fact that driving conditions are not equally bad on all road segments during a winter snow event, with higher values corresponding to worse driving conditions. Additional approaches and potential considerations for vulnerability scores can be found in [50, 6, 49, 47].

To find the right balance of treating high-traffic streets versus trouble spots and neighborhood roads, a two-stage network flow model is presented to model traffic on the road network. It is assumed that the traffic capacity of a road segment is bounded by a function of the traffic count of that road segment under normal driving conditions, the RVI of the road segment, and the degree to which the road segment has been treated. This, together with population and traffic data, allows for the modeling of sources and sinks, as well as various kinds of traffic in the network. The objective is to find a snow plow route that will maximize the flow through the road network after treatment, subject to salt, fuel, and time constraints. This allows for the identification not only of which roads should be treated but also the amount of deicing solution that should be applied to the treated roads.

2.3 MIP Formulation

The problem studied consists of routing deicing vehicles to a subset of the roads in the network in a way that maximizes the utility of the treated network. These routes must obey constraints on the salt and fuel capacities of the vehicles, as well as constraints on the total salt and fuel usage and total time allotted.

Let $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ be a multigraph, where the vertex set $\mathcal{V}$ represents the intersections and various locations within the city, and the arc set $\mathcal{A}$ represents the road segments of the road network. We denote by $\mathcal{S} \subset \mathcal{V}$ the set of residences within the city and will refer to these as supply nodes, and by $\mathcal{D} \subset \mathcal{V}$ the set of commercial and government locations, that we will call demand nodes. Also let $Dep \in \mathcal{V}$ be the fuel and salt depot location. In the model presented, we assume a single depot, but this can be easily generalized to multiple depots by using techniques similar to those in [29]. Further, let $\mathcal{P}$ be the set of heterogenous deicing vehicles. We denote the salt capacity of vehicle $p \in \mathcal{P}$ as Q_S^p and the fuel capacity as Q_F^p.

For an arc $(i,j) \in \mathcal{A}$, let $TC_{i,j}$ and $RVI_{i,j}$ be the traffic count and RVI, respectively, for the corresponding road segment. Let $S_{i,j}^{max}, S_{i,j}^{min}$ be the maximum and minimum, respectively, amount of deicing material allowed for road segment (i,j), and S^{total} the total amount of deicing material available. Also, we denote by $T_{i,j}$ the amount of time it takes to traverse the road segment and $F_{i,j}^p$ be the amount of fuel vehicle $p \in \mathcal{P}$ uses to traverse road segment (i,j), with T^{total} and F^{total} being the total amount of plow time and fuel, respectively, available. Note that a vehicle may traverse a road segment without treating that road segment. Due to the speeds typically driven within a city, we assume that all vehicles take approximately the same amount of time to traverse a given road segment – that is, travel time across a road segment is a function of the road segment alone.

Below we provide a concise list of the parameters and variables within the mathematical formulation.

$\mathcal{G} = (\mathcal{V}, \mathcal{A})$ Directed multigraph of the road network.

$\mathcal{S}$ Subset of nodes that represent first-stage flow sources/second-stage flow sinks.

$\mathcal{D}$ Subset of nodes that represent first-stage flow sinks/second-stage flow sources.

$\mathcal{P}$ Set of deicing vehicles.

S_i First-stage flow supply/second-stage flow demand at node $i \in \mathcal{S}$.

D_i First-stage flow demand/second-stage flow supply at node $i \in \mathcal{D}$.

$S_{i,j}^{max}$ Maximum amount of deicing material allowed for road segment $(i,j) \in \mathcal{A}$.

$S_{i,j}^{min}$ Minimum amount of deicing material possible for road segment $(i,j) \in \mathcal{A}$.

S^{total} Total amount of deicing material available.

$T_{i,j}$ Amount of time required to traverse road segment $(i,j) \in \mathcal{A}$.

T^{total} Total amount of vehicle time available.

$F^p_{i,j}$ Amount of fuel required for vehicle $p \in \mathcal{P}$ to traverse road segment $(i,j) \in \mathcal{A}$.

F^{total} Total amount of fuel available.

Q^p_F Fuel capacity for vehicle $p \in \mathcal{P}$.

Q^p_S Deicing material capacity for vehicle $p \in \mathcal{P}$.

$TC_{i,j}$ Traffic count for road segment $(i,j) \in \mathcal{A}$.

$RVI_{i,j}$ RVI for road segment $(i,j) \in \mathcal{A}$.

$\gamma_{i,j}, \alpha_{i,j}, \beta_{i,j}$ Weights to account for the impact of traffic count ($\gamma_{i,j}$), RVI ($\alpha_{i,j}$), amount of deicing material ($\beta_{i,j}$) on capacity of a road segment $(i,j) \in \mathcal{A}$.

f Second-stage flow through the network (continuous).

$f_{i,j}$ First-stage flow across arc $(i,j) \in \mathcal{A}$ (continuous).

$f'_{i,j}$ Second-stage flow across arc $(i,j) \in \mathcal{A}$ (continuous).

$y^p_{i,j}$ Number of times vehicle $p \in \mathcal{P}$ traverses road segment $(i,j) \in \mathcal{A}$ (integer).

$x^p_{i,j}$ Indicates whether vehicle $p \in \mathcal{P}$ traverses road segment $(i,j) \in \mathcal{A}$ (binary).

$s^p_{i,j}$ Deicing material applied to road segment $(i,j) \in \mathcal{A}$ by vehicle $p \in \mathcal{P}$ (continuous).

Since we are representing traffic in the road network via network flows, we will consider two flows through the network. These flows are the first- and second-stage flows. The first-stage flow is to represent individuals trying to leave their residence to go to a commercial or government location, while the second-stage flow represents individuals trying to return to their residence from one of these locations. Less formally, the first-stage flow represents the morning traffic, while the second-stage flow represents the evening traffic.

As is typically done in network flow models with several sources and sinks, we add two artificial nodes s and t as dummy source and sink nodes for the simplicity of modeling. Let $s \in \mathcal{V}$ be the dummy source node for the first-stage flow and $t \in \mathcal{V}$ be the dummy sink node for the first-stage flow. We note that the first-stage flows, in a sense, represent traffic in a certain direction, while the second-stage flows represent traffic in the opposite direction. Because of this, t will be the dummy source node for the second-stage flow, while s is the dummy sink for the second-stage flow. We assume, for all $i \in \mathcal{S}$, the arcs (s, i) and (i, s) are in the graph. Similarly, for all $j \in \mathcal{D}$, we assume (j, t) and (t, j) are in the graph. We further assume arcs (s, t) and (t, s) are in the graph, each with capacity of ∞. Since these dummy nodes provide easy-to-find minimum cut sets, they will also be used to limit the flows to and from the residence and destination nodes.

For each arc $(i, j) \in \mathcal{A}$ and each vehicle $p \in \mathcal{P}$, let $y^p_{i,j}$ be the number of times p traverses arc (i, j), $x^p_{i,j}$ indicate whether or not p traverses arc (i, j), $s^p_{i,j}$ be the amount of deicing material applied to (i, j) by p. We note that the $x^p_{i,j}$ variables are redundant, however their introduction allows some of the constraints to be written more concisely. In a sense, the x variables can be thought of as dictating the underlying graph of a Chinese Postman Problem, while the y variables are used to construct the routes on the graph defined by the x variables.

We note that the underlying graph is assumed to be a multigraph, allowing for multiple arcs between two nodes. This allows the model to account for multiple lanes on a given road segment by having multiple edges between the two nodes representing the end points of the road segment. However, it is important to note that this will induce multiple symmetries within the model that will have to be dealt with in order to solve instances of any practical size. One way to cut off such symmetries would be forcing one lane to be treated prior to the others. As an example, suppose e_1 and e_2 are edges between i and j with identical characteristics. To ensure e_1 is treated before e_2 is treated, for each vehicle $p \in \mathcal{P}$, the constraint $x^p_{e_1} \geq x^p_{e_2}$ is added. This ensures that e_2 is treated only if e_1 is also treated.

Since the goal is to allocate deicing resources in such a way as to maximize the number of people that can leave home, travel as they need, then return home, the objective of the MIP model is to maximize the second-stage flow through the network. Let f be the variable to measure the second-stage flow through the network. For each arc $(i, j) \in \mathcal{A}$, let $f_{i,j}$ be the first-stage flow across the arc, and $f'_{i,j}$ be the second-stage flow across the arc. Since these flows represent traffic flows that occur at different times of the day, they will interfere with one another minimally. Note that, for the dummy source s and a supply node v, the first-stage flow across (s, v) represents the number of people who are able to leave this source, while for the dummy sink t and a demand node u, the second-stage flow across arc (t, u) represents the number of poeple who are able to leave this demand node. For standard (multi-commodity) network flow problems, the sum of the flows across an arc cannot exceed the arc's capacity. However, since the flows in this model do not interfere with one another, we allow the sum of the flows to (perhaps) exceed the arc's capacity, so long as neither of the flows individually do. For example, if an arc (i, j) has a capacity of 5, then $f_{i,j} = f'_{i,j} = 5$ is a feasible flow. Figure 2.1 illustrates this.

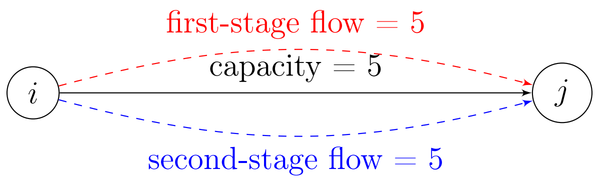


Figure 2.1: Feasible first- and second-stage flows across arc (i, j).

Many municipalities for which this approach would be useful primarily deal with snow and ice removal via deicing materials, as most snowfall events result in too little snow accumulation for plows to be useful. As a result, the model presented here does not include variables for snow plowing. The model is as follows:

$$\max f \tag{2.1}$$

subject to:

$$f \leq \sum_{i \in \mathcal{D}} f'_{t,i} \tag{2.2}$$

$$f_{i,j} \leq \gamma_{i,j} TC_{i,j} - \alpha_{i,j} RVI_{i,j} + \beta_{i,j} \sum_{p \in \mathcal{P}} s^p_{i,j} \qquad \forall (i,j) \in \mathcal{A} \tag{2.3}$$

$$f'_{i,j} \leq \gamma_{i,j} TC_{i,j}$$

$$- \alpha_{i,j} RVI_{i,j} + \beta_{i,j} \sum_{p \in \mathcal{P}} s^p_{i,j} \qquad \forall (i,j) \in \mathcal{A} \tag{2.4}$$

$$f_{i,j} \leq \gamma_{i,j} TC_{i,j} \qquad \forall (i,j) \in \mathcal{A} \tag{2.5}$$

$$f'_{i,j} \leq \gamma_{i,j} TC_{i,j} \qquad \forall (i,j) \in \mathcal{A} \tag{2.6}$$

$$\sum_{i|(i,j)\in\mathcal{A}} f_{i,j} = \sum_{i|(j,i)\in\mathcal{A}} f_{j,i} \qquad \forall j \in \mathcal{V} \tag{2.7}$$

$$\sum_{i|(i,j)\in\mathcal{A}} f'_{i,j} = \sum_{i|(j,i)\in\mathcal{A}} f'_{j,i} \qquad \forall j \in \mathcal{V} \tag{2.8}$$

$$f_{s,i} \leq S_i \qquad \forall i \in \mathcal{S} \tag{2.9}$$

$$f_{i,t} \leq D_i \qquad \forall i \in \mathcal{D} \tag{2.10}$$

$$f'_{i,s} = f_{s,i} \qquad \forall i \in \mathcal{S} \tag{2.11}$$

$$f'_{t,i} = f_{i,t} \qquad \forall i \in \mathcal{D} \tag{2.12}$$

$$\sum_{i\in\mathcal{S}\cup\{t\}} f_{s,i} = \sum_{i\in\mathcal{S}} S_i \tag{2.13}$$

$$\sum_{i\in\mathcal{D}\cup\{s\}} f'_{t,i} = \sum_{i\in\mathcal{D}} D_i \tag{2.14}$$

$$S^{min}_{i,j} x^p_{i,j} \leq s^p_{i,j} \qquad \forall (i,j) \in \mathcal{A}\ \forall p \in \mathcal{P} \tag{2.15}$$

$$\sum_{p\in\mathcal{P}} s^p_{i,j} \leq S^{max}_{i,j} \qquad \forall (i,j) \in \mathcal{A} \tag{2.16}$$

$$\sum_{p\in\mathcal{P}} \sum_{(i,j)\in\mathcal{A}} s^p_{i,j} \leq S^{total} \tag{2.17}$$

$$M x^p_{i,j} \geq y^p_{i,j} \qquad \forall (i,j) \in \mathcal{A}\ \forall p \in \mathcal{P} \tag{2.18}$$

$$x^p_{i,j} \leq y^p_{i,j} \qquad \forall (i,j) \in \mathcal{A}\ \forall p \in \mathcal{P} \tag{2.19}$$

$$\sum_{(i,j)\in\mathcal{A}} s^p_{i,j} \leq Q^p_S \qquad \forall p \in \mathcal{P} \tag{2.20}$$

$$\sum_{(i,j)\in\mathcal{A}} F^p_{i,j} y^p_{i,j} \leq Q^p_F \qquad \forall p \in \mathcal{P} \tag{2.21}$$

$$\sum_{p\in\mathcal{P}} \sum_{(i,j)\in A} F^p_{i,j} y^p_{i,j} \leq F^{total} \tag{2.22}$$

$$\sum_{p\in\mathcal{P}} \sum_{(i,j)\in A} T_{i,j} y^p_{i,j} \leq T^{total} \tag{2.23}$$

$$\sum_{i|(i,j)\in\mathcal{A}} y^p_{i,j} = \sum_{i|(j,i)\in\mathcal{A}} y^p_{j,i} \qquad \forall j \in \mathcal{V}\ \forall p \in \mathcal{P} \tag{2.24}$$

$$\sum_{j|(Dep,j)\in\mathcal{A}} y^p_{Dep,j} \geq 1 \qquad \forall p \in \mathcal{P} \tag{2.25}$$

$$\sum_{\substack{(i,j)\in\mathcal{A}\\ i\in C, j\notin C}} M x^p_{i,j} \qquad \forall C \subsetneq V \setminus \{Dep\}$$

$$\geq \sum_{\substack{(i',j')\in\mathcal{A}\\ i',j'\in C}} x^p_{i',j'} \qquad \forall p \in \mathcal{P} \tag{2.26}$$

$$x^p_{i,j} \in \{0,1\} \qquad \forall (i,j) \in \mathcal{A}\ \forall p \in \mathcal{P} \tag{2.27}$$

$$y^p_{i,j} \in \mathbb{Z}_+ \qquad \forall (i,j) \in \mathcal{A}\ \forall p \in \mathcal{P} \tag{2.28}$$

$$s^p_{i,j} \geq 0 \qquad \forall (i,j) \in \mathcal{A}\ \forall p \in \mathcal{P} \tag{2.29}$$

$$f_{i,j} \geq 0 \qquad \forall (i,j) \in \mathcal{A} \tag{2.30}$$

$$f'_{i,j} \geq 0 \qquad \forall (i,j) \in \mathcal{A} \tag{2.31}$$

$$f \geq 0 \tag{2.32}$$

Constraints ($2.2 - 2.14$) are the flow aspect of the model. ($2.1 - 2.2$) ensures that the objective function captures the total second-stage flow to the residences. ($2.3 - 2.4$) sets the capacity of each arc to depend on the traffic count, RVI, and deicing material on the arc, with ($2.5 - 2.6$) being trivial upper bounds on the flow across an arc. ($2.7 - 2.8$) are flow conservation constraints. (2.9) ensures first-stage flow from a residence cannot exceed the number of people at that residence, and (2.10) ensures first-stage flow to a destination cannot exceed the number of people who want to go to that destination. ($2.11-2.12$) ensures that the second-stage flow from a destination does is equal to the first-stage flow arriving at that destination, and similarly for residences. ($2.13 - 2.14$) ensure that the total first-stage flow (resp. second-stage flow) from the artificial source and sink nodes is equal to the sum of the sources (resp. sinks), with flows along the arcs (s,t) and (t,s) being un-routed flows.

Constraints ($2.15-2.32$) model the deicing, resource constraints, and vehicle routing. Due to the nature of some of the valves used to disperse deicing materials, there may be a non-zero minimum amount of deicing material that can be applied if the valve is open. Constraints (2.15) capture this. There is also a maximum amount of deicing material that can be laid on a road segment, (2.16) models this. (2.17) ensures the total deicing chemicals applied does not exceed the budget. (2.18 − 2.19) tie the x and y variables together. It should be noted that M in these constraints is a large constant, with $M = \max_{(i,j)\in\mathcal{A},p\in\mathcal{P}}\{Q_F^p/F_{i,j}^p\}$ being sufficient. (2.20 − 2.21) are the fuel and salt capacity constraints for each vehicle, and (2.22 − 2.23) are the fuel and time budget constraints. (2.24) ensures that if a vehicle goes to a location, it must leave that location, and vice versa, while (2.25) makes sure that every vehicle leaves the depot. (2.26) are similar to subtour elimination constraints in Traveling Salesman or Vehicle Routing instances, and we will refer to these constraints as *disconnected subtour elimination* constraints. The disconnected subtour elimination constraints ensure that, for any subset of nodes that does not include the depot, any vehicle route that includes a closed walk along these nodes must eventually leave these nodes. We note that these constraints are written in terms of the x variables, whereas the y variables ultimately contain the information needed to construct the individual routes. As mentioned previously, the x variables determine the underlying set of edges that will have vehicles traversing them, while the y variables contain the routes that will cover these edges. The disconnected subtour elimination constraints force the subgraph induced by these edges to be connected. This, together with (2.24), force each vehicle's route to be a single closed walk. Just as in the subtour elimination constraints in TSP and VR instances, we note that there are exponentially-many disconnected subtour elimination constraints. As before, M is a large constant, with $M = |\mathcal{A}|$ being large enough.

A practical consideration in any deicing plan is quickly treating certain routes, such as emergency routes or bridges. The MIP model presented does not account directly for these considerations, and, to some degree, this is intentional. The approach presented is attempting to use deicing resources so as to maximize the people who can use the road network, not necessarily to maximize the treatment (or promptness of treatment) of emergency routes. Additionally, since many municipalities have effective treatment for emergency routes already in place, this approach is meant to augment current decision-making processes rather than wholly replace them. Regardless, there are at least two simple constraints that can be added to force certain edges to be treated. To ensure only that a given arc (i, j) is treated by some vehicle, the constraint $\sum_{p\in\mathcal{P}} x_{i,j}^p \geq 1$ can be added. Often, critical road segments need to be fully treated. To force a certain minimum treatment level TL on road segment (i, j), the constraint $\sum_{p \in \mathcal{P}} s^p_{i,j} \geq TL$ can be added to the model. We note that the second approach is tighter than the first, as any solution that satisfies the second constraint also satisfies the first. Finally, the constructive heuristic presented later can easily be modified to account for these considerations, which we explain in Section 2.4.

2.4 Heuristic Approach

In this section, we will discuss the details of a constructive heuristic approach for this problem. The heuristic has two phases: first, to build a priority queue of arcs to be treated, and second to route the deicing vehicles so as to treat the arcs in the queue. In order to build the priority queue of arcs to be treated, we take inspiration from a similar problem known as the Capacity Expansion in a Flow Network Problem or the Parametric Budget Problem [35, 22]. We will first discuss the Capacity Expansion in a Flow Network Problem, as well as its solution approaches. We will then adapt this approach to building a priority queue of arcs to be treated in our heuristic approach.

2.4.1 Capacity Expansion in a Flow Network Problem

The Capacity Expansion in a Flow Network Problem, first described by Fulkerson in [22] (where it was called the Parametric Budget Problem), is a variant of the Maximum Network Flow Problem. In this variant, the objective remains the same (find a feasible flow from the source to the sink that is maximum), but the capacities of the arcs in the graph can be increased, subject to some budgeting constraints. Typically, the amount by which the capacity of a particular arc can increase is bounded, as is the total amount by which all capacities can be increased.

More formally, let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph. Let $s \in \mathcal{V}$ be the source node, and $t \in \mathcal{V}$ be the sink node for a network flow. Let f be the flow from s to t, with f_{uv} the flow along edge $uv \in \mathcal{E}$. Further, we let c_{uv} be the capacity of arc $uv \in \mathcal{E}$ and b_{uv} the "budget" expended to increase the capacity of arc $uv \in \mathcal{E}$. Also let a_{uv} be the weight that determines the effect of spending budget on arc $uv \in \mathcal{E}$ on the capacity of the arc. The Capacity Expansion in a Flow Network Problem is as follows:

$$\max f \tag{2.33}$$

subject to: (2.34)

$$\sum_{v|(v,u)\in\mathcal{E}} f_{vu} - \sum_{v|(u,v)\in\mathcal{E}} f_{uv} = 0 \qquad \forall u \in \mathcal{V}, u \neq s,t \tag{2.35}$$

$$f + \sum_{v|(v,s)\in\mathcal{E}} f_{vs} = \sum_{v|(s,v)\in\mathcal{E}} f_{sv} \tag{2.36}$$

$$\sum_{v|(v,t)\in\mathcal{E}} f_{vt} = \sum_{v|(t,v)\in\mathcal{E}} f_{tv} + f \tag{2.37}$$

$$f_{uv} \leq c_{uv} + a_{uv}b_{uv} \qquad \forall (u,v) \in \mathcal{E} \tag{2.38}$$

$$b_{uv} \leq b_{uv}^{max} \qquad \forall (u,v) \in \mathcal{E} \tag{2.39}$$

$$\sum_{(u,v)\in\mathcal{E}} b_{uv} \leq b^{max} \tag{2.40}$$

$$f_{uv} \geq 0 \qquad \forall (u,v) \in \mathcal{E} \tag{2.41}$$

$$b_{uv} \geq 0 \qquad \forall (u,v) \in \mathcal{E} \tag{2.42}$$

$$f \geq 0 \tag{2.43}$$

Since this problem is a continuous linear programming problem, it can be solved in polynomial time [27]. However, a graph theoretic primal-dual approach is known that will solve this problem. The basic idea is as follows. First, solve the flow problem, ignoring the expansion aspect of the problem. Then, as long as there is some budget left to increase the capacities, find the $s-t$ path with the lowest marginal cost to send additional flow. If this cost is finite, increase the capacities of the saturated arcs in this $s-t$ path until the marginal cost changes or the budget is exhausted. Repeat until the budget is exhausted, or no finite cost $s-t$ path exists [35].

Note that, given the marginal cost of sending additional flow along each arc within $\mathcal{G}$, the cheapest $s-t$ path can be determined via a shortest path approach. If an arc uv is not saturated (i.e. $f_{uv} < c_{uv}$), then it is not necessary to increase the capacity of the arc to send additional flow along the arc. In this case, the marginal cost of the arc is 0. If the arc uv is saturated but it is still possible to increase the capacity ($f_{uv} = c_{uv}$ and $b_{uv} < b_{uv}^{max}$), then the marginal cost of the arc is $1/a_{uv}$. Lastly, if the arc uv is saturated and the capacity cannot be expanded ($f_{uv} = c_{uv}$ and $b_{uv} = b_{uv}^{max}$), then no additional units of flow may be sent along this arc. This corresponds to a marginal cost of ∞. Equation 2.44 computes the marginal cost of sending additional flow across an arc, using the above observations. The algorithm to solve the Capacity Expansion in a Flow Network Problem is given in Algorithm 1. We note that Algorithms 1 and 2 are due to Fulkerson and are contained in [22].

$$\pi(uv) = \begin{cases} 0 & \text{if } f_{uv} < c_{uv} \\ \frac{1}{a_{uv}} & \text{if } f_{uv} = c_{uv} \text{ and } b_{uv} < b_{uv}^{max} \\ \infty & \text{if } f_{uv} = c_{uv} \text{ and } b_{uv} = b_{uv}^{max} \end{cases} \tag{2.44}$$

Notice that if we are only concerned with allocating deicing materials along the roadways, while ignoring the vehicle-routing aspects present, then Problem 2.1-2.32 is very similar to Problem 2.33-2.43.

2.4.2 Capacity Expansion Inspired Heuristic

As mentioned previously, the approach used for the heuristic can be broken into two main phases. Phase I consists of creating a priority queue of edges to treat, and phase II is routing the trucks.

In order to create the priority queue, we take inspiration from Algorithm 1. The main idea behind Algorithm 1 is to continue to find the $s-t$ path in $\mathcal{G}$ with the lowest marginal cost of sending additional flow. The deicing vehicle routing problem studied in this work can be thought of as a capacity expansion max flow variant, with a complicating underlying subproblem (vehicle routing in this case). Thus we can adapt Algorithm 1 to approximate the ideal set of arcs to treat while providing an ordering of the arcs to approximate the importance of treating said arc. We note that, even while ignoring the vehicle routing aspect of the problem, there are two significant differences between Problem (2.33-2.43) and Problem (2.1-2.32). Namely, the flow problem aspect of Problem (2.1-2.32) has multiple sources and sinks and multiple types of flows, and constraints 2.15 may force a semi-continuous nature on the deicing material variables.

Algorithm 1: Fulkerson's algorithm for solving the Capacity Expansion in a Network Flow problem

```
1  function CapacityExpansion ($G = (V, E), s, t, a, b, c, b^{max}$);
   Input  : A graph $G$, source vertex $s$, sink vertex $t$, a vector containing all $a_{uv}$
            values $a$, a vector containing all $b_{uv}$ values $b$, a vector containing all $c_{uv}$
            values $c$, a maximum budget vector $b^{max}$;
   Output: maximum flow value $F$;
2  initialization;
3  Solve Maximum Network Flow problem on $G$ with all expansion variables ($b_{uv}$) set
    to 0;
4  $G' \leftarrow (V, E)$ a weighted graph, where $w(uv) = \pi(uv)$; // used to compute path with
    lowest marginal cost
5  $budgetUsed \leftarrow 0$; //keeps track of total expansion budget used
6  $F \leftarrow$ maxFlow from 3; // maximum flow
7  while $budgetUsed < b^{max}$ do
8      $p \leftarrow$cheapest $s - t$ path in $G'$;
9      $eF \leftarrow extraFlow(p, b, c, f)$;
10     if $eF = 0$ then
11         break; // unable to send additional flow
12     $budgetRequired \leftarrow \sum_{e \in p} w(e)_{G'} eF$; // how much budget is required to send $eF$
        additional units of flow
13     if $budgetRequired > b^{max} - budgetUsed$ then
14         $eF \leftarrow eF \cdot \frac{b^{max} - budgetUsed}{budgetRequired}$; // adjust eF to be maximum amount possible by
            budget constraint
15     $f_{uv} \leftarrow f_{uv} + eF$ for all $uv \in p$;
16     $F \leftarrow F + eF$;
17     Update weights of edges in $G'$;
18     $budgetUsed \leftarrow budgetUsed + budgetRequired$;
19 end
20 return $F$;
```

Algorithm 2: Function to determine the maximum amount of additional flow that can be sent along an $s-t$ path at the current cost

```
1  function extraFlow ($p, b, c, f$);
   Input  : An $s-t$ path $p$, vectors $b, c, f$ from 2.33-2.43
   Output: Extra flow that can be sent along $p$ at current cost
2  initialization;
3  $eF \leftarrow \infty$; // initialize to something big
4  for edge $e$ in $p$ do
5      if $f_e < c_e$ then
6          $eF \leftarrow \min\{eF, f_e - c_e\}$;
7      else if $f_e = c_e$ and $b_e < b_e^{max}$ then
8          $eF \leftarrow \min\{eF, b_e^{max} - b_e\}$;
9      else
10         return 0; // $f_e = c_e$ and $b_e = b_e^{max}$, unable to send additional flow along path
11     end
12 end
13 return $eF$;
```

Often within network flow problems with multiple sources and sinks, an artificial master source node and an artificial master sink node are added to the problem. This allows for the model to then be treated as a single source/sink problem. The deicing vehicle routing problem we are studying has multiple flows in addition to multiple sources and sinks. While master source and sink nodes can be added to make the MIP easier to formulate, considering this modified graph will not work for Algorithm 1. To address the fact that multiple sources and sinks are present, we instead propose computing multiple shortest paths, and taking the shortest of these. For each source $s_i \in \mathcal{S}$ and each sink $t_j \in \mathcal{D}$, we need the $s_i - t_j$ path with the lowest marginal cost of sending additional first-stage flow along it as well as the $t_j - s_i$ path with the lowest marginal cost of sending additional second-stage flow along it. Note that the marginal cost of the $t_j - s_i$ path may depend on the optimal $s_i - t_j$ path and vice versa. Since these shortest paths are being computed so that their capacities can be expanded, it is possible that the shortest $s_i - t_j$ path and the shortest $t_j - s_i$ path share an edge. If an edge has a non-zero cost in the $s_i - t_j$ path, then it should have a zero cost in the $t_j - s_i$ path, as its appearance in the $s_i - t_j$ path means it has been chosen to have its capacity expanded. To see this, consider the graph in Figure 2.2, where s is the source for the first-stage flow (and the sink to the second-stage flow), and t is the sink for the first-stage flow (source for the second-stage flow). Note that the only $s - t$ path is not disjoint from the only $t - s$ path. Thus, if the capacity of arc ab is increased to allow more first-stage flow across the arc, it is not necessary to increase this arc's capacity for the second-stage flow.

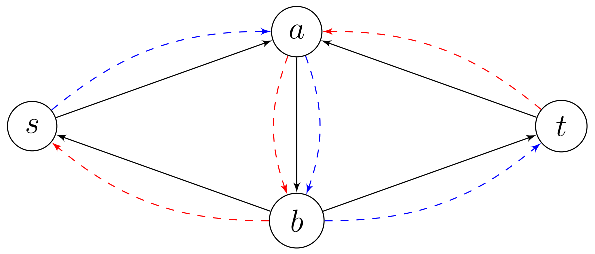


Figure 2.2: $s - t$ and $t - s$ paths may not be disjoint.

To account for this, instead of a single dual graph being maintained, as in Algorithm 1, two dual graphs are maintained. One is to compute the shortest paths with respect to the first-stage flows, and the other is to compute the shortest paths with respect to the second-stage flows. Additionally, since the objective function of Problem (2.1-2.32) can be thought of as the amount of flow that can leave its source, find a sink, leave said sink, and return to its source, it is necessary to compute all $s_i - t_j$ shortest paths in the first dual graph, then use those paths to update the weights in the second dual graph and compute the $t_j - s_i$ shortest paths in the second dual graph. We note that this approach ultimately suffers from the same problems that graph theoretic multi-commodity flow algorithms suffer from. While these problems can be solved in polynomial time via linear programming, there is no known polynomial time algorithm that does not use linear programming.

The complication due to constraints 2.15 are addressed in a simple manner. We note that, for an arc ij, after $S_{i,j}^{min}$ deicing material has been applied to it, then the marginal cost to send additional flow across that arc no longer needs to consider the semi-continuous nature of the $s_{i,j}^{p}$ variables. Thus the marginal cost of an arc is approximately given by Equation 2.44. However, prior to that, we need to account for the fact that there is a minimum amount of deicing material that must be applied to the arc. Since our approach is only approximating the path with the lowest marginal cost currently, we modify Equation 2.44 as follows:

$$\pi'(uv) = \begin{cases} 0 & \text{if } f_{uv} < c_{uv} \\ \frac{1}{a_{uv}} & \text{if } f_{uv} = c_{uv},\ b_{uv} < b_{uv}^{max} \\ & \text{ and } \sum_{p \in \mathcal{P}} s_{uv}^{p} > 0 \\ \frac{1}{a_{uv}} + S_{uv}^{min} & \text{if } f_{uv} = c_{uv},\ b_{uv} < b_{uv}^{max} \\ & \text{ and } \sum_{p \in \mathcal{P}} s_{uv}^{p} = 0 \\ \infty & \text{if } f_{uv} = c_{uv},\ b_{uv} = b_{uv}^{max} \end{cases} \tag{2.45}$$

At each iteration, once the source-sink pair with the lowest estimated marginal cost has been identified, the approach used to determine the expansion of the edges in the relevant paths is identical to that of Algorithm 1. That is, simply find the edge in the relevant paths whose cost will change with the least amount of additional flow. The increase in flow required to increase the cost of this edge is the amount of additional flow that can be sent along these paths at the current cost.

In order to produce the priority queue mentioned, we note that at each iteration, the previously discussed approach searches for the cheapest bottleneck *at that iteration*. Thus, the first set of arcs that should be treated is the set of arcs that are at capacity (and can have their capacities increased) in the paths identified in the first iteration. The second set of arcs that should be treated is the set of arcs that are at capacity in the second iteration. And so forth. This is the basis of Algorithm 3.

After the queue is built, phase II of the heuristic begins, detailed in Algorithm 4. This phase is where the deicing vehicles are routed. While vehicle routing problems can often be difficult to solve, especially on large instances, we note that heuristics will likely be sufficient for this problem. Deicing vehicles hold enough deicing materials to treat approximately 70 lane-miles, while holding enough fuel to travel approximately 300 miles. Put simply, the trucks will often run out of salt before they run out of fuel. As a result, it is often unnecessary to route the trucks optimally with respect to distance traveled.

Since a priority queue is present, the idea behind the vehicle routing heuristic used is to build up the routes with a simple insertion rule as arcs from the queue are processed, similar in spirit to the Clarke and Wright savings heuristic [12]. We assume that the beginning and ending location is known for each truck. We initialize the route of each deicing vehicle as the shortest path from its beginning location to its ending location. For each arc in the priority queue, the truck used to treat this arc is determined by, of the trucks that have enough deicing material remaining to treat the arc, which truck has to travel the smallest marginal distance to treat this arc. The distance required to treat a new arc is determined by a simple insertion heuristic. Considering the route p as a series of vertices $p = v_0, v_1, ..., v_{n-1}$, the cost to treat an arc ij, denoted $\nu(ij)$, is defined as

$$\nu(ij) := \ell(ij) + \min_{1<=k<=n-1} \{sp(v_{k-1}, i) + sp(j, v_k) - \ell(v_{k-1}, v_k)\} \tag{2.46}$$

where $sp(u, v)$ is the length of the shortest path between u and v, and $\ell(u, v)$ is the length of the arc (u, v). This is simply the increase in the route length by adding ij to the route while keeping the order of the other arcs in the route the same. Note that this "cost" is in terms of the length added to the route of the truck. While it is expected that the trucks will run out of salt before they run out of fuel, some care needs to be given to ensure the

Algorithm 3: Algorithm to build priority queue of arcs to treat

```
1  function BuildQueue $\mathbb{P}$;
   Input  : A deicing vehicle routing problem instance $\mathbb{P}$;
   Output: priority queue of arcs to treat $Q$;
2  initialization;
3  Solve $\mathbb{P}$ with all $x, y, s$ variables set to 0;
4  $Q \leftarrow [\ ]$; // priority queue of arcs to treat
5  $budgetUsed \leftarrow 0$; //keeps track of total expansion (deicing material) budget used
6  while $budgetUsed < b^{max}$ do
7      $currentBestPathCost \leftarrow \infty$; // initialize, cost of shortest paths for current best
         source/sink pair
8      $currentBestSource \leftarrow -1$; // initialize, current best source
9      $currentBestSink \leftarrow -1$; // initialize, current best sink
10     for $s_i$ a source node, $t_j$ a sink node do
11         $G'_{to} \leftarrow (V, E)$ a weighted graph, where $w(e) = \pi'(e)$;
12         $p_{to} \leftarrow$shortest $s_i - t_j$ path in $G'_{to}$;
13         $G'_{from} \leftarrow (V, E)$ a weighted graph, where $w(e) = 0$ if $e \in p_{to}$ and $w(e) > 0$ in
             $G'_{to}$, $w(e) = \pi'(e)$ otherwise;
14         $p_{from} \leftarrow$shortest $t_j - s_i$ path in $G'_{from}$;
15         if $cost(p_{to}) + cost(p_{from}) < currentBestPathCost$ then
16             // found a better source/sink pair, update
17             $currentBestPathCost \leftarrow cost(p_{to}) + cost(p_{from})$;
18             $currentBestSource \leftarrow s_i$;
19             $currentBestSink \leftarrow t_j$;
20     end
21     $eF \leftarrow extraFlow(p_{to}, s, TC, f)$; // find most restrictive arc in $G'_{to}$
22     $eF \leftarrow \min\{eF, extraFlow(p_{from}, s, TC, f')\}$; // find most restrictive arc in $G'_{from}$
         and compare to previous most restrictive arc
23     if $eF = 0$ then
24         break; // unable to send additional flow
25     compute $actualBudgetRequired$; // how much budget is required to send $eF$
         additional units of flow
26     if $actualBudgetRequired > b^{max} - budgetUsed$ then
27         $eF \leftarrow eF \cdot \frac{b^{max} - budgetUsed}{actualBudgetRequired}$; // adjust eF to be maximum amount possible
             by budget constraint
28     $f_{uv} \leftarrow f_{uv} + eF$ for all $uv \in p_{to}$; $f'_{uv} \leftarrow f'_{uv} + eF$ for all $uv \in p_{from}$;
29     $F \leftarrow F + eF$;
30     Push arcs to treat, and how much treatment they receive, to $Q$;
31     $budgetUsed \leftarrow budgetUsed + budgetRequired$;
32 end
33 return $Q$;
```

Algorithm 4: Algorithm to route vehicles to treat arcs

```
1  function RouteTrucks $\mathbb{P}, Q$;
   Input  : A deicing vehicle routing problem instance $P$ and arc priority queue $Q$;
   Output: A feasible truck route $R$;
2  initialization;
3  for $vehicle \in \mathcal{P}$ do
4      $R_{vehicle} \leftarrow$ shortest path from $vehicle$ start depot to $vehicle$ end depot; // route
        of $vehicle$
5      $saltUsed_{vehicle} \leftarrow 0$; // amount of salt used by $vehicle$
6  end
7  for $e \in Q$ do
8      for $vehicle \in \mathcal{P}$ if vehicle has enough salt to treat $e$ do
9          $vehicleToTreat \leftarrow \arg\min\{\nu(e)\}$; // vehicle that can treat $e$ at cheapest cost
10         if $vehicleToTreat$ is able to treat $e$ then
11             update $R_{vehicleToTreat}$ and $saltUsed_{vehicleToTreat}$;
12         else
13             return R; // if the most-capable vehicle is unable to treat the arc, no
                 vehicle is capable
14         end
15     end
16 end
17 return R;
```

truck routes remain feasible with respect to fuel usage time allowance. Further, since it is easy to determine if a given truck has enough salt remaining to treat an arc, this approach provides a nice compromise between approximating the difficult aspect of the problem while still capturing the important information.

As mentioned previously, critical routes are an important consideration in any winter road maintenance scheme. While the MIP model presented is not meant to incorporate these, the heuristic approach is meant to be an in-between strategy, incorporating ideas from both the current approach that many municipalities use (priority queues) and the MIP model (RVI values and overall network flow). As such, the heuristic can be easily adapted to account for critical routes and links. Since these arcs represent high-priority road segments, simply add them to the beginning of the priority queue at the start of phase I of the heuristic. This will ensure these road segments receive the treatment needed. Further, as treatment to these segments is often time-sensitive, phase II can be modified so that, after these critical arcs are treated, no less-critical arcs may be inserted into the vehicles' routes before these arcs. This forces these arcs to be treated first.

2.5 Computational Experiments

2.5.1 Experimental Procedure

All coding was done in C++ with extensive use of the Boost C++ Libraries [9]. The integer programming solver used was Gurobi version 7.5.1. All tests were done on a 16 core (32 thread) workstation with 2 Intel Xeon E5-2670 processors (2.6 GHz), 256 GB RAM, running Ubuntu 14.04.5. Due to the large number of disconnected subtour elimination constraints, these constraints were enforced in a lazy fashion by checking candidate solutions for subtours that do not include the depot. If such subtours were found, a violated disconnected subtour elimination constraint was added as a lazy constraint. Experiments were run using default parameters. Each instance was given a time limit of 30 minutes.

Two sets of instances were created, one using random graphs and one using a combination of actual and simulated data. In the random set, 3 graphs were created, with 200, 300, and

400 nodes. Each node was randomly assigned as a residence (10% chance), a destination (10% chance), or an intersection (80% chance). One intersection node was randomly chosen as the fuel/salt depot for each graph. In the data-driven graphs, 3 locations within Knox County, TN were selected at random, while a fourth was created using data from downtown Knoxville. For each of the 7 graphs, 11 different deicing material budget levels were considered. For all the graphs except the downtown Knoxville graph, the levels corresponding to a total deicing material budget of between 5% and 25% of what would be necessary to treat the network fully. For the downtown Knoxville graph, significantly less deicing material was necessary, and so these budget levels were between 0.5% and 1.5%.

In the random instances, the arc sets of the graphs were generated in two phases. The first phase generated the subgraph on the intersection nodes. This was done in an Erdős-Rényi fashion [40] so that the average in- and out-degree of a node was 3. If the resulting graph was not strongly connected, the arcs were removed and another arc set was generated. This was repeated until the resulting graph was strongly connected. The second phase added the remaining arcs. This was done with a modified Barabási-Albert model [4] so that the average degree of the residences and destinations was 2.

As for the parameter values in the random graphs, RVI values were assigned randomly from a uniform distribution, while arc lengths were assigned randomly from an exponential distribution. Fuel requirements, traversal times, and salt requirements for each road segment were assumed to be proportional to the length of the segment. Additionally, residence and destination data were selected randomly from an exponential distribution.

For the data-driven graphs, 4 locations within Knox County, TN were utilized to generate these graphs, with 3 being chosen at random and the last corresponding to downtown Knoxville. Lengths of each road segment is known for Knoxville, TN, so this data was not generated randomly. Again, traversal times, fuel requirements, and salt requirements for each road segment was assumed to be proportional to the length of the segment. Further, it is known whether each node is an intersection, residence, or destination. While RVI values are meant to take weather forecasts into account, assuming uniform snow coverage, RVI values for Knoxville, TN are known, and these values were used.

In both sets, because the flows are meant to represent the flow of traffic, the parameter $\gamma_{i,j}$ was set to 1 for all arcs. We note that the RVI values are between 0 and 6, inclusive, with 0 representing road segments that receive full sunlight and have little to no pitch, with 6 representing road segments that receive little sunlight and have a grade of at least 10%. The $\alpha_{i,j}$ parameter of each road segment was scaled such that an RVI value of 0 corresponds to a flow capacity of 100% of the clear-weather traffic count with no treatment, and an RVI value of 6 corresponds to a flow capacity of 20% of the clear-weather traffic count with no treatment. That is, $\alpha_{i,j} = \frac{2}{15}TC_{i,j}$. The $\beta_{i,j}$ parameter was scaled so that each lane requires 200 lbs. of deicing material per mile to be treated fully [17], so $\beta_{i,j} = \frac{TC_{i,j}}{200\ell(i,j)}$, where $\ell(i,j)$ is the length of arc (i,j).

Additionally, since many cities consider only traffic counts to determine priorities for the roads to treat, we compare our solutions with solutions obtained by treating the arcs with the highest traffic counts first.

2.5.2 Results

Figures 2.3-2.5 report the computational results for the random instances, while Figres 2.6-2.8 report the results for the Knoxville instances. n is the number of nodes in the graph, m the number of arcs in the graph, and treatment level is the total salt budget for the instance, given as a proportion of the total amount of salt required to treat all roads fully. *MIP* corresponds to the solution obtained by using Gurobi as the solver, *Heuristic* reports the data for the heuristic developed in Section 2.4, and *Current Approach* is the results of ranking roads according to traffic counts only. While the number of arcs treated is not part of the objective function of the model, a significant motivation behind this work is to utilize de-icing resources so as to be able to treat more roads. As a result we also report the number of arcs that were treated. Since the MIP approach was given a time limit of 1800 seconds, in the event of a timeout, we only report the data from the best-found solution.

The quality of the solutions found by the currently used approach were significantly surpassed by the solutions obtained from the MIP formulation and the heuristic approach, both in terms of the objective function and the number of arcs treated. In the random instances, the MIP approach sees an average flow improvement of 68.9% over the current

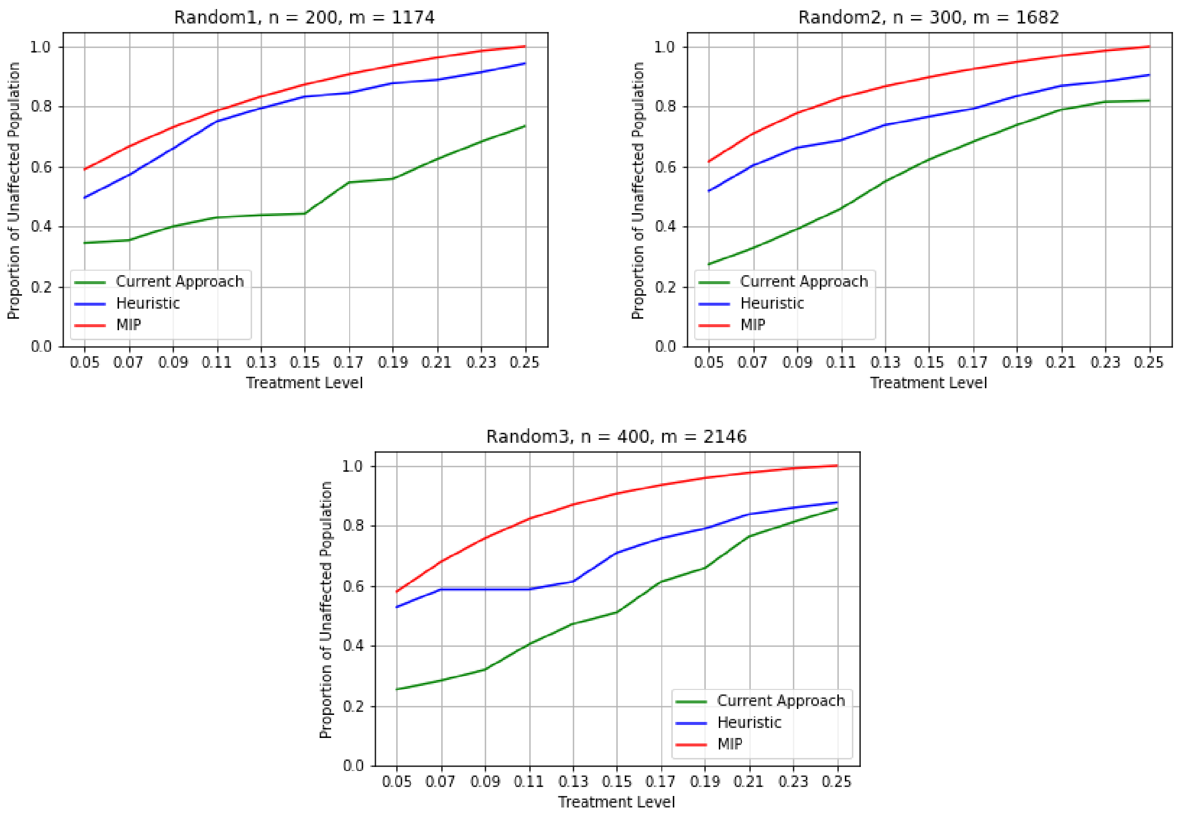


Figure 2.3: Proportion of population unaffected by snow event under different treatment techniques and levels of salt budget, random instances.

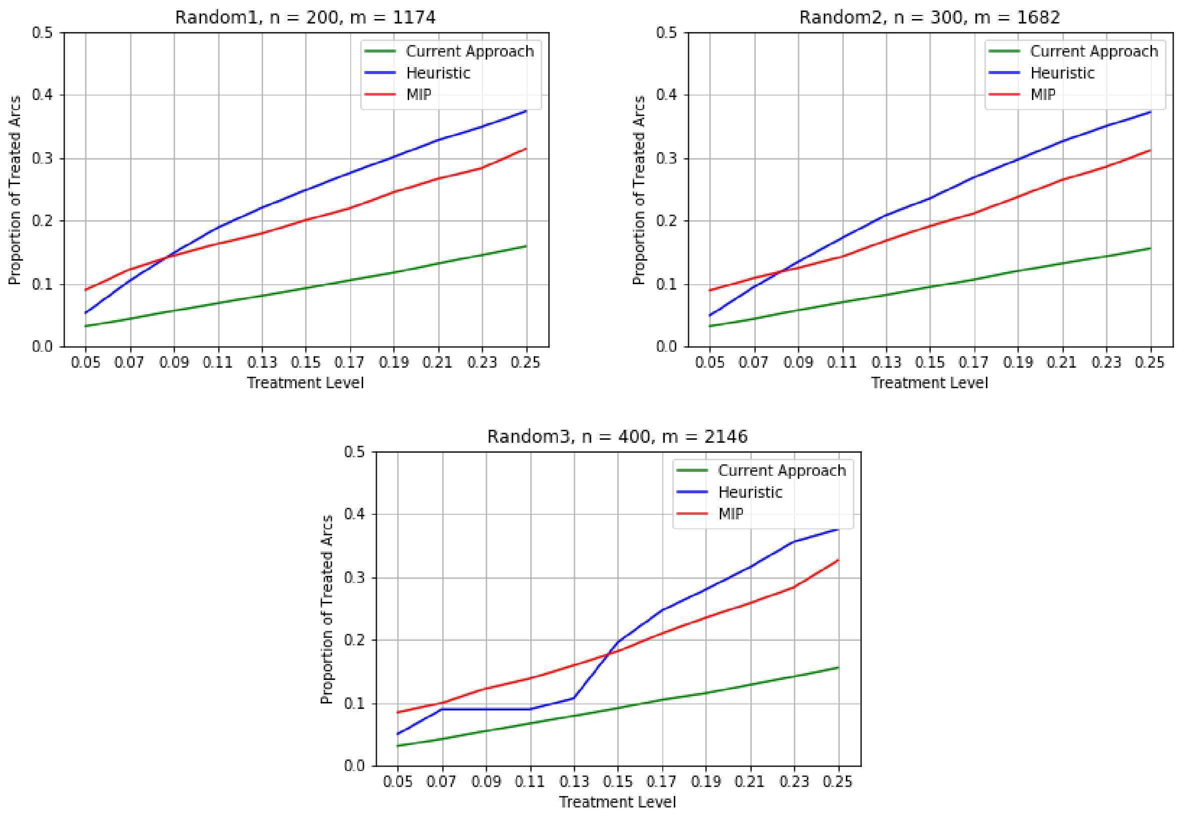


Figure 2.4: Proportion of arcs in network treated under different treatment techniques and levels of salt budget, random instances.

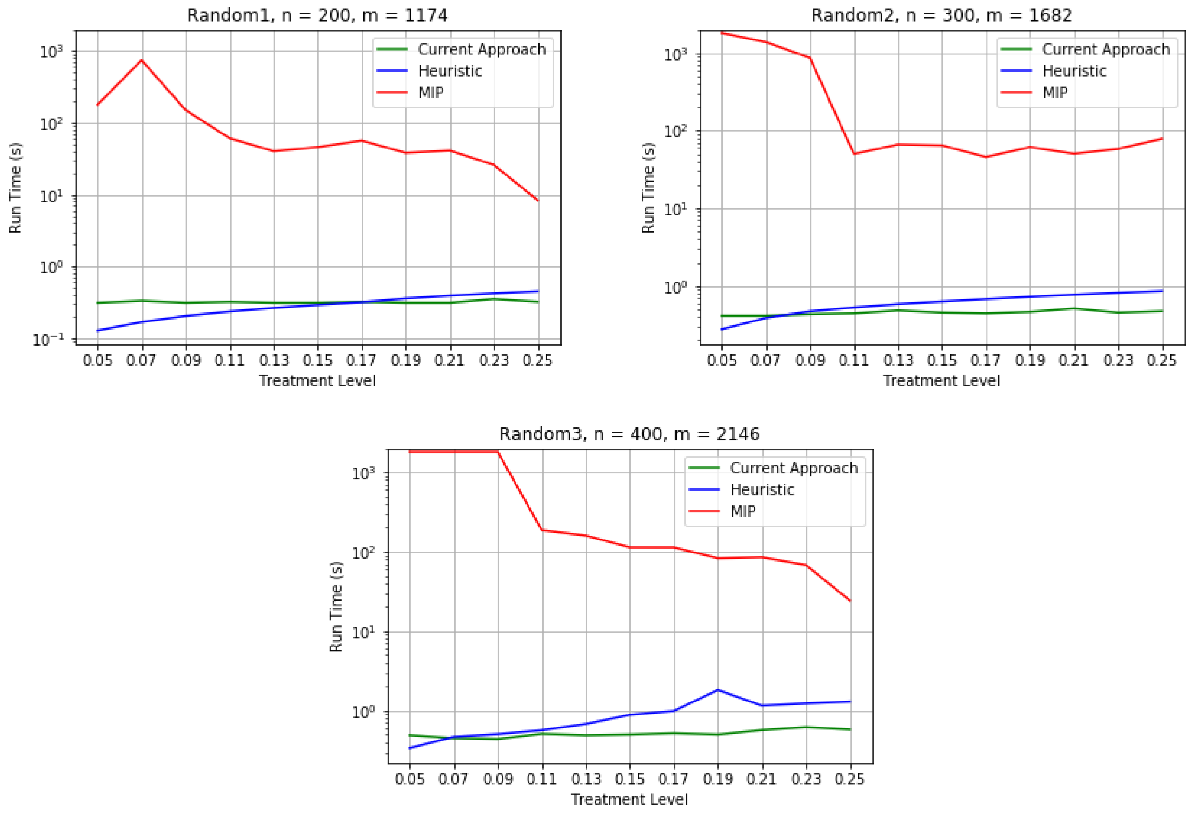


Figure 2.5: Run time of different treatment techniques at various levels, random instances.

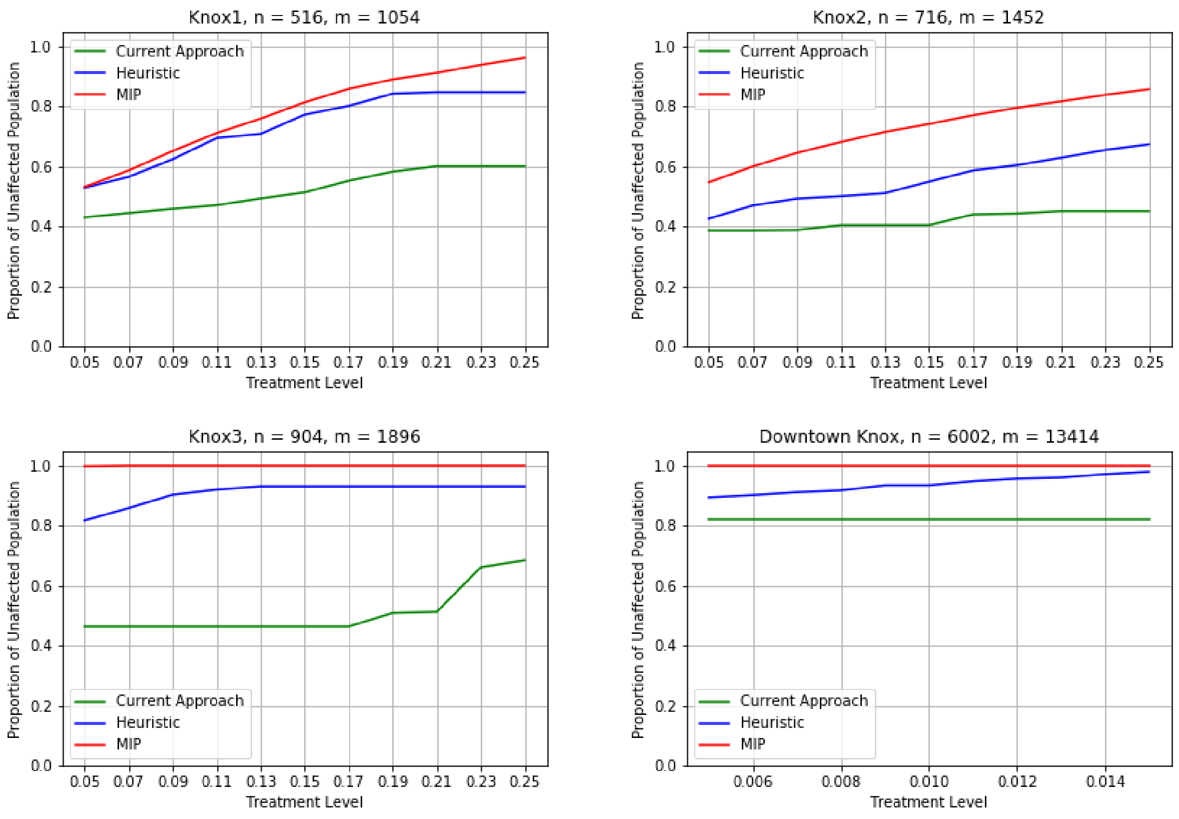


Figure 2.6: Proportion of population unaffected by snow event under different treatment techniques and levels of salt budget, Knoxville instances.

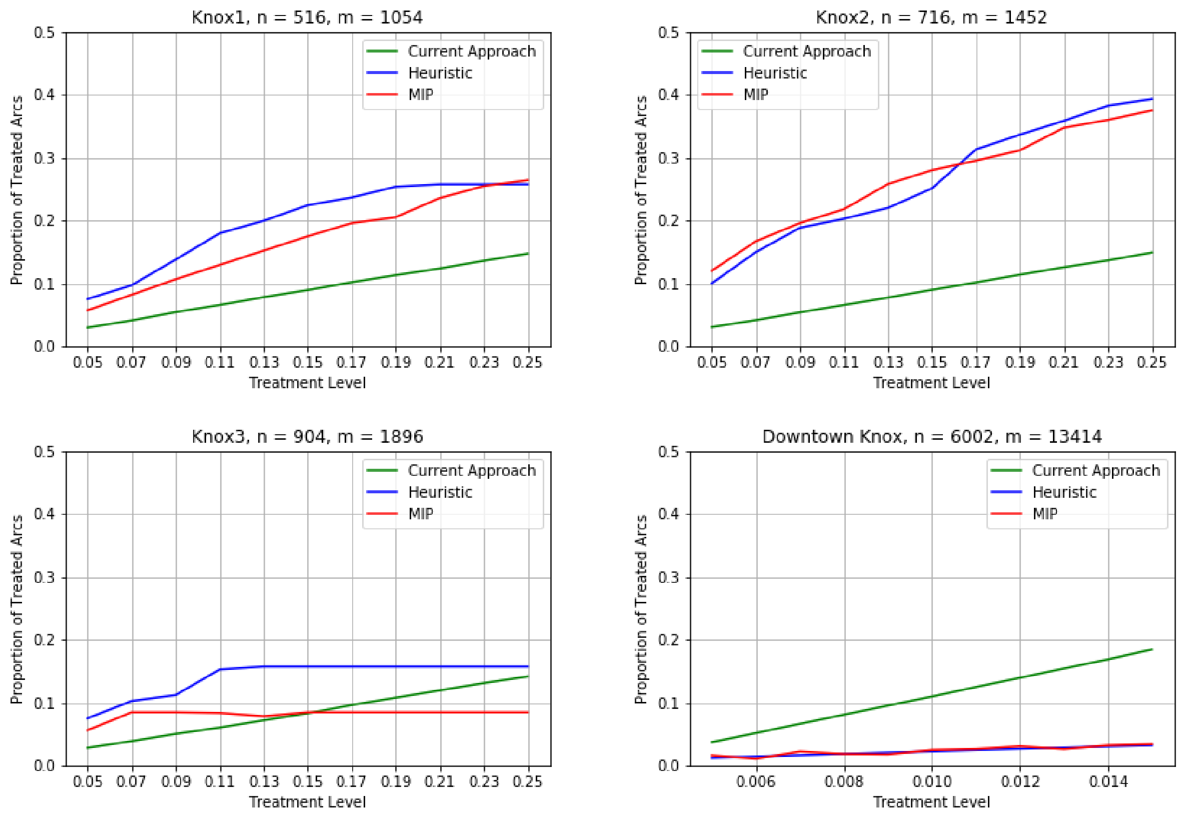


Figure 2.7: Proportion of arcs in network treated under different treatment techniques and levels of salt budget, Knoxville instances.

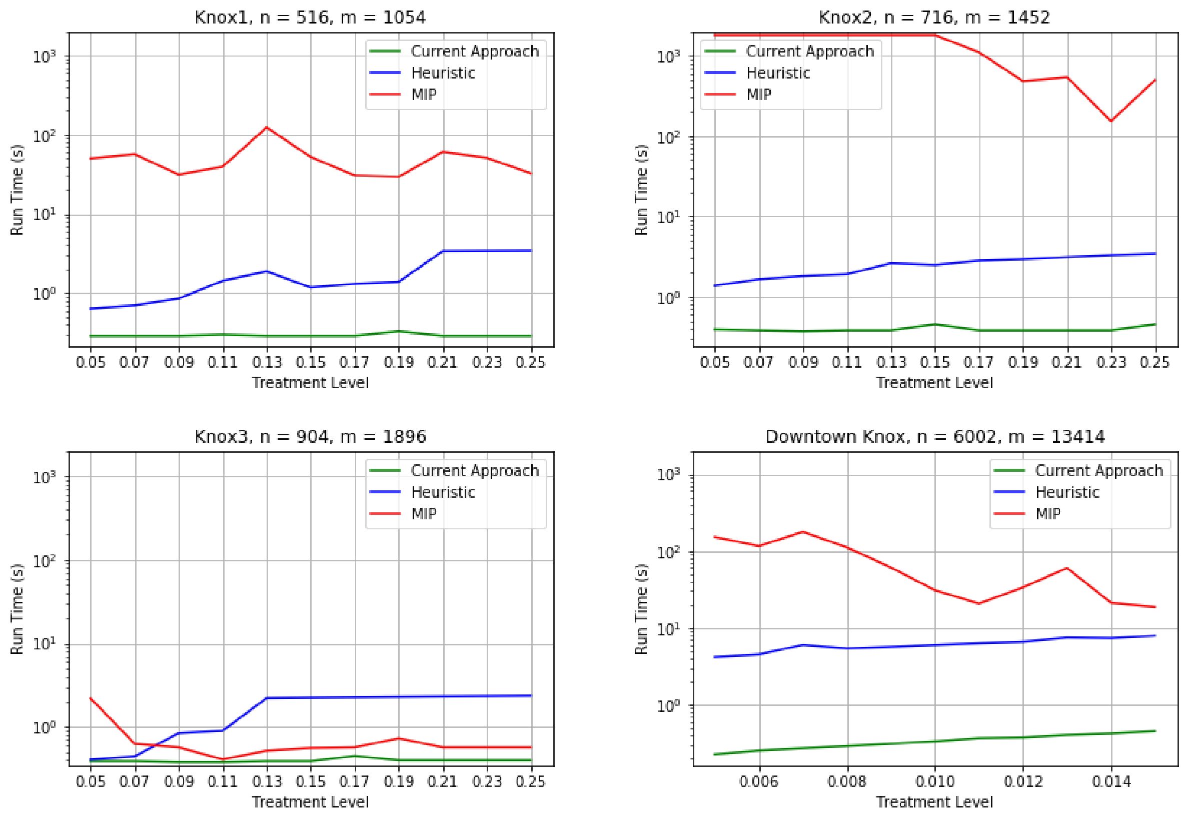


Figure 2.8: Run time of different treatment techniques at various levels, Knoxville instances.

approach, while the heuristic approach sees an average improvement of 45.9% over the current approach. Additionally, in the Knoxville instances, the MIP approach provides an objective function improvement of 60.9% over the current approach, while the heuristic approach performs 41.9% better than the current approach.

We note that the instances became computationally easier for the MIP solver as the budget increased, while the opposite trend was exhibited for the heuristic approach. While not reported directly, the MIP approach reached the time limit of 1800 seconds in 4 of the 33 random instances, and in 6 of the 44 Knoxville instances. Interestingly, all 6 of the timeouts were on the same instance, Knox2. Additionally, the heuristic approach never took longer than 10 seconds in any of the 77 instances, having an average run time of 3.01 seconds on the Knoxville instances, and 0.60 seconds on the random instances.

2.6 Conclusion and Future Work

Traffic counts alone are not enough when planning winter road maintenance strategies. By utilizing traffic patterns, meteorological data, and road network topology and topography, better maintenance strategies can be developed. By decreasing over-treatment of non-critical road segments, we can develop strategies which not only reduce the population affected by snow events, but also decrease deicing material use in general. In addition to the obvious savings enjoyed by having more efficient strategies, using our resources more wisely also reduces many of the non-primary costs associated with winter road maintenance, such as road damage and pollution.

Lastly, the current approach in many municipalities always treats the same roads. The model proposed here can suffer from the same limitation without high-quality weather forecasts. While certainly some people within any area won't mind this, many find themselves living on roads that never get treated. With this in mind, some natural questions are:

1. How many optimal/near-optimal solutions are there, and how different are these solutions?

2. If the problem were to find a set of solutions such that each road is treated in at least one of these solutions, how far from optimal would the worst solution be?

3. Is there a set of near-optimal solutions that will treat most roads with roughly equal probability? How much in terms of optimality would have to be sacrificed to find such a set?

4. Many large urban areas will have residence nodes interspersed with commercial locations, which can interfere with a two-stage flow model's results. However, this can be overcome by introducing multiple first and second-stage flows, but these problems can quickly become computationally taxing. What is the fewest number of flows that can be used to address this issue?

Chapter 3

Using Optimization To Budget For Winter Road Maintenance Activities

In the previous chapter, an optimization model was presented which seeks to maximize the utility of a deicing allocation scheme under a given budget. In this chapter, we seek to address the inverse question: that is, given a certain minimum acceptable utility level, what is the deicing budget that is necessary in order to attain that level? We show that the techniques developed in the previous chapter can also be applied to this setting. We provide computational experiments which validate these approaches.

3.1 Introduction

Many local governments across the United States of America face significant costs related to winter storm road maintenance. A significant portion of these costs are due to ice and snow removal from roadways through the use of plows and deicing agents, which totals \$1.5 billion annually in the US [17]. Further, improperly managed roadways also have several indirect costs associated with them. For example, untreated roadways are more dangerous for drivers, which can lead to increases in traffic congestion and traffic-related fatalities. Because of this, many local governments will temporarily close certain offices and services during winter snow events, which can further impact the social, educational, and economic well-being of local citizens and businesses. As a result, it is important that local municipalities adopt effective winter road maintenance strategies and budget for these events appropriately.

Because of the potential for savings in these areas, optimizing the various decisions related to winter road maintenance strategies is a well-studied family of problems. A thorough series of survey papers was written by Perrier et al. in 2006-2007 [41, 42, 43, 44] wherein many of the aspects of winter road maintenance were discussed. In the past, authors have primarily focused on system designs for spreading deicing materials and removing snow from roadways, approaches to dispose of snow removed from roadways, where to locate salt and fuel depots, fleet sizing, and optimizing the routes the individual deicing vehicles take when performing maintenance. Many of these approaches either directly or indirectly utilize some variant of the Snow Plow Routing Problem (SPRP), which is itself a generalization of the Chinese Postman Problem [36] and the Capacitated Arc Routing with Intermediate Facilities problem [23]. The primary focus of these models deals with routing the individual road maintenance vehicles, with the objective often being to minimize some metric that captures costs or inefficiencies in the management system.

In almost all of the previous work on routing individual vehicles, an implicit assumption within the models presented is that enough resources are present to treat the road network under consideration. While this assumption is reasonable in many of the locations where these problems have been studied in the past, such as Boston, MA and Pittsburgh, PA, it is simply untrue in other locations, such as Knoxville, TN, Atlanta, GA, and Raleigh, NC, to name a few. Because this assumption is violated in many locations, several local municipalities are unable to leverage large portions of the work that has been done towards these problems when designing their winter road maintenance strategies. Due to the fact that several locations either lack the fleet size, deicing material budget, or both, necessary to treat their entire road network during a snow event, these municipalities must determine which roads will be treated and which will not, in addition to the previously mentioned considerations. To our knowledge, no works addressing limited resource variations of this type have been considered until recently.

Due to this gap in the literature, in a previous work [2] we developed a results-oriented approach to allocating limited deicing materials during winter snow events that incorporates meteorological, environmental, population, and traffic data to make decisions about which roads to treat and the degree to which these roads should be treated. Our approach can be summarized as minimizing the number of citizens negatively affected by the snow event, subject to deicing material, fuel, and vehicle-hour budget limitations, as well as individual route feasibility. This is done by identifying bottlenecks in the road network due to snow by modeling traffic as a two-stage flow through a network in which the capacities of arcs can be increased by applying deicing materials along the corresponding roadways.

Notably, however, there is still a gap we were unable to find addressed in previous works. Namely, given a certain road maintenance strategy, how much money should a particular municipality expect to spend on deicing materials, fuel, and person-hours? While additional funds can be allocated throughout the season if needed, this often incurs additional costs that could have been avoided if appropriate plans had been made initially. Because substantial winter snow events occur with a high frequency in locations such as Boston, MA, which receives an average of 44 inches of snow per year, these local governments ensure enough resources are available for each snow event to treat their road network promptly. However, many places are more similar to Knoxville, TN, where the typical winter sees 7 snow events, with an average accumulation of slightly less than one inch per event. Because snow events tend to be mild, it is often not in the citizens of Knoxville's best interests to ensure every road is treated in each snow event, as increasing winter road maintenance budgets necessitates lowering other budgets. Thus there are two opposing objectives that must be taken into account. On one hand, it is important that enough resources are present to provide some minimum acceptable level of service for these winter snow events. But on the other, increases in winter road maintenance budgets often necessitate lowering other funds and services. In short, winter road maintenance budgets that are too low can lead to undesirable outcomes when snow events occur, while budgets that are too high can decrease the utility gained from those funds by limiting other services.

Our goal in this manuscript is to extend the work in [2] to fill this gap in the literature, allowing for estimates of the amount of deicing material required to meet a given level of service in a winter snow event. This paper lays the groundwork by defining the problem and introducing a mixed integer programming (MIP) formulation of the problem. The remainder of the paper is structured as follows. Section 3.2 contains the definition of the problem and the mathematical formulation. Section 3.3 outlines the experimental procedure and results, while 3.4 draws some conclusions and discusses future work.

3.2 Deicing Budgeting Problem

A natural way to address budget allocation problems is through stochastic programming techniques [26]. Stochastic programming, or optimization under uncertainty, is a modeling paradigm and framework that allows optimization problems to incorporate uncertainties present within the various quantities and parameters of the underlying situation represented by the model. Because of its broad applicability and usefulness, some examples of budgeting problems that have been studied with stochastic programming techniques are portfolio allocation [31], reliability in flow networks [34], and highway project selection [1]. However, in order to use stochastic programming techniques to optimize decisions, it is necessary to be able to solve the deterministic instance of a given problem. For this reason, we will focus our efforts here on determining the deicing budget necessary to attain a certain pre-determined level of service on a given road network in a single realization of a snow event. We call this the *Deicing Budgeting Problem*.

More formally, let $\mathcal{Q}(\xi, Z)$ be the utility function for a deicing budget level Z under a snow event ξ. The Deicing Budgeting Problem is to minimize the budget level Z subject to some minimum acceptable utility score ω. The simple mathematical programming formulation (3.1) captures this.

$$\min Z \tag{3.1}$$

subject to:

$$\mathcal{Q}(\xi, Z) \geq \omega \tag{3.2}$$

While the approach discussed here can be used with a general utility function $\mathcal{Q}$, the one we will consider in this work is the function in [2]. For the sake of brevity, below we provide a complete list of the variables and parameters of our utility function $\mathcal{Q}$. We offer the MIP formulation in (3.3) and provide a limited description of the model. For a more thorough explanation, the reader is referred to [2].

$\mathcal{G} = (\mathcal{V}, \mathcal{A})$ Directed multigraph of the road network.

$\mathcal{S} \subset \mathcal{V}$ Subset of vertices which represent to-flow sources/from-flow sinks.

$\mathcal{D} \subset \mathcal{V}$ Subset of vertices which represent to-flow sinks/from-flow sources.

$\mathcal{P}$ Set of plows.

S_i **for** $i \in \mathcal{S}$ The to-flow supply/from-flow demand at vertex $i \in \mathcal{S}$.

D_i **for** $i \in \mathcal{D}$ The to-flow demand/from-flow supply at vertex $i \in \mathcal{D}$.

$S_{i,j}^{max}$ **for** $(i,j) \in \mathcal{A}$ Maximum amount of deicing material allowed for road segment (i,j).

$S_{i,j}^{min}$ **for** $(i,j) \in \mathcal{A}$ Minimum amount of deicing material possible for road segment (i,j).

Z Total amount of deicing material available.

$T_{i,j}$ **for** $(i,j) \in \mathcal{A}$ Amount of time it takes a vehicle to traverse road segment (i,j).

T^{total} Total amount of plow time available.

$F_{i,j}^{v}$ **for** $(i,j) \in \mathcal{A}, v \in \mathcal{P}$ Amount of fuel required for vehicle v to traverse road segment (i,j).

F^{total} Total amount of fuel available.

Q_F^v **for** $v \in \mathcal{P}$ Fuel capacity for vehicle v.

Q_S^v **for** $v \in \mathcal{P}$ Deicing material capacity for vehicle v.

$TC_{i,j}$ **for** $(i,j) \in \mathcal{A}$ Traffic count for road segment (i,j).

$RVI_{i,j}$ **for** $(i,j) \in \mathcal{A}$ RVI value for road segment (i,j).

$\gamma, \alpha, \beta_0, \beta_1$ Weights to account for the impact of traffic count (γ), RVI (α), amount of deicing material (β_0), whether the road segment has been plowed (β_1) on a road segment's capacity.

$s, t \in \mathcal{V}$ Dummy source and sink nodes for simplicity in modeling.

$Dep \in \mathcal{V}$ The fuel/deicing material depot vertex.

f dummy variable used to measure flow through the network, continuous.

$f_{i,j}$ **for** $(i,j) \in \mathcal{A}$ to-flow across arc (i,j), continuous.

$f'_{i,j}$ **for** $(i,j) \in \mathcal{A}$ from-flow across arc (i,j), continuous.

$w^v_{i,j}$ **for** $(i,j) \in \mathcal{A}, v \in \mathcal{P}$ whether vehicle v traverses road segment (i,j) at least once, binary

$y^v_{i,j}$ **for** $(i,j) \in \mathcal{A}, v \in \mathcal{P}$ number of times vehicle v traverses road segment (i,j), integer

$x^v_{i,j}$ **for** $(i,j) \in \mathcal{A}, v \in \mathcal{P}$ whether vehicle v treats road segment (i,j), binary.

$s^v_{i,j}$ **for** $(i,j) \in \mathcal{A}, v \in \mathcal{P}$ deicing material applied to road segment (i,j) by vehicle v, continuous.

$p^v_{i,j}$ **for** $(i,j) \in \mathcal{A}, v \in \mathcal{P}$ whether vehicle v plows road segment (i,j), binary.

$$\mathcal{Q}(\xi,Z) = \max f \tag{3.3}$$

subject to:

$$f \le \sum_{i \in \mathcal{D}} f'_{t,i} \tag{3.4}$$

$$f_{i,j} \le \gamma(\xi) TC_{i,j} - \alpha(\xi) RVI_{i,j}$$

$$+\beta_0(\xi)\sum_{v\in\mathcal{P}} s_{i,j}^v \qquad \forall (i,j)\in\mathcal{A} \tag{3.5}$$

$$f'_{i,j} \le \gamma(\xi)TC_{i,j} - \alpha(\xi)RVI_{i,j}$$

$$+\beta_0(\xi)\sum_{v\in\mathcal{P}} s_{i,j}^v \qquad \forall (i,j)\in\mathcal{A} \tag{3.6}$$

$$f_{i,j} \le \gamma(\xi)TC_{i,j} \qquad \forall (i,j)\in\mathcal{A} \tag{3.7}$$

$$f'_{i,j} \le \gamma(\xi)TC_{i,j} \qquad \forall (i,j)\in\mathcal{A} \tag{3.8}$$

$$\sum_{i\in\delta^+(j)} f_{i,j} = \sum_{i\in\delta^-(j)} f_{j,i} \qquad \forall j\in\mathcal{V} \tag{3.9}$$

$$\sum_{i\in\delta^+(j)} f'_{i,j} = \sum_{i\in\delta^-(j)} f'_{j,i} \qquad \forall j\in\mathcal{V} \tag{3.10}$$

$$f_{s,i} \le S_i \qquad \forall i\in\mathcal{S} \tag{3.11}$$

$$f_{i,t} \le D_i \qquad \forall i\in\mathcal{D} \tag{3.12}$$

$$f'_{i,s} = f_{s,i} \qquad \forall i\in\mathcal{S} \tag{3.13}$$

$$f'_{t,i} = f_{i,t} \qquad \forall i\in\mathcal{D} \tag{3.14}$$

$$\sum_{i\in\mathcal{S}\cup\{t\}} f_{s,i} = \sum_{i\in\mathcal{S}} S_i \tag{3.15}$$

$$\sum_{i\in\mathcal{D}\cup\{s\}} f'_{t,i} = \sum_{i\in\mathcal{D}} D_i \tag{3.16}$$

$$S_{i,j}^{min} x_{i,j}^v \le s_{i,j}^v \qquad \forall (i,j)\in\mathcal{A}\ \forall v\in\mathcal{P} \tag{3.17}$$

$$\sum_{v\in\mathcal{P}} s_{i,j}^v \le S_{i,j}^{max} \qquad \forall (i,j)\in\mathcal{A} \tag{3.18}$$

$$\sum_{v\in\mathcal{P}}\sum_{(i,j)\in\mathcal{A}} s_{i,j}^v \le Z \tag{3.19}$$

$$x_{i,j}^v \le y_{i,j}^v \qquad \forall (i,j)\in\mathcal{A}\ \forall v\in\mathcal{P} \tag{3.20}$$

$$y_{i,j}^v \le M\cdot w_{i,j}^v \qquad \forall (i,j)\in\mathcal{A}\ \forall v\in\mathcal{P} \tag{3.21}$$

$$\sum_{(i,j)\in\mathcal{A}} s_{i,j}^v \le Q_S^v \qquad \forall v\in\mathcal{P} \tag{3.22}$$

$$\sum_{(i,j)\in\mathcal{A}} F_{i,j}^v y_{i,j}^v \le Q_F^v \qquad \forall v\in\mathcal{P} \tag{3.23}$$

$$\sum_{v\in\mathcal{P}}\sum_{(i,j)\in A} F_{i,j}^v y_{i,j}^v \le F^{total} \tag{3.24}$$

$$\sum_{v\in\mathcal{P}}\sum_{(i,j)\in A} T_{i,j}y^v_{i,j}\leq T^{total} \tag{3.25}$$

$$\sum_{i\in\delta^+(j)} y^v_{i,j}=\sum_{i\in\delta^-(j)} y^v_{j,i} \qquad \forall j\in\mathcal{V}\ \forall v\in\mathcal{P} \tag{3.26}$$

$$\sum_{j\in\delta^+(Dep)} y^v_{Dep,j}=1 \qquad \forall v\in\mathcal{P} \tag{3.27}$$

$$\sum_{\substack{(i,j)\in\mathcal{A}\\ i\in C,j\notin C}} w^v_{i,j}\geq\frac{1}{|G[C]|_E}\sum_{\substack{(i,j)\in\mathcal{A}\\ i,j\in C}} w^v_{i,j} \qquad \forall C\subsetneq V\setminus\{Dep\}\ \forall v\in\mathcal{P} \tag{3.28}$$

$$w^v_{i,j}\in\{0,1\} \qquad \forall(i,j)\in\mathcal{A}\ \forall v\in\mathcal{P} \tag{3.29}$$

$$x^v_{i,j}\in\{0,1\} \qquad \forall(i,j)\in\mathcal{A}\ \forall v\in\mathcal{P} \tag{3.30}$$

$$y^v_{i,j}\in\mathbb{Z}_+ \qquad \forall(i,j)\in\mathcal{A}\ \forall v\in\mathcal{P} \tag{3.31}$$

Let $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ be a multigraph describing the road network under consideration. It is assumed that the nodes are one of three types: nodes to represent intersections, nodes to represent residences, and nodes to represent businesses or other locations people travel to throughout the day. The mathematical formulation in (3.3) models traffic through the road network as a two-stage network flow. A two-stage flow model is used in order to capture the dominant traffic patterns throughout the day, with the first stage representing morning traffic (people leaving home to go to work) and the second stage representing evening traffic (people leaving work to return home). As such, the nodes representing residences act as sources for the first-stage flow and sinks for the second-stage flow, while the nodes representing businesses or other locations of interest act as sinks for the first-stage flows and sources for the second-stage flows. In (3.3), we note that the second-stage flows depend on the first stage flows (i.e. you cannot leave work to go home if you were unable to get to work initially). However, one important difference between this flow model and other forms of multi-flow models is how the first and second stage flows interact. For most multi-commodity network flow models, the sum of the flows across any given arc must obey the capacity limitations of that arc. Because these flows are meant to represent the flows of traffic that occur at different times of the day, we only require that each flow *individually* obey the capacity limitations of that arc. For example, if an arc (i, j) has a capacity of 5, then the first-stage flow must be no more than 5, and the second-stage flow must be no more than 5, but the sum of the first- and second-stage flows may be more than 5. This point is illustrated in Figure 3.1.

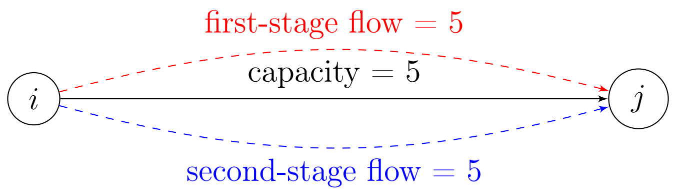


Figure 3.1: Feasible first- and second-stage flows across arc (i, j).

The objective of (3.3) is to maximize the second stage flow through the network, which is the number of people who were able to go to and return from work successfully. We assume the capacities of the arcs are a function of the traffic counts on the corresponding road segment on a typical day, the driving conditions on the road due to the snow event, and the degree to which the road segment has been treated. The degree to which the capacity of a given arc has been lowered due to the driving conditions depends on a metric we call the *road vulnerability index* (RVI), with higher RVI values corresponding to more dangerous driving conditions. RVI values are computed using meteorological and road slope data, with the idea being that road segments which are shaded and hilly will, all other factors being equal, be more dangerous to drive on than flat road segments in direct sunlight. We assume that the capacity of a particular arc is a linear function of these parameters. Informally, we think of the traffic count on a typical day as being the default capacity of that road segment, with the capacity being discounted proportionally to the RVI of the road segment. Capacities of road segments can be increased through treatment (plowing and/or applying deicing materials), with higher levels of treatment reducing the discount factor of the snow event.

Overall, (3.3) can be thought of as having three primary aspects. In a sense, the flow aspect of the problem is the primary aspect of (3.3). The flows across the network have the standard network flow constraints, which are contained in (3.4)–(3.16). Because the capacities of arcs can be increased through treating road segments, the flow aspect of (3.3) is subject to resource allocation constraints, contained in (3.17)–(3.25). Lastly, the resource allocation aspect of the problem is contingent upon being able to produce feasible vehicle routes to apply the treatments. The constraints that model this aspect are (3.26)–(3.28).

In (3.3), there have four parameters that play a significant role, namely $\alpha, \gamma, \beta_0, \beta_1$. These are weights that determine the impact of traffic counts, RVI values, and treatment levels on the capacity of an arc. We assume that a given snow event can be described by the values these parameters take for that snow event. As a result, these are treated as functions of the snow event realization ξ.

We note that the formulation given in (3.1) relies on solving (3.3) for a given budget Z and parameter realization ξ. An instance of this problem on the scale of a modest city such as Knoxville, TN would result in a substantial MIP. Due to the combinatorial nature of these types of problems, MIPs of this scale are often intractable, in the sense that we are typically unable to solve them within a reasonable amount of time using current hardware and solution techniques. However, we note that for this particular problem, assuming a fixed snow event realization ξ, (3.1) is a problem in a single variable, Z. Further, we note that (3.1) is monotonically increasing with respect to Z. Because of this, a number of simple optimization algorithms are suitable for solving (3.1), such as a binary search, provided (3.3) can be solved or approximated reasonably quickly.

Because of the need to solve (3.3) quickly, we developed a heuristic that exploits the nature of the problem based off work by Fulkerson in [22], in which he introduces the *Parametric Budget Problem* (PBP). Other authors will refer to this problem as the *Capacity Expansion Network Flow Problem* [35]. We note that the complicating aspect of (3.3) is the vehicle routing portion. Without this aspect, (3.3) is nearly identical to the PBP, with the only difference being the PBP assumes a single source and sink, whereas (3.3) has multiple sources and sinks. Fulkerson presents an algorithm in [22] which solves the PBP by iteratively finding a source-sink path along which the flow can be increased with the lowest marginal cost per unit flow. This is repeated until either the budget for increasing arc capacities runs out, or no finite-cost path is present in the graph. This technique can be extended to produce high-quality solutions to a relaxed version of (3.3) in which the vehicle routing aspect of the problem is temporarily ignored until a near-optimal allocation of deicing resources which exhausts all available resources is achieved. This is done by computing approximate marginal costs for source-sink paths, in much the same fashion as Fulkerson's approach. It is worth noting that this approach is just a heuristic to solving (3.3) even with the vehicle routing aspect being ignored, as it ultimately falls victim to the same difficulties that graph theoretic approaches to solving multi-commodity flow problems do. After the quasi-ideal resource allocation scheme is computed, a constructive vehicle routing heuristic is employed to attempt to find feasible deicing vehicle routes that allow these road segments to be treated.

3.3 Computational Experiments

3.3.1 Experimental Procedure

Two sets of instances were generated for the computational experiments. One set of instances were generated in a hybrid Erdős-Rényi [40] and modified Barabási-Albert [4] fashion, and another set was generated using a combination of actual and simulated data for the Knoxville, TN road network. All coding was done using Python 2.7, with the integer programming solver Gurobi version 7.0.1 being used. All tests were done on a 16 core (32 thread) workstation with 2 Intel Xeon E5-2670 processors (2.6 GHz), 256 GB RAM, running Ubuntu 14.04.5. In the exact approach, as detailed in [2], certain constraints were added on the fly in a lazy fashion by checking candidate solutions for violated constraints. The time limit given to each instance depended on the type of instance, with the random problems being given a time limit of 30 minutes and the Knoxville instances being given 60 minutes.

In the flow-based model, we assume each node in the graph corresponds to a location in the road network. The locations are divided into residences, intersections of road segments, and other locations of interest. As a result, there are three sets of nodes in the corresponding graphs. In the random instances, the arc set for the intersection nodes were generated in an Erdős-Rényi fashion [40] so that the average in- and out-degree of a node was 3. If the resulting graph was not strongly connected, the arcs were removed and another arc set was generated. This was repeated until the resulting graph was strongly connected. At this point, each node corresponding to a residence or other location of interest was added to the graph in a Barabási-Albert fashion with degree 2. Three instances of varying sizes were generated in this manner, with 200, 300, and 400 nodes. In each instance, 80% of the nodes were assigned as intersections, 10% as residences, and 10% as other locations. One of the intersections was chosen at random to act as the location of the salt and fuel depot. All graph parameters except traffic flows for these instances were generated from exponential distributions. The traffic counts were generated by solving a network flow problem on the graph under the assumption that no traffic had been interdicted by snowfall.

For the other instances, 3 areas within Knox County, TN were utilized to generate the graphs. For each location within these areas, data is available that allows us to distinguish intersections from non-intersections. Additionally, population estimates exist for each non-intersection location. A rather complete picture is available for these instances in regard to all parameters except traffic counts. The limited traffic count data available was used in conjunction with an approach similar to that used in the random graphs to generate traffic counts for these instances. Since the areas chosen do not contain salt depots, an intersection node was chosen at random for each area to act as the salt and fuel depot.

3.3.2 Results

Figures 3.2, 3.3 summarize the results. For each instance, n reports the number of nodes in the graph and m reports the number of arcs in the graph. *Budget* reports the deicing material budget, which has been normalized to fall between 0 and 1, with 0 being no deicing materials allotted and 1 being enough to treat the entire network completely. *MIP* corresponds to the solution obtained using the full model in (3.3) with Gurobi as the solver, while *Heuristic* reports the data for the constructive heuristic based off [22] which was outlined in Section 3.2. While we note in Section 3.2 that a simple binary search should be sufficient to solve (3.1), we have constructed the Pareto frontier for the budget levels that are of interest for this problem. We have highlighted the budget necessary to ensure that no more than 20% of the population is negatively affected by the snow event.

While solve times are not reported here, we note that the MIP approach unsurprisingly had difficulties solving some of the larger instances in the allowed amount of time. In the random $n = 400$ instance, the MIP approach ran out of time in almost half of the tests. The

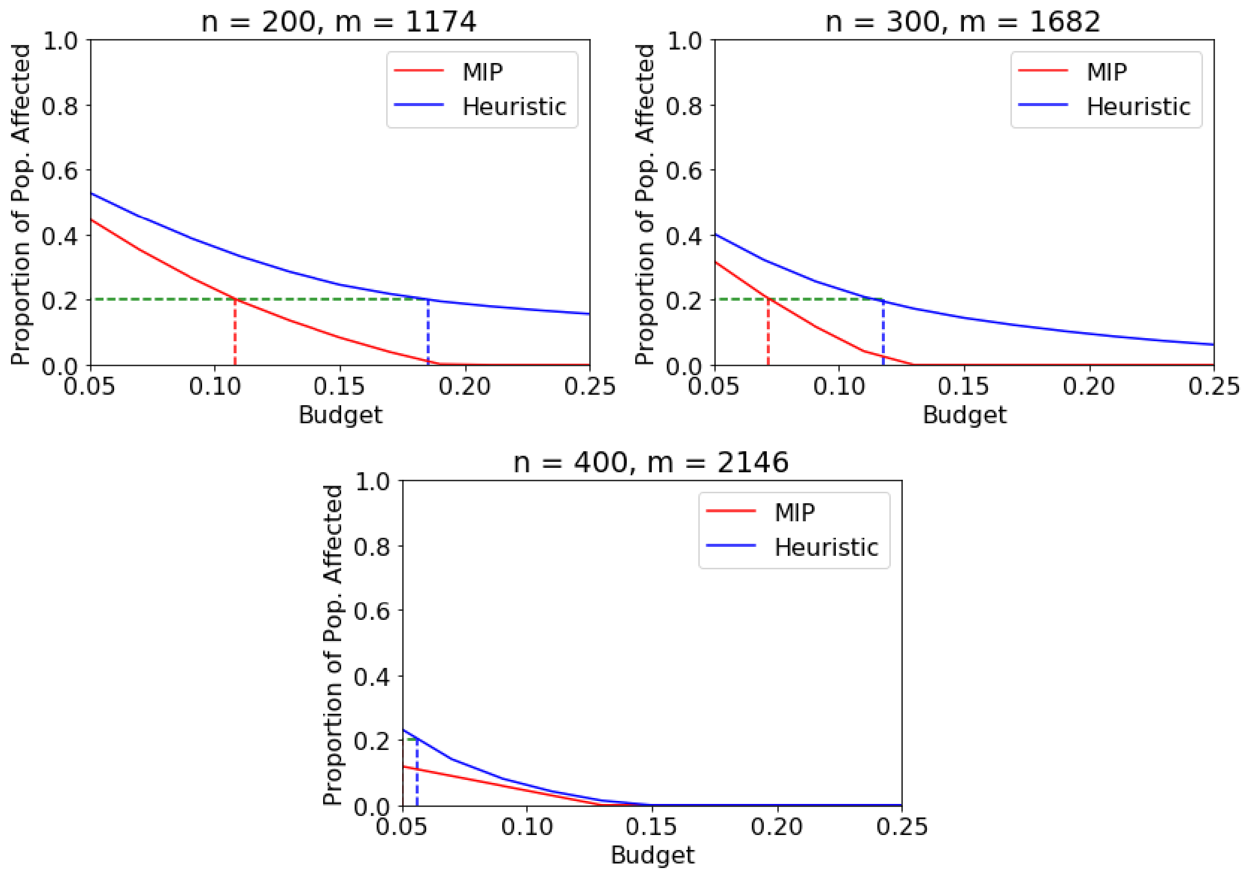


Figure 3.2: Proportion of deicing materials needed to achieve various outcomes in random instances.

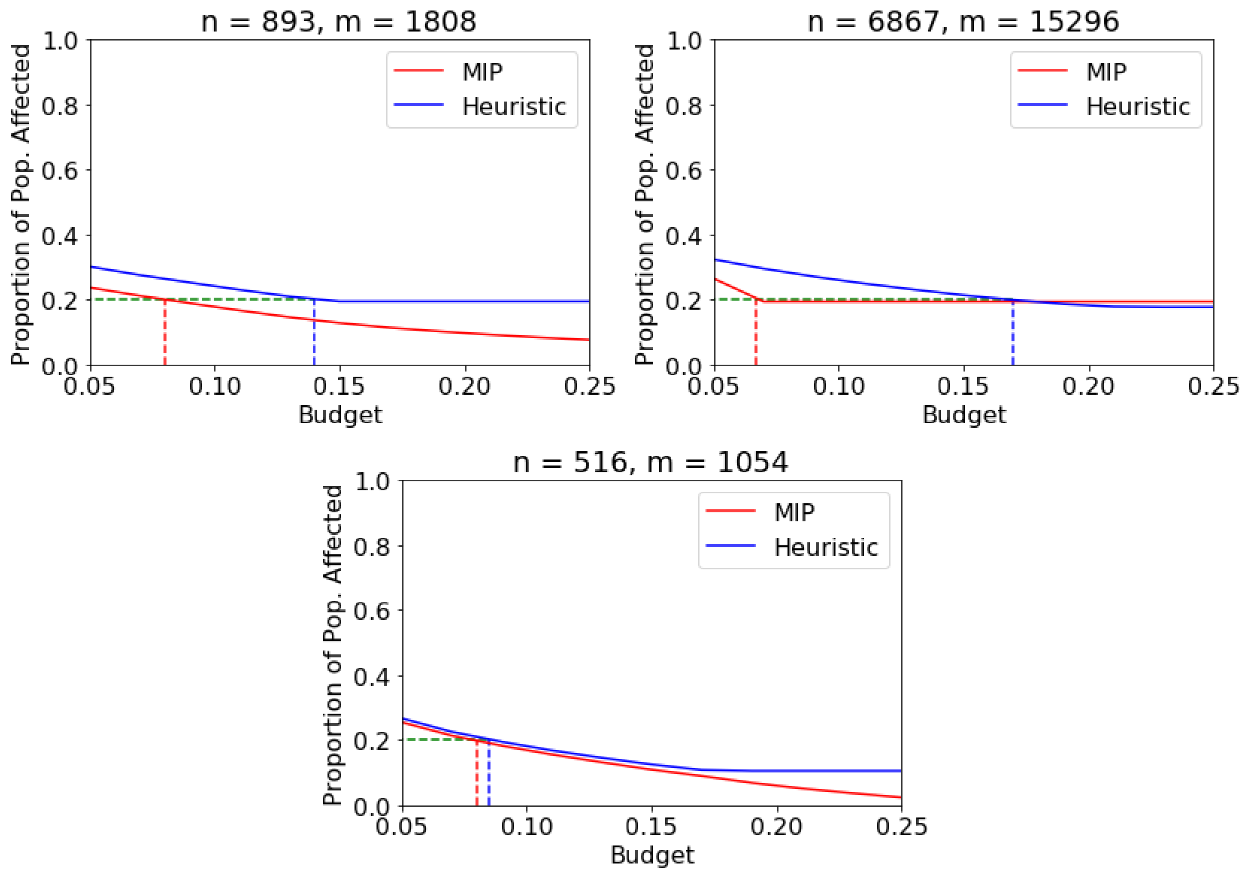


Figure 3.3: Proportion of deicing materials needed to achieve various outcomes in Knoxville, TN instances.

tests on the largest Knox County instance, $n = 6867$, the MIP approach ran out of time in every test. When this occurred, the best incumbent solution was used for Figures 3.2-3.3.

While the heuristic approach was always able to outperform the MIP approach in regard to run times, often the MIP approach was able to solve the problem to provable optimality. In these cases, it is of course impossible for the heuristic solution to be of higher quality than the MIP solution. However, in the largest Knox County instance, we note that the heuristic approach was able to outperform the MIP approach at the higher budget levels.

3.4 Conclusion

We note the sometimes drastic convexity of the Pareto frontiers within the results. This illustrates that the proportion of affected population was highly sensitive to the budget at lower budget levels, with modest increases in budget levels sometimes corresponding to quite drastic reductions in the affected population. This suggests a few routes for future research, with the most obvious one being to further explore the sensitivity of this region. This sensitivity also suggests the need for high-quality weather forecasts in order to predict required budget levels accurately. Additionally, in a sense, this work can be seen as an attempt to maximize the utility gained from public funds. Perhaps instead of optimizing under the constraint of a minimum acceptable level of service as was done in (3.1), one could instead use the slope of the Pareto frontier and determine a budget while considering the marginal utility gained from additional funds.

Chapter 4

Deploying Automated Mobility Districts

This chapter is related to a paper submitted for publication:

> H M Abdul Aziz, Tony K. Rodriguez, Venu Garikapati, Lei Zhu, Stanley E. Young, Yuche Chen. Optimizing Fleet Operations in Automated Mobility Districts: serving On-demand Mobility with Automated Electric Shuttles. *Submitted to Transportation Research Part C.*

Figures 4.1, 4.2, 4.4, 4.3, and 4.5 come from that paper, as well as the example in Section 4.4.3.

In this chapter, we develop a mixed integer programming formulation which can be used to address the operational decisions of routing and fleet composition for automated mobility districits. Because this formulate scales poorly with the input size of instance, we also develop a two constructive heuristics to produce solutions, as well as a Tabu search that can be used to refine these solutions. Computational experiments are run using data from Greenville, SC. These experiments demonstrate that the methods developed are capable of producing viable solutions and aid in operations planning.

4.1 Motivation and Background

Due to automated vehicle technology, the mobility-as-a-service landscape will soon see changes that would have been difficult to imagine until recently. However, there are still many hurdles present which prevent shared, automated vehicle systems from completely overhauling the status quo, with the main limitations being economic considerations for the automobile industry as a whole, regulations which are ill-suited for fully autonomous vehicles, and a lack of infrastructure suitable for such fleets [19, 3]. Despite these challenges, low-capacity automated electric shuttles (AES) fleets have been deployed in several cities, typically within dense urban areas, in an effort to provide cost-effective and energy-saving alternatives in personally-owned vehicles. Currently, Europe seems to be leading the way, with *EasyMile* having deployed EZ10 AESs in dozens of locations[1]. *Local Motors'* olli is being developed and planned for deployment in several US cities, including Knoxville, TN[2], and recently the Federal Highway Administration awarded a grant to deploy these technologies in three locations within Greenville, SC [20].

Due to the limitations previously mentioned, many of the currently deployed AES fleets are within geo-fenced regions, often with limited access to outside traffic. While most locations will adopt these technologies incrementally, in a "something everywhere" fashion, these current regions are realizing more of an "everything somewhere" approach. Due to this, Young et al. [51] has proposed a development framework known as an Automated Mobility District (AMD). An AMD is a small-scale implementation of automated connected vehicle technology that is designed to see the full benefits an AES fleet can offer. There are four key characteristics of an AMD deployment: (i) a fleet of fully autonomous vehicles (such as an AES fleet); (ii) a geo-fenced service area; (iii) strict regulations on the access of roadways to outside traffic; and (iv) multi-modal access at the boundaries of the geo-fence.

4.2 Automated Mobility Districts

Automated mobility districts (AMDs) have often been studied in the context of mobility as a service. As such, much of the focus of these works has been towards optimizing mobility or customer satisfaction. Because of the looming threats of climate change and energy insecurities, overall energy consumption of the system has become a motivating factor in

[1]http://www.easymile.com/
[2]https://localmotors.com/meet-olli/

studying and deploying AMDs, with Zhu et al. [52] reporting greater energy efficiency, in addition to increased mobility, from the use of AMDs.

This work is meant to extend the modeling and simulation of AMD models developed by Young et al. [51] and Zhu et al. [52] by optimizing fleet operations within the AMD while accounting for several practical considerations. To capture the efficiency of fleet operations, the objective function will be the total on-the-road time of the vehicles within the AMD.

The first aspect we will address is customer satisfaction. In this work, customer satisfaction is modeled by having time windows in which customers must be picked up via constraints. We note that, in essence, this approach addresses customer satisfaction by ensuring that 100% of customers are picked up within their preferred time window. It is important to note that implementing optimization models within physical AMDs will likely necessitate using stochastic or robust approaches to account for the inherently stochastic nature of ride-hailing and sharing. However, because common approaches to dealing with such stochasticity involve solving multiple deterministic models, we will not directly address the stochastic nature of this problem in the optimization model. We note, though, that chance constraints or penalty methods for unsatisfied customers seem to be an obvious way to incorporate these considerations into the model directly.

Since the vehicles are satisfying passenger requests, in addition to time windows for customer pickups, another aspect of AMDs is customer loading and unloading times as well as the capacities of the vehicles within the AMD. Both of these will be modeled by constraints within the MIP model presented. Further, because the range of electric vehicles is often very limited, the model will also include constraints limiting the range of the vehicle routes. We show the impacts of having full and partial demand information available. Lastly, in the context of vehicle routing problems, we show a method that can be used to account for travel costs that vary over time.

4.3 MIP Formulation

The problem studied in this work consists of routing autonomous electric vehicles to satisfy pickup and drop off locations within a network using the least amount of vehicle-hours possible. In addition to the typical aspects of vehicle routing models, these routes must obey pickup and drop off time window constraints, maximum distance constraints, vehicle capacity constraints, loading and unloading times of pickups and deliveries, and varying travel times.

For the parameters of the problem, we let $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ be a multigraph, where $\mathcal{V}$ is the set of pickup/delivery locations within the network as well as the starting and stopping depots for each vehicle, and the arc set $\mathcal{A}$ represents the paths to and from each location. We denote by $\mathcal{K}$ the set of vehicles, $\mathcal{R}$ the set of pickup/delivery requests, and $\mathcal{T}$ the time steps. For a request $r \in \mathcal{R}$, we let $p(r) \in \mathcal{V}$ be the pickup node for r, and $d(r) \in \mathcal{V}$ be the delivery node. Since each pickup and delivery is assumed to require some non-zero loading and unloading time, we let $\sigma(r)$ be the loading time required to pick up r, and $\theta(r)$ be the unloading time required to deliver r. Further, the pickups must occur within some time window, and we denote this window by $[s_r, e_r]$, with s_r being the earliest pickup time, and e_r being the latest pickup time. With respect to the on-demand pickup and delivery problem, s_r can be thought of as the time at which the pickup request comes in.

For each arc $(i, j) \in \mathcal{A}$ and time period $t \in \mathcal{T}$, we let $\ell_{i,j}^t$ be the length of the path from i to j in time period t. For each vehicle $k \in \mathcal{K}$, $\tau_{i,j}^{k,t}$ is the time required for vehicle k to travel from i to j in time period t, and $c_{i,j}^{k,t}$ is the "cost" of traveling from i to j in time period t be vehicle k. In our experiments, we assume $c_{i,j}^{k,t} = \tau_{i,j}^{k,t}$, however we make the distinction here for the sake of generality. Lastly, for each vehicle $k \in \mathcal{K}$, we let u_k be the passenger capacity of k, and d_k be the maximum distance vehicle k can travel. A summary of the parameters is given below.

$\mathcal{G} = (\mathcal{V}, \mathcal{A})$ Directed multigraph of the road network.

$\mathcal{K}$ Set of vehicles.

$\mathcal{R}$ Set of pickup-delivery requests.

T Maximum number of time steps.

u_k Capacity of vehicle $k \in \mathcal{K}$.

$o(k), o'(k)$ Start and end depots of $k \in \mathcal{K}$, respectively.

$p(r), d(r)$ Pickup and delivery location for $r \in \mathcal{R}$, respectively.

$\sigma(r), \theta(r)$ Amount of time required to pickup and drop off $r \in \mathcal{R}$, respectively.

c_{ij}^{kt} Cost for vehicle $k \in \mathcal{K}$ to traverse arc (ij) in time step t.

s_r, e_r Window for pickup times for $r \in \mathcal{R}$.

d_k Maximum distance $k \in \mathcal{K}$ can travel on a single charge.

ℓ_{ij} Length of arc $(ij) \in \mathcal{A}$.

τ_{ij}^{kt} Time required for $k \in \mathcal{K}$ to traverse $(ij) \in \mathcal{A}$ in time step t.

b^t Time at which period t begins, $1 \leq t \leq T$.

For the variables, because the travel costs (and thus paths, times, and lengths) can change in each time period, we must modify the typical VRP variables to take into account which time period events occur. For each $(i,j) \in \mathcal{A}$, $k \in \mathcal{K}$, and $t \in \mathcal{T}$, variable $x_{i,j}^{k,t}$ indicates if vehicle k traverses (i,j) in time period t. To ensure the x variables take on the appropriate values, we also introduce $w_i^{k,t}$ to indicate if vehicle $k \in \mathcal{K}$ leaves node $i \in \mathcal{V}$ in time period $t \in \mathcal{T}$. Further, as pickups and deliveries are being made on vehicles with capacity constraints, for each $(i,j) \in \mathcal{A}$, $k \in \mathcal{K}$, and $r \in \mathcal{R}$, we let $y_{i,j}^{k,r}$ indicate if vehicle k traverses (i,j) while actively satisfying request r. To account for the subtour elimination constraints in a polynomial fashion [32], for every pair of nodes $i \in \mathcal{V}, j \in \mathcal{V}$ and for each vehicle $k \in \mathcal{K}$, the variable $z_{i,j}^k$ indicates if i precedes j in the route of vehicle k (not necessarily immediately). Lastly, as the times at which events occur is important, for each node $i \in \mathcal{V}$ and for each vehicle $k \in \mathcal{K}$, we let t_i^k be the arrival time of k at node i, and $\bar{t}_i^k$ be the departure time of k from node i. A description of these variables, followed by the MIP model, is presented below.

w_i^{kt} Indicates if vehicle $k \in \mathcal{K}$ leaves $i \in \mathcal{V}$ in time period t. Binary.

x_{ij}^{kt} Indicates if vehicle $k \in \mathcal{K}$ traverses $(ij) \in \mathcal{A}$ in time period t. Binary.

y_{ij}^{kr} Indicates if vehicle $k \in \mathcal{K}$ traverses $(ij) \in \mathcal{A}$ while transporting request $r \in \mathcal{R}$. Binary.

z_{ij}^k Indicates if $i \in \mathcal{V}$ precedes $j \in \mathcal{V}$ in route of vehicle $k \in \mathcal{K}$. Binary.

t_i^k Arrival time of $k \in \mathcal{K}$ at node $i \in \mathcal{V}$. Continuous.

$\bar{t}_i^k$ Departure time of $k \in \mathcal{K}$ at node $i \in \mathcal{V}$. Continuous.

$$\min \sum_{k \in \mathcal{K}} \sum_{ij \in \mathcal{A}} \sum_{1 \leq t \leq T} c_{ij}^{kt} x_{ij}^{kt} \tag{4.1}$$

$$\text{s.t.} \quad \sum_{j | ij \in \mathcal{A}} x_{ij}^{k1} \leq 1 \qquad \forall k \in \mathcal{K}\ \forall i = o(k) \tag{4.2}$$

$$\sum_{j | ij \in \mathcal{A}} x_{ij}^{k1} = \sum_{j | j\ell \in \mathcal{A}} \sum_{1 \leq t \leq T} x_{j\ell}^{kt} \qquad \forall k \in \mathcal{K}\ \forall i = o(k)\ \forall \ell = o'(k) \tag{4.3}$$

$$\sum_{i | ij \in \mathcal{A}} x_{ij}^{kt} = \sum_{\ell | j\ell} \sum_{t' \geq t} x_{j\ell}^{kt'} \qquad \forall t \leq T\ \forall k \in \mathcal{K}\ \forall j \in \mathcal{V} \setminus \{o(k), o'(k)\} \tag{4.4}$$

$$\sum_{k \in \mathcal{K}} \sum_{j | ij \in \mathcal{A}} y_{ij}^{kr} = 1 \qquad \forall r \in \mathcal{R}\ \forall i = p(r) \tag{4.5}$$

$$\sum_{k \in \mathcal{K}} \sum_{j | ji \in \mathcal{A}} y_{ji}^{kr} = 1 \qquad \forall r \in \mathcal{R}\ \forall i = d(r) \tag{4.6}$$

$$\sum_{i | ij \in \mathcal{A}} y_{ij}^{kr} = \sum_{\ell | j\ell \in \mathcal{A}} y_{j\ell}^{kr} \qquad \forall k \in \mathcal{K}\ \forall r \in \mathcal{R}\ \forall j \in \mathcal{V}, i \neq p(r), \ell \neq d(r) \tag{4.7}$$

$$\sum_{r \in \mathcal{R}} y_{ij}^{kr} \leq u_k \sum_{1 \leq t \leq T} x_{ij}^{kt} \qquad \forall (ij) \in \mathcal{A}\ \forall k \in \mathcal{K} \tag{4.8}$$

$$\sum_{1 \leq t \leq T} x_{ij}^{kt} \leq z_{ij}^k \qquad \forall i, j \in \mathcal{V}\ \forall k \in \mathcal{K}, o(k) \neq i, o'(k) \neq j \tag{4.9}$$

$$z_{ij}^k + z_{ji}^k \leq 1 \qquad \forall i,j \in \mathcal{V}\ \forall k \in \mathcal{K}, o(k) \neq i, o'(k) \neq j \tag{4.10}$$

$$z_{ij}^k + z_{j\ell}^k + z_{\ell i}^k \leq 2 \qquad \forall i,j,\ell \in \mathcal{V}\ \forall k \in \mathcal{K} \tag{4.11}$$

$$\bar{t}_i^k + \tau_{ij}^{kt} - t_j^k \leq M\left(1 - x_{ij}^{kt}\right) \qquad \forall (ij) \in \mathcal{A}\ \forall k \in \mathcal{K}\ \forall t \leq T \tag{4.12}$$

$$t_{p(r)}^k + \sigma(r) \leq \bar{t}_{p(r)}^k \qquad \forall r \in \mathcal{R}\ \forall k \in \mathcal{K} \tag{4.13}$$

$$t_{d(r)}^k + \theta(r) \leq \bar{t}_{d(r)}^k \qquad \forall r \in \mathcal{R}\ \forall k \in \mathcal{K} \tag{4.14}$$

$$s_r \leq t_{p(r)}^k \qquad \forall r \in \mathcal{R}\ \forall k \in \mathcal{K} \tag{4.15}$$

$$\bar{t}_{p(r)}^k \leq e_r \qquad \forall r \in \mathcal{R}\ \forall k \in \mathcal{K} \tag{4.16}$$

$$\sum_{ij \in \mathcal{A}} \sum_{1 \leq t \leq T} \ell_{ij} x_{ij}^{kt} \leq d_k \qquad \forall k \in \mathcal{K} \tag{4.17}$$

$$\sum_{j | ij \in \mathcal{A}} x_{ij}^{kt} \leq w_i^{kt} \qquad \forall k \in \mathcal{K}\ \forall t \leq T\ \forall i \in \mathcal{V} \tag{4.18}$$

$$\sum_{1 \leq t \leq T} w_i^{kt} \leq 1 \qquad \forall k \in \mathcal{K}\ \forall i \in \mathcal{V} \tag{4.19}$$

$$b^t - M(1 - w_i^{kt}) \leq \bar{t}_i^k \leq b^{t+1} + M(1 - w_i^{kt}) \qquad \forall i \in \mathcal{V}\ \forall k \in \mathcal{K}\ \forall t \leq T \tag{4.20}$$

$$\bar{t}_{p(r)}^k \leq t_{d(r)}^k \qquad \forall k \in \mathcal{K}\ \forall r \in \mathcal{R} \tag{4.21}$$

$$w_i^{kt} \in \{0,1\} \qquad \forall t \leq T\ \forall (ij) \in \mathcal{A}\ \forall k \in \mathcal{K}\ \forall r \in \mathcal{R} \tag{4.22}$$

$$x_{ij}^{kt} \in \{0,1\} \qquad \forall t \leq T \; \forall (ij) \in \mathcal{A} \; \forall k \in \mathcal{K} \; \forall r \in \mathcal{R} \tag{4.23}$$

$$y_{ij}^{kr} \in \{0,1\} \qquad \forall t \leq T \; \forall (ij) \in \mathcal{A} \; \forall k \in \mathcal{K} \; \forall r \in \mathcal{R} \tag{4.24}$$

$$z_{ij}^{k} \in \{0,1\} \qquad \forall t \leq T \; \forall (ij) \in \mathcal{A} \; \forall k \in \mathcal{K} \; \forall r \in \mathcal{R} \tag{4.25}$$

For the details of the model, 4.2 ensure that each vehicle is assigned to at most a single route, while 4.3 ensure the routes start and stop at allowable depots. As with many VRP variations that use a network flow-inspired approach, 4.4 force vehicles to leave each node traveled to (flow in equals flow out).

Since the vehicles are being routed to pickup and deliver requests, constraints 4.5-4.6 require that each request is satisfied, and 4.7 are flow conversation constraints for passenger transportation (passengers must leave intermediary nodes they enter). Constraints 4.8 perform two roles. First, they ensure that vehicles which transport passengers over an arc must traverse that arc, and second that vehicles cannot carry more passengers than their capacity allows along any given arc.

The next few constraints constitute the subtour elimination constraints. 4.9 ensures that node i precedes node j in the route of a vehicle if that vehicle traverses arc (i,j). Further, since it cannot be the case that two nodes both precede one another in any given route, constraints 4.10 are added. Lastly, constraints 4.11 act as a kind of triangle "precedence" inequality constraints. These three sets together constitute the subtour elimination constraints [32].

The next sets of constraints account for the temporal nature of this problem. Constraints 4.12 ensure that, if a vehicle travels from i to j, then the arrival time at j cannot be earlier than the departure time from i plus the time required to travel from i to j. To account for the fact that pickups and deliveries require some amount of time to perform, constraints 4.13-4.14 are added. To model the fact that the pickups must be within some time window, constraints 4.15-4.16 are included. We note that while this model does not account for time windows on deliveries, constraints similar to these could be added to model that as well.

Finishing out the model, 4.17 are distance traveled constraints for the vehicles. 4.18 tie the x and w variables together, ensuring that if a vehicle traverses an arc leaving a node i in time period t, then it leaves node i in time period t. 4.19 force vehicles to leave a node in at most one time period. To ensure the w variables take on the appropriate values, we include constraints 4.20. Since pickups must occur before deliveries, 4.21 are present. And lastly, 4.22-4.25 are the integrality constraints.

It is important to note that the model presented in 4.1 assumes complete knowledge of the problem parameters. While many classical optimization problems have this assumption built into them, many practical problems often have the specifics of the parameters revealed over time rather than a priori. We will further discuss this limitation in Section 4.4, but it should be noted that this will likely prove to be an important detail in any practical implementation of this approach for this problem.

4.4 Solution Method

Since a mixed integer formulation of the problem is given in 4.1, it is tempting simply to attempt to use a commercial solver, such as Gurobi or CPLEX, to solve this problem. However, while the formulation presented is polynomial in size, we note that this formulation is still quite large for even modest instances. Because of this limitation, we present a constructive heuristic with a refinement procedure. In the first phase, an online approach is used which assigns incoming requests to routes based on the marginal cost of adding that request to a given route. In order to refine these initial routes, a Tabu search is utilized [24].

4.4.1 Online Route Construction

In order to construct the initial routes of the vehicles, an online framework is utilized. In many classical optimization problems, it is assumed that all of the information relevant to solving the problem is known a priori. By contrast, in an online problem, only an incomplete knowledge of the future is available initially. In many of these problems, information is revealed over time, often requiring the solutions previously found to be updated. While the fields of robust and stochastic optimization have been developed to address these types of problems, two major drawbacks to utilizing these frameworks exist. First, in order to produce solutions which can be used in practice, high-quality data about the information of the parameters contained in the problem is often required. In addition to needing high-quality data, any naive implementation of these approaches will require significant computational resources.

Because of these limitations, another common approach to addressing the online nature of certain problems is the use of online algorithms. An online algorithm is one which is designed to be able to construct partial solutions when given only partial information, while being able to update these solutions as more information is revealed. Due to the on-demand nature of the problem considered in this work, there is an obvious online greedy algorithm that can be used to produce a set of initial solutions. Each vehicle maintains a route which contains the nodes visited by the vehicle, the times at which these visits occur, and the pickups and deliveries performed. Previously, the pickups were assumed to have a time window $[s_r, e_r]$ during which they could be picked up. Another way to think of this is the pickup request comes in at time s_r, and the passenger is willing to wait up to $e_r - s_r$ units of time for the pickup to occur. For this approach, the requests are ordered according to the time at which the request is made (s_r). When a new request is made, the marginal cost of adding this request to each vehicle's route is computed. The vehicle which can satisfy this request at the lowest marginal cost is then assigned to this request, and its route is updated to reflect this.

Since there are many ways in which a request can be added to a vehicle's route, all pickups and deliveries are done in a first-in, first-out (FIFO) fashion. That is, given two requests r and r' and any route which contains both of them, if $s_r < s_{r'}$ (the request time of r occurs before the request time of r'), then r must be picked up before r', and r must be delivered before r'. Since the pickups have time windows in which they must occur and the deliveries do not, the times at which pickups occur is more restrictive. Due to the greater freedom of time frames in which deliveries can occur, performing pickups is given priority over performing deliveries. This was done by scheduling the deliveries in a greedy fashion, assigning them to occur at the first feasible time; that is, after the corresponding pickup had occurred, and in between other node visits in such a way as to maintain overall route feasibility.

We now turn our attention to the online aspect of the problem. As stated previously, the MIP model assumes all of the parameter information is available a priori, whereas any (at least naive) real-world implementation will find this assumption untenable. Because of this, we consider two different ways to construct the initial routes. The first assumes every aspect of the route can be reconfigured each time the route is updated. We will call this the *look-ahead* approach, as, in a sense, it mimics what the outcome would be if the vehicles could, at each node, look ahead in their future route and take the path that would result in the lowest cost. When constructing the look-ahead routes, and especially in the Tabu search refinement, it is useful to think of the routes as a set of requests to be fulfilled, rather than the sequence of nodes, times, and requests, since the route can be readily reconstructed given just the requests to be fulfilled.

The second method used to construct the initial routes is truly an online approach. Each time a new request r is made, it is assumed that the node visits in the route which occurred before s_r (the time at which request r is made) are fixed and only the future route can be updated. We will call this approach the *without look-ahead* approach.

4.4.2 Refinement with Tabu Search

After the initial routes are constructed, the refinement phase begins. We note that the refinement phase is meant to provide a solution approach for the MIP given in 4.1, and as such, the initial routes will be the look-ahead routes constructed. In this phase, as mentioned previously, it is useful to think of the routes as just a set of requests to be satisfied. In order to refine the initial routes, the approach used was a Tabu search [24] with request interchanges. Tabu search is a type of local search heuristic with short-term memory, while request interchange defines what it means for solutions to be neighbors. Tabu search is an iterative heuristic in which neighbors of the current solution are examined. The neighbor with the best objective value is then chosen as the current solution in the next iteration of the search. We note that a neighbor does not need to be improving to be chosen, it simply needs to be the neighbor with the best objective value. Because of this, Tabu search is useful for escaping locally optimal points which are not globally optimal. To avoid cycling (that is, continuously choosing only a small set of solutions), a Tabu list is maintained. Items or structures on the Tabu list represent characteristics which the examined neighbors are forbidden from having. Neighbors found to have the characteristics on the Tabu list are thus not considered as potential candidates for the next iteration.

To define the neighbors of a solution, we think of the routes as a set of clusters (or sets) of the requests. Two solutions (sets of routes, which are themselves sets of requests) are considered neighbors if there are two routes within them that both differ from one another by at most one request. Another way to think of this is by constructing a neighbor from a given solution. To construct a neighbor, select two routes from the solution, call them R_0 and R_1. There are two allowable moves to construct a neighbor: move some request from R_0 to R_1, or move some request from R_0 to R_1 and another request from R_1 to R_0. We call these operations request interchanges.

The Tabu search algorithm can be found in Algorithm 5. From the perspective of the Tabu list, the forbidden structure would be the routes from the previous iteration that differ from the routes of the current solution. That is, forbidding neighbors from having the same clustering of the requests as the previous solution. The stopping criterion for the Tabu search was failing to find an improving solution after a certain number of iterations.

4.4.3 An Example

We illustrate the online route building approach and the Tabu search with the following example. For the sake of simplicity, we will ignore the time-dependent travel costs, and assume simply that each edge has a travel cost and length of 1. Suppose the following is the request schedule, given as O – D pairs:

- R_0: 9 – 4, $s_0 = 0, e_0 = 2$
- R_1: 1 – 6, $s_1 = 0, e_1 = 2$
- R_2: 4 – 9, $s_2 = 4, e_2 = 6$
- R_3: 6 – 1, $s_3 = 4, e_3 = 6$

Algorithm 5: Function to refine routes by tabu search.

```
1  function TabuRefinement $R = \{R_1, R_2, ..., R_{|\mathcal{K}|}\}$;
   Input  : A set of routes $R$.
   Output: A set of routes bestRoutes with cost less than or equal to the cost of
           initial routes.
2  initialization;
3  bestRoutes ← $R$; // current best set of routes
4  tabuList ← {}; // short-term memory of forbidden exchanges
5  listTime ← {}; // iteration at which each tabu item was added to list
6  while iterationCount < iterationLimit do
7      bestSavings ← $\infty$;// best savings to be found from a request exchange
        bestExchange ← $((0,0),(0,0))$; // best ((request, route),(request, route)) pair
        so far
8      foreach $R_i \neq R_j$ do
9          foreach $x \in R_i, (x, R_j) \notin$ tabuList do
10             foreach $y \in R_j, (y, R_i) \notin$ tabuList do
11                 Compute route for $R'_i = R_i \cup \{y\} \setminus \{x\}, R'_j = R_j \cup \{x\} \setminus \{y\}$;
12                 if cost($R'_i$) + cost($R'_j$) - cost($R_i$) - cost($R_j$) < bestSavings then
13                     bestSavings ← cost($R'_i$) + cost($R'_j$) - cost($R_i$) - cost($R_j$);
                        bestExchange ← $((x, R_i), (y, R_j))$;
14                 end
15             end
16         end
17     end
18     Perform bestExchange;
19     tabuList.push_back($(x, R_i)$);
20     tabuList.push_back($(y, R_j)$);
21     if cost(R) < cost(bestRoutes) then
22         bestRoutes ← $R$;
23         tabuList ← {};
24         iterationCount ← 0;
25     end
26 end
27 return bestRoutes;
```

Suppose we have the road network given in Figure 4.1.

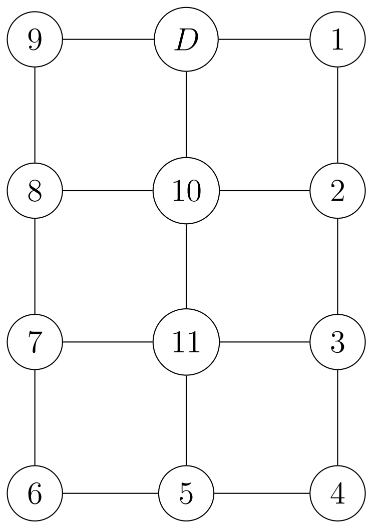


Figure 4.1: Road network for constructive and Tabu search example.

Given two vehicles with sufficient maximum travel distance and capacity, the optimal routes are

- V_0: $D - 1^1 - 2 - 3 - 4^2 - 5 - 6^1 - 7 - 8 - 9^2 - D$
- V_1: $D - 9^0 - 8 - 7 - 6^3 - 5 - 4^0 - 3 - 2 - 1^3 - D$

with blue superscripts indicating pickups and red superscripts indicating drop offs. The travel time of each route is 10 for a total cost of 20. Consider what the initially constructed routes would be.

When the first two requests come in, the initial routes could be

- V_0: $D - 9^0 - D - 1 - 2 - 3 - 4^0 - 3 - 2 - 1 - D$
- V_1: $D - 1^1 - D - 9 - 8 - 7 - 6^1 - 7 - 8 - 9 - D$

At the time requests 2 and 3 come in, V_0 would be at node 2 and V_1 would be at node 8, with their remaining routes being

- V_0: $2 - 3 - 4^0 - 3 - 2 - 1 - D$
- V_1: $8 - 7 - 6^1 - 7 - 8 - 9 - D$

In the without look-ahead approach, the best option is to assign V$_0$ to satisfy request 2 and V$_1$ to satisfy request 3. This results in the overall routes

- V$_0$: $D - 9^{0} - D - 1 - 2 - 3 - 4^{02} - 3 - 2 - 1 - D - 9^{2} - D$
- V$_1$: $D - 1^{1} - D - 9 - 8 - 7 - 6^{13} - 7 - 8 - 9 - D - 1^{3} - D$

In this case, each route has a cost of 12, for a total cost of 24 (20% higher than optimal). Each route can also be considered as a set of requests, with V$_0$'s route being $\{0, 2\}$ and V$_1$'s route being $\{1, 3\}$. Note that all (reasonable) routes that satisfy $\{0, 2\}$ have a cost of 12. In this example, the look-ahead solution presented would construct optimal routes because, in a sense, this approaches allows decisions made in the past to be changed.

In the Tabu search phase, we search for other ways of clustering the requests. The initial routes as sets again are

- V$_0$: $\{0, 2\}$
- V$_1$: $\{1, 3\}$

The first request interchange considered would be

- V$_0$: $\{0, 3\}$
- V$_1$: $\{1, 2\}$

Consider building V$_0$'s route. Since it is not possible to pick up and drop off R$_0$ and still get to R$_3$ (R$_3$ is only willing to wait until time $t = 6$ to be picked up), R$_3$ must be picked up before R$_0$ is dropped off. Since drop offs are done in a first in, first out fashion, the outline of V$_0$'s route must be:

1. Pick up R$_0$
2. Pick up R$_3$
3. Drop off R$_0$
4. Drop off R$_3$

This will result in the overall route being

- V_0: D – 9^0 – 8 – 7 – 6^3 – 5 – 4^0 – 3 – 2 – 1^3 – D

which has a cost of 10. Similarly, we will get V_1's route to be

- V_1: D – 1^1 – 2 – 3 – 4^2 – 5 – 6^1 – 7 – 8 – 9^2 – D

These routes are the optimal routes.

4.5 Computational Experiments

Recently, Greenville, SC has been awarded a Federal Highway Administration (FHWA) grant to deploy automated taxi shuttle systems in three areas in the county [20]. In order to effectively implement these systems, we analyzed AMD deployment scenarios for the Greenville, SC network, which is illustrated in Figure 4.2. This network, together with the corresponding request data, represent the bulk of the computational experiments performed.

The overall network configuration and origin-destination trip requests were obtained from the regional travel demand model in the Greenville-Pickens Area Transportatoin Study (GPATS). This study contains datasets on origin-destination trip requests for four periods throughout the day: (i) AM Peak, which is from 6:01 to 9:00, (ii) Mid-Day, which is 9:01 to 16:00, (iii) PM Peak, 16:01 til 19:00, and (iv) Night Time, covering 19:01 through 6:00. The AM Peak dataset contains information on 378 trips, covering the following modes of transportation: (i) on-demand fixed-route automated shuttle service (20%), (ii) on-demand door-to-door automated shuttle service (30%), (iii) walking (10%), (iv) regular vehicle traffic (40%). The origin-destination data from (i) and (ii) were combined to form the baseline on-demand requests for our scenarios. After data processing and cleansing, 177 on-demand requests distributed across the network were used. The distribution of the origins of these requests can be found in Figure 4.3, while the distribution of the destinations of these requests can be found in Figure 4.4.

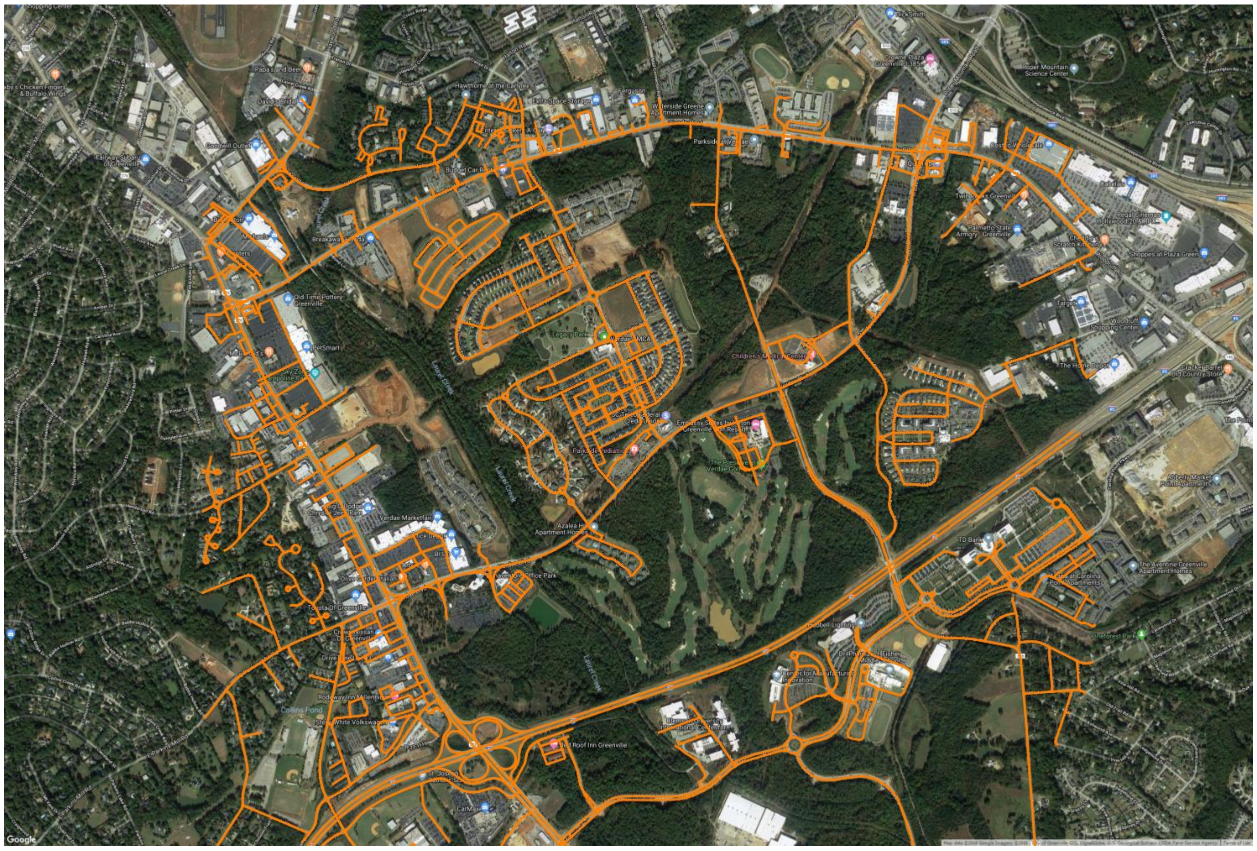


Figure 4.2: Test network from Greenville, SC. The corresponding graph contains 554 nodes and 1340 edges.

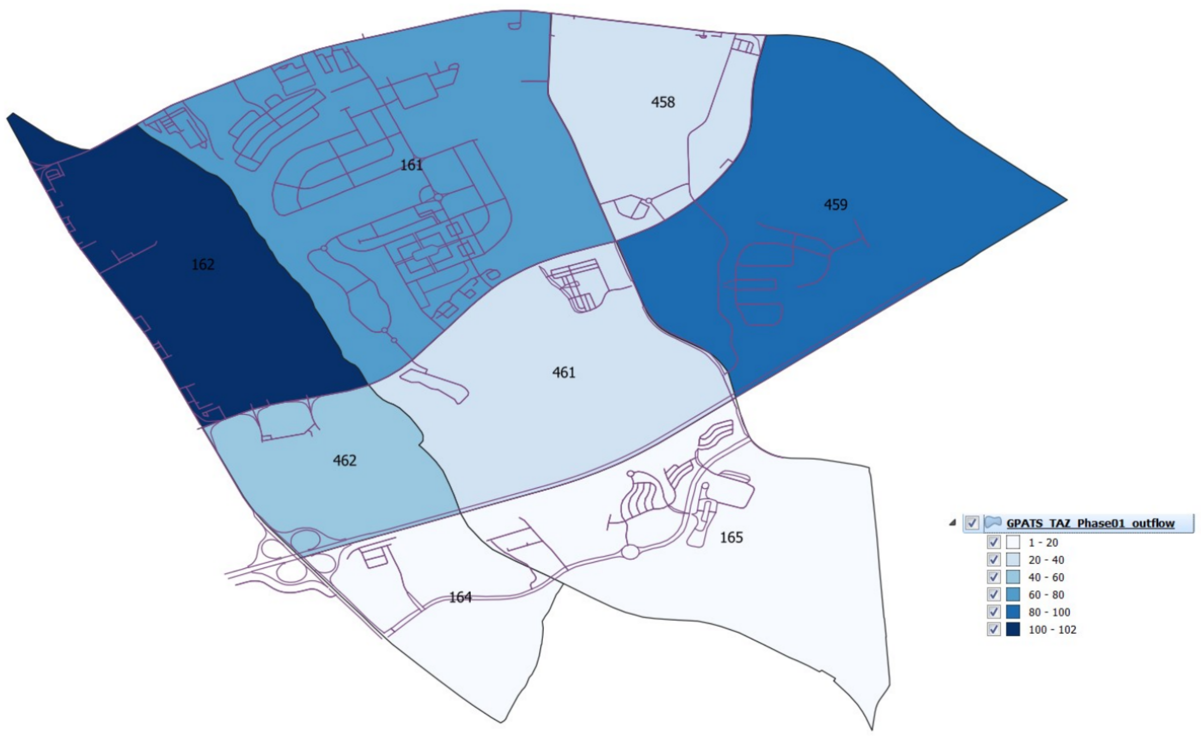


Figure 4.3: Distribution of origins of requests within the network. Lighter colored regions correspond to fewer requests, while darker regions correspond to more requests.

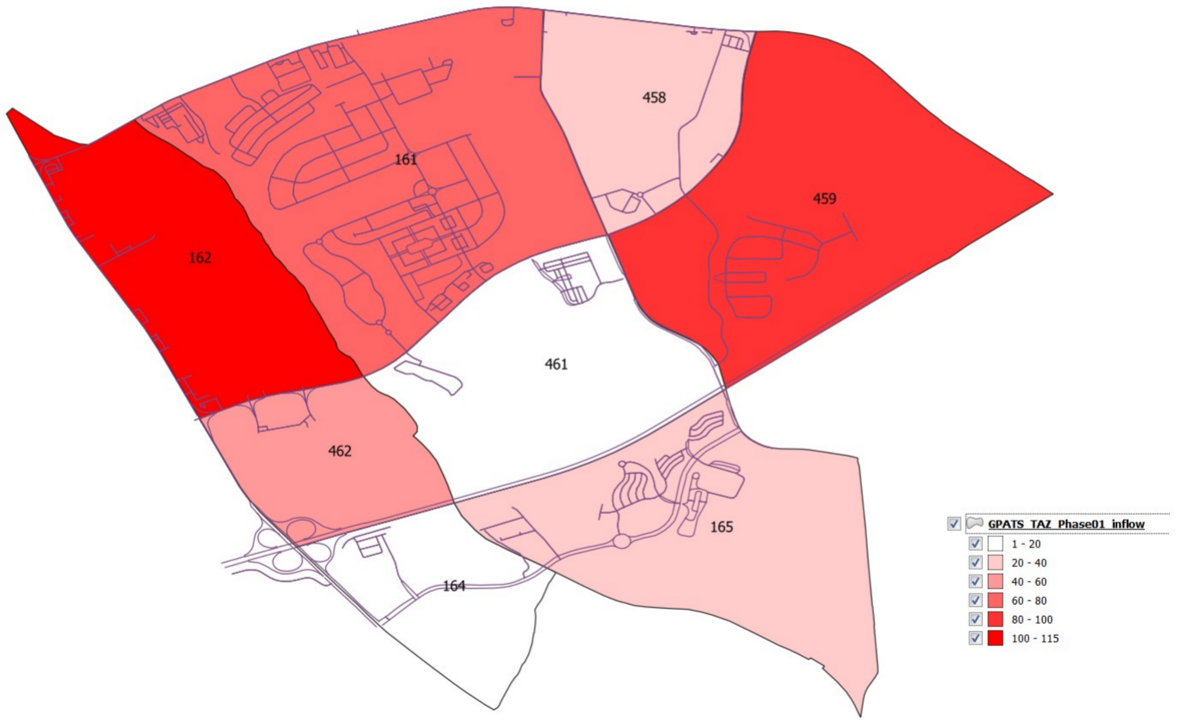


Figure 4.4: Distribution of destinations of requests within the network. Lighter colored regions correspond to fewer requests, while darker regions correspond to more requests.

In addition to this baseline setup, two additional demand levels were also considered. In one of these, 25% of the requests were chosen at random to be ignored in order to simulate fewer requests. This resulted in an instance with 134 requests. In the other, 10% of the requests were chosen at random to be duplicated while having the duplicate's request time chosen at random in order to simulate a higher demand, which resulting in 194 requests.

For the road segment parameters, we note that the road segments were given as a series of latitude and longitude points in a Shapefile. Using a linear interpolation of the points, the length of each road segment was computed.

In order to determine the travel time across the road segments of the network, the open source traffic simulation tool SUMO [30] was used at a resolution of 5 minutes. These were then used to estimate travel times on a 15-minute scale, which is what was ultimately used as the travel times. In the event SUMO failed to produce a travel time along a road segment, it was assumed traffic would move at 90% of the speed limit for that segment, and the travel time implied by this was used. Since the total time horizon considered was 180 minutes, 12 time periods of 15 minutes each were used.

In addition to travel times and distances, the energy required for each route was also a metric of interest because the automated shuttles are expected to be electric vehicles. The energy required to traverse a road segment is a function of the length of the road segment and the travel time, with longer road segments requiring more energy, and slower travel times also requiring more energy. The specific values were developed by using the Future Automotive Systems Technology Simulator (FASTSim), which is an open-source vehicle powertrain analysis model for light-duty vehicles [10]. Using FASTSim and the characteristics of a popular automated electric shuttle EasyMile EZ10. Figure 4.5 shows the distribution of average speeds along the road segments as well as the energy consumption at each of those speeds.

We note that the model proposed also accounts for passenger loading/unloading times, as well as the maximum amount of time passengers are willing to wait for a shuttle to arrive. Passenger loading and unloading times had three levels that were considered: 30 seconds, 60 seconds, and 90 seconds. Similarly, the amount of time allotted to fulfill a request had three levels: 60 seconds, 120 seconds, and 180 seconds.

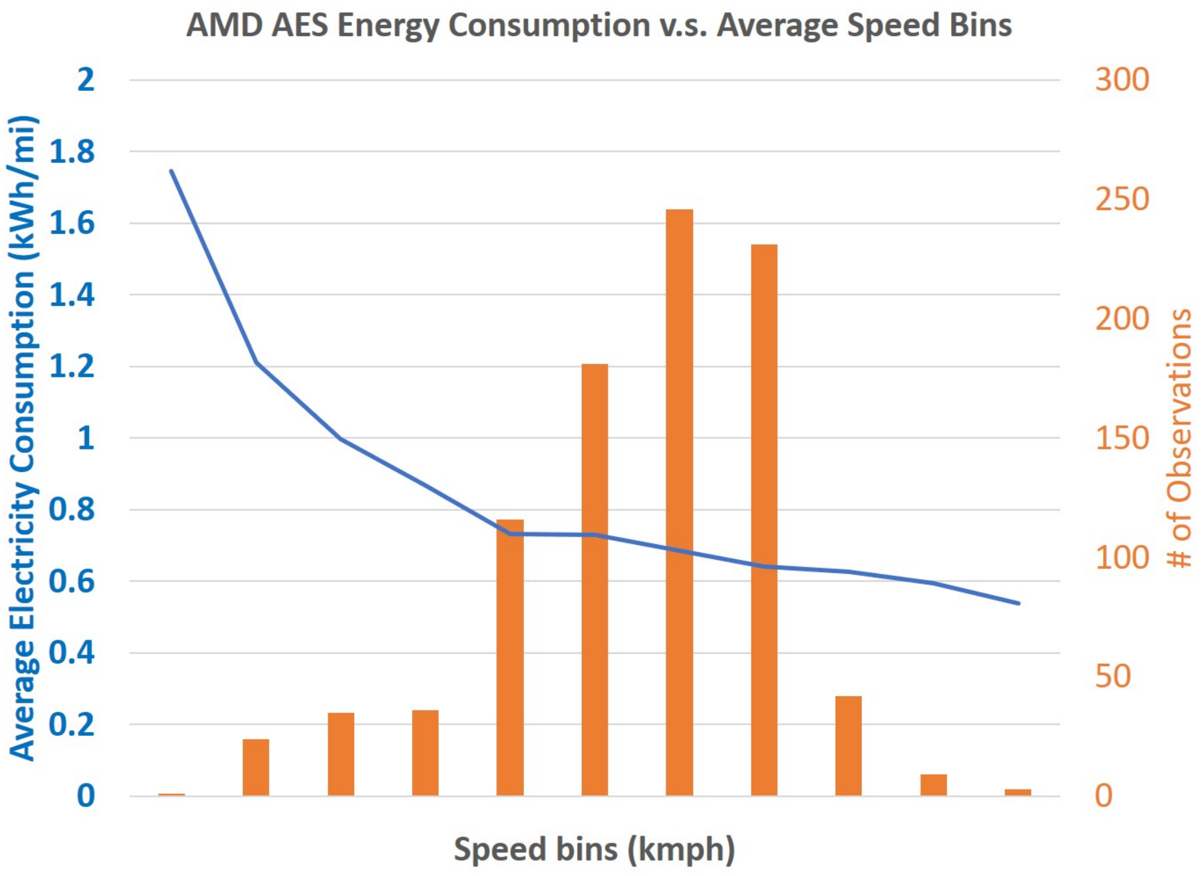


Figure 4.5: Distribution of average travel speeds on road segments together with energy consumption at those speeds.

From an operational perspective, most (if not all) of the characteristics previously mentioned will not be easily controlled by an operating entity. There are two primary aspects that can be readily addressed: (i) routing, and (ii) fleet composition. For this reason, in addition to comparing the three solution approaches mentioned in Section 4.4, sensitivity analysis on the vehicle characteristics was also performed. The two characteristics were examined: (i) maximum vehicle travel distance, and (ii) passenger capacity of each vehicle.

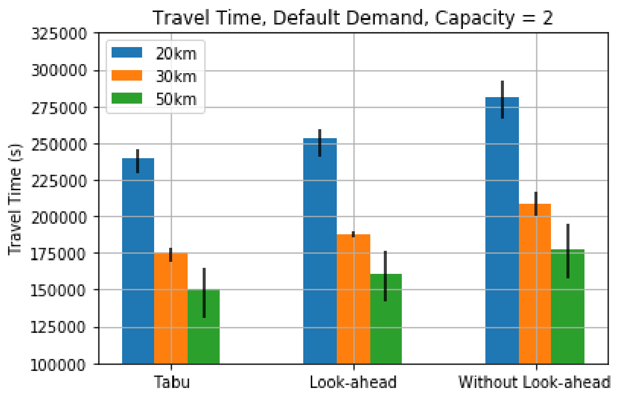
As the MIP model presented allows for rather generic objective functions, total travel time of the fleet of vehicles was used for our experiments. This was done to attempt to find approximate the concerns most operators would have with routing the individual vehicles. Travel times should be short because (i) overall system utilization, and therefore customer satisfaction, is important, and (ii) having the vehicles on the road for less time tends to cost less money.

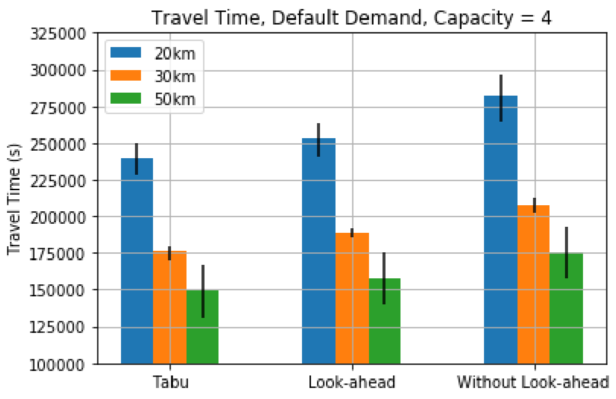
We note that computational results using the MIP model are unavailable. This is due to the size of the model resulting from the Greenvills, SC data. While we were unable to determine the exact size of the model, our machine, running Ubuntu 18.04, with 256 GB of RAM, lacked the memory to be able to fully build the model. As a result, we do not present exact solutions, nor do we present data on the optimality gap of the various solution techniques.

All experiments were done using a 16 core (32 thread) machine, running Ubuntu 18.04, with 256 GB of RAM. All coding was done in C++. Standard graph algorithms used were implementations from Boost's Graph Library [9].

4.6 Results

Figures 4.6, 4.7, 4.8 contain the travel time results. As would be expected, having vehicles with longer ranges decreases total travel time, and having vehicles with higher capacities also decreases total travel time. Across the different strategies, Tabu search on average outperformed the with look-ahead strategy by 10% and the without look-ahead strategy by 16%.





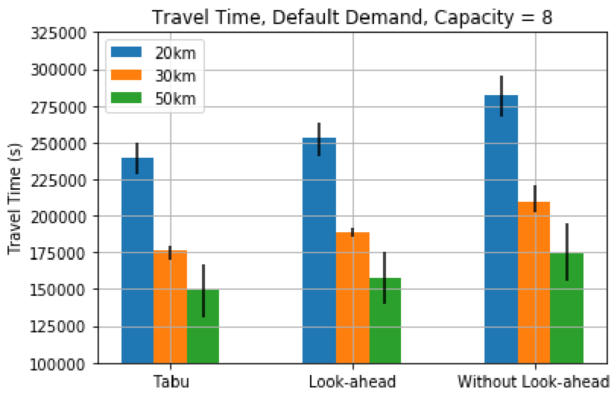


Figure 4.6: Average objective value of best found solution under various strategies in scenarios using default request level. Black bars indicate spread of best and worst scenarios.

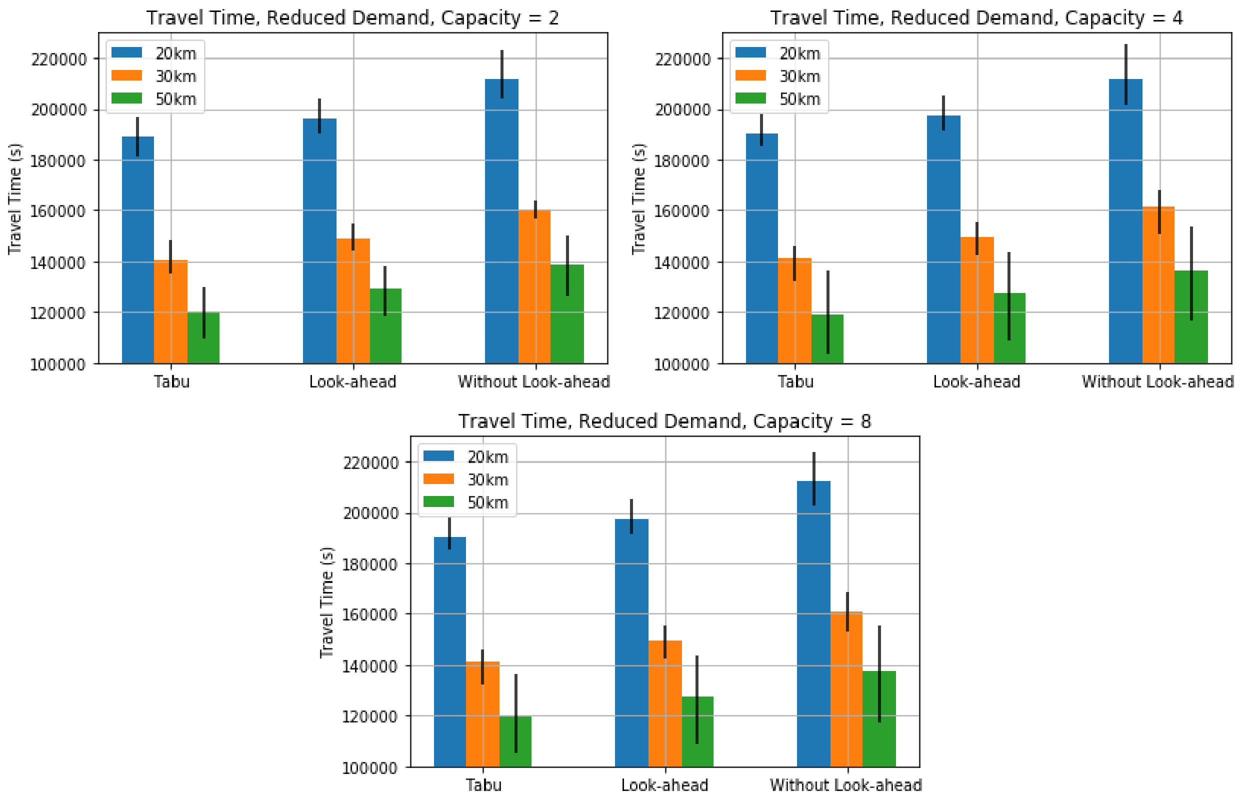


Figure 4.7: Average objective value of best found solution under various strategies in scenarios using reduced request level. Black bars indicate spread of best and worst scenarios.

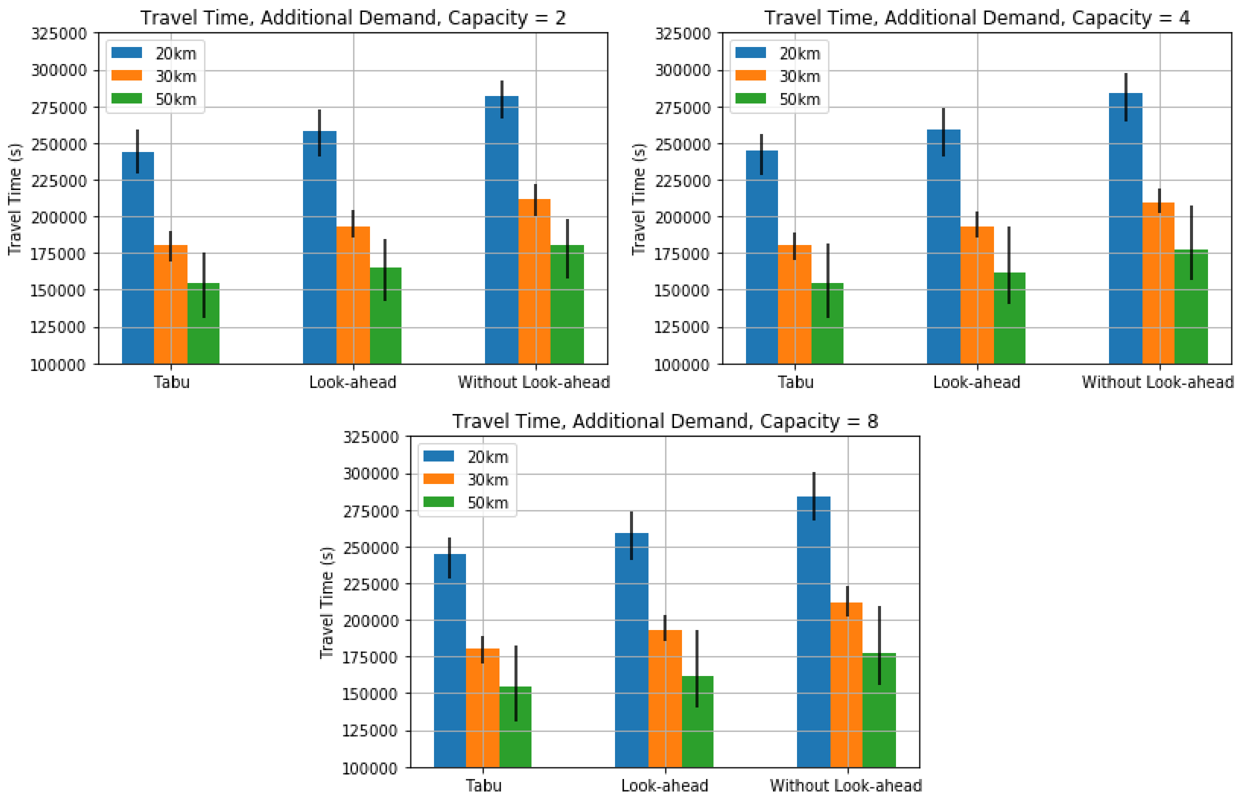


Figure 4.8: Average objective value of best found solution under various strategies in scenarios using additional request level. Black bars indicate spread of best and worst scenarios.

When considering just the effects of maximum distance on the solutions, we note that there was often a significant improvement in total travel time when going from a 20km vehicle range to 30km, with some instances seeing a nearly 30% reduction in total travel time. The difference was markedly less pronounced when going from a range of 30km to 50km, however, despite the relative difference in ranges being larger in this gap. At some point, having vehicles with higher ranges will not improve solution quality for any vehicle routing problem. We begin to see these diminishing returns when moving to the 50km range for these instances.

The effect of passenger capacities on total travel time was less pronounced, averaging between 0% and 2% across the instances. While several instances saw a change in solution quality when going from a capacity of 2 passengers to 4 passengers, very few saw any changes at all when going from 4 passengers to 8 passengers.

The effects of vehicle capacity on the number of vehicles required saw a similar story. As the capacity increased from 2 to 4, there were often a reduction in the number of vehicles required, but never more than a few percent. When increasing the capacity from 4 to 8, though, few instances saw any change at all. This can be seen in Figure 4.12.

As before, vehicle range played a large role in the amount of vehicles needed, with higher rangers corresponding to requiring fewer vehicles to satisfy all of the requests. And again, we see the largest difference when moving from vehicles with a range of 20km to a range of 30km. On average, this larger range allows all the requests to be satisfied using nearly 30% fewer vehicles. As was the case before, the relative difference between the 30km and 50km ranges were significantly less.

The energy consumption of the routes provided the most surprising results. These can be found in Figures 4.9, 4.10, and 4.11. Typically, the routes produced by the Tabu search were more energy efficient, in terms of energy per vehicle mile, than the other strategies, but this was not uniformly the case as it was with travel times. In many instances, the Tabu search actually produced routes which were less energy efficient than the other strategies. The same trend held when considering the effects of vehicle range on the energy efficiency. As the vehicle range increased, the routes found required less time, but the energy consumption per mile increased.

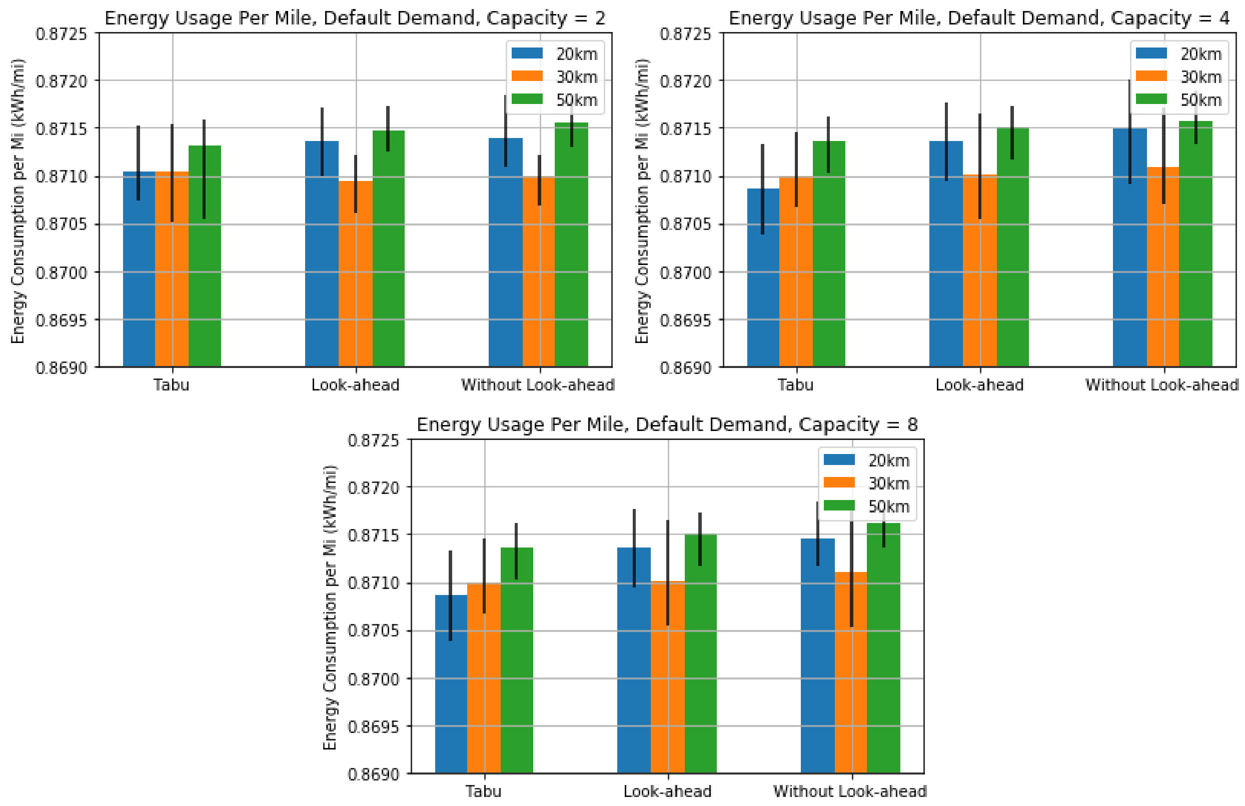


Figure 4.9: Average energy consumption per mile (kWh/mi) of best found solution under various strategies in scenarios using default request level. Black bars indicate spread of best and worst scenarios.

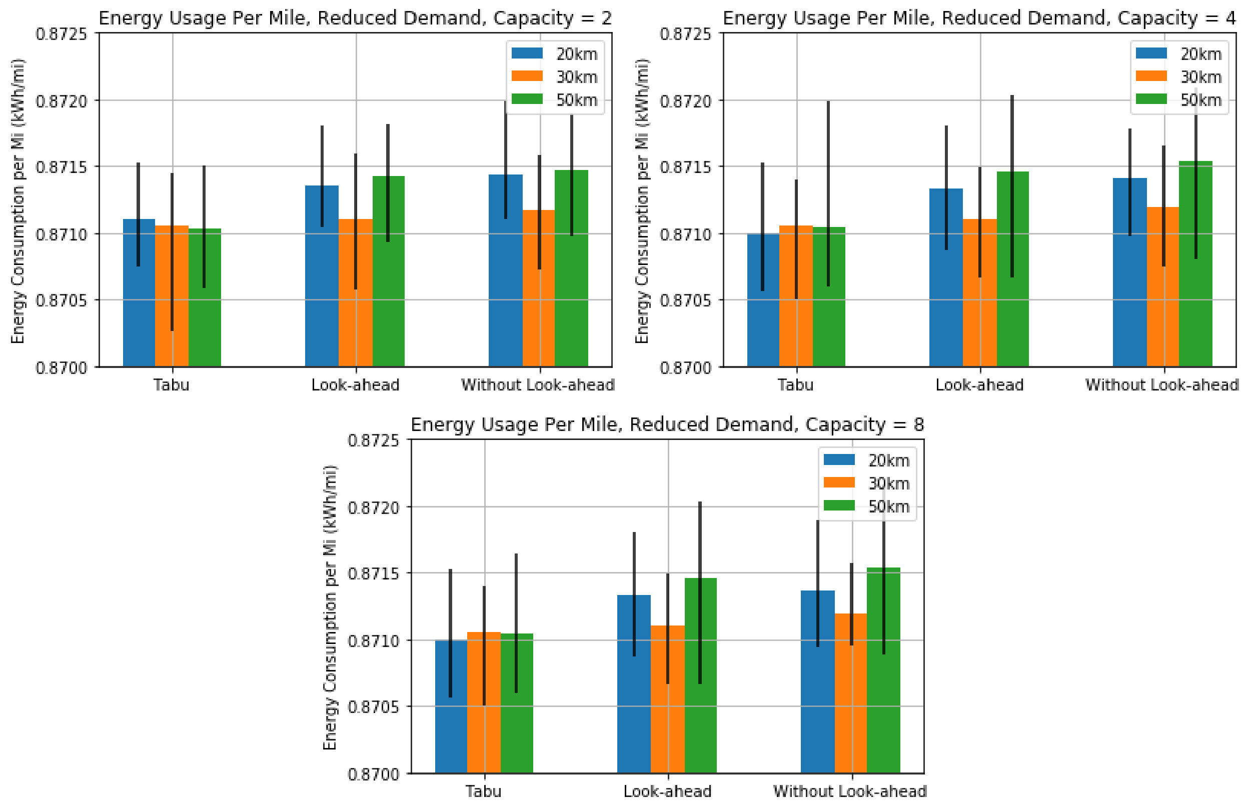


Figure 4.10: Average energy consumption per mile (kWh/mi) of best found solution under various strategies in scenarios using reduced request level. Black bars indicate spread of best and worst scenarios.

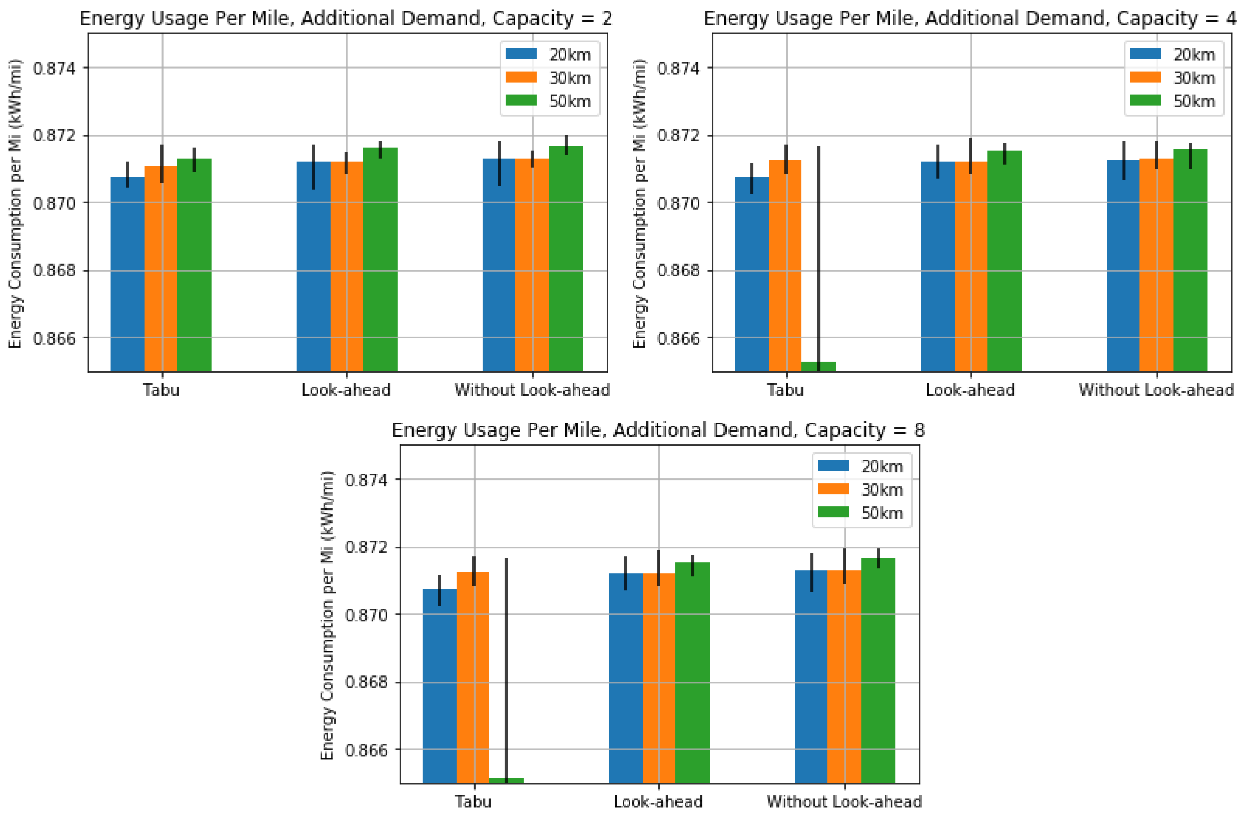


Figure 4.11: Average energy consumption per mile (kWh/mi) of best found solution under various strategies in scenarios using additional request level. Black bars indicate spread of best and worst scenarios.

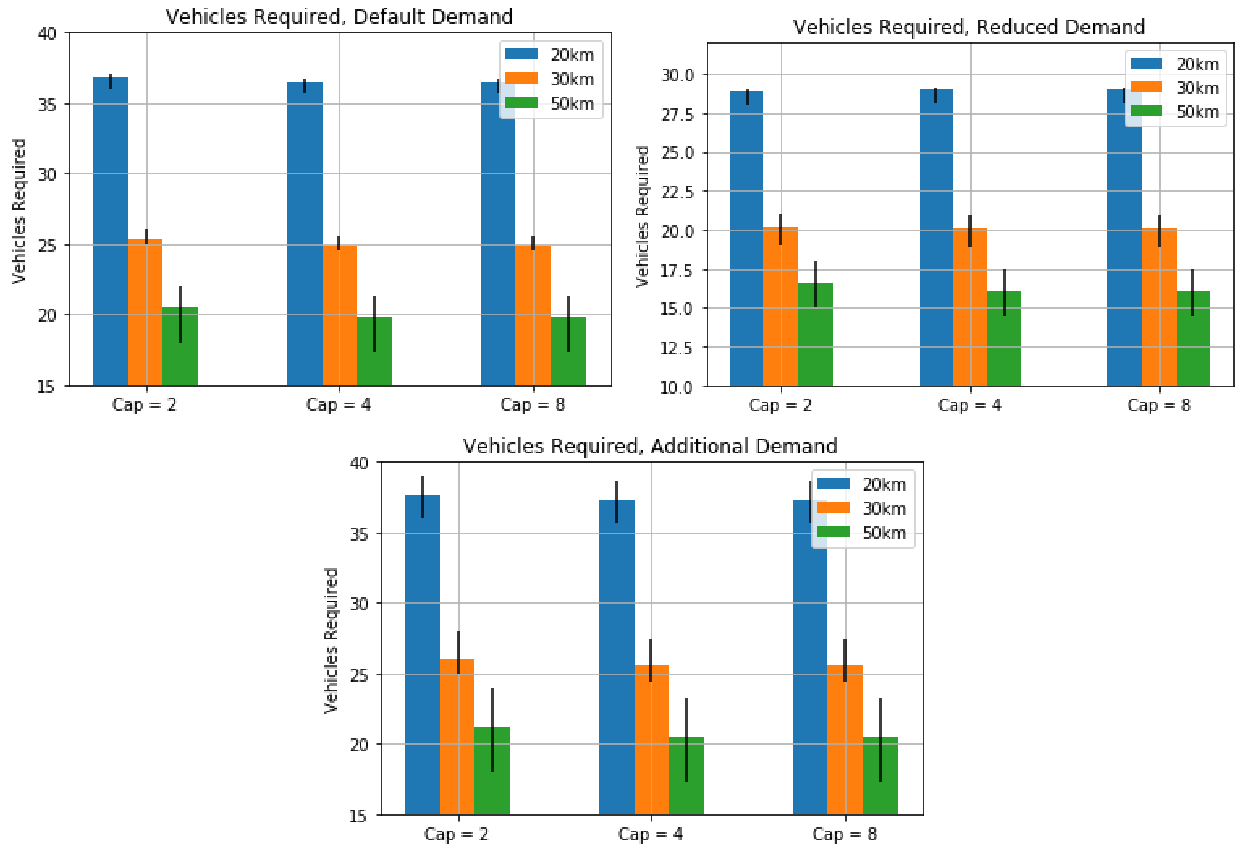


Figure 4.12: Average minimum number of vehicles required to be able to satisfy all requests with without look-ahead initial route construction in each demand level. Black bars indicate spread of best and worst scenarios.

This can seem especially paradoxical when considering the information contained in Figure 4.5. Since the Tabu search is looking for routes which require less time, it seems reasonable that these would also be routes that have several road segments which can be traversed at a high speed. As seen in Figure 4.5, the higher the speed, the lower the energy consumption. So it would seem reasonable that the Tabu search should produce routes which consume less energy per vehicle mile. The same trend, under this reasoning, would also be expected from increasing the vehicle range.

One possible mechanism for explaining this apparent discrepency is hinted at by Figures 4.4 and 4.3. Several of the requests are to essentially go across the network. Towards both the beginning and end of these trips, many of the road segments are neighborhood or otherwise congested streets, while in between them lie highways and road segments with a higher throughput. In many of these scenarios, the most energy-efficient leg of the journey is when after the vehicle has picked up all passengers and before it begins to drop them off. With higher ranges, more of these passengers can be picked up in each vehicle, thus reducing the number of trips across the network that are required to satisfy these requests. What is not reduced, however, is the amount of vehicle miles that must be spent while active picking up or delivering passengers, which is the less energy-efficient aspect of the journey. A illustrative example of this is given below.

Suppose the graph given in Figure 4.13 is the road network, where each link has length 1 mile, traffic on the blue links travel at 10km/h, and traffic on the red links travel at 20km/h. That is, the travel time on the blue links is 0.161 hr, and the travel time on the red links is 0.0805 hr, and the energy cost to traverse a blue link is 1.21 KWh, and the energy cost to traverse a red link is 0.867 KWh. Suppose D is the vehicle depot (where the vehicles start and stop their routes), and there are two requests: $(A) - (Z)$ and $(C) - (X)$.

Now suppose each vehicle has a maximum distance of 8 miles. Then the only set of feasible routes is:

- $D - B - A - B - D - Y - Z - Y - D$
- $D - B - C - B - D - Y - X - Y - D$

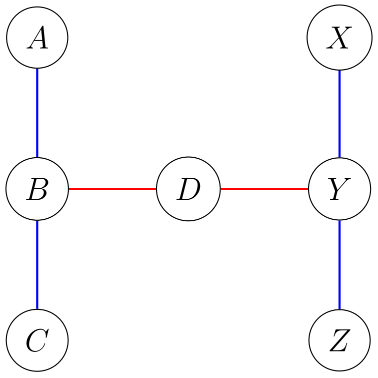


Figure 4.13: Example network to show quicker routes may be less energy efficient.

The total travel time is then $8\cdot(0.161+0.0805)$ hr $= 1.93$ hr, while the total energy used is $8\cdot(1.21+0.867)$ KWh $= 16.6$ KWh. Thus the average energy per mile is 16.6 KWh/16 mi $=$ 1.04 KWh/mi.

Suppose now that each vehicle has a maximum distance of 12 miles. A new route becomes feasible which satisfies both requests with a single vehicle.

- $D - B - A - B - C - B - D - Y - Z - Y - X - Y - D$

For this new route, the total travel time is $8 \cdot 0.161 + 4 \cdot 0.0805$ hr $= 1.61$ hr, and the total energy used is $8 \cdot 1.21 + 4 \cdot 0.867$ KWh $= 13.1$ KWh. Note that this route requires less vehicle travel time, and is improving with respect to the objective function used. For this scenario, the average energy per mile is 13.1 KWh/12 mi $= 1.09$ KWH/mi.

4.7 Conclusion

We have developed a mathematical model that can be used to route on-demand shuttles within an automated mobility district. We have also shown that techniques that have been proven to work well for many vehicle routing problems can also be applied to this problem. We have also shown that using a Tabu search can offer significant savings in terms of vehicle travel time when compared to on-the-fly route generation. Because this approach requires a priori knowledge of the demand distribution, this suggests that significant savings can be achieved with high-quality models and data that can be used to forecast demand and overall traffic patterns within an AMD.

Further, we have shown that these approaches can be used at-scale on city-wide instances for not only routing and day-to-day planning, but also in the context of overall fleet composition planning. Our results from Greenville, SC suggest that having moderate-range automated electric shuttles (approximately 30km range) with a passenger capacity similar to that of a typical sedan (4 people) offers many of the benefits seen from larger, more expensive automated shuttles. This is desirable because batteries often represent a significant portion of the costs of electric vehicles, and both the size of the vehicle and the maximum range of the vehicle directly affect the battery storage needed.

Lastly, while not directly related to automated mobility districts, we have also shown an approach that can be used on other vehicle routing problems to model travel times and costs changing as the temporal aspect of the problem evolves.

Chapter 5

Conclusions

In this chapter we conclude the thesis and offer directions on future research.

5.1 Allocating Deicing Resources

Chapters 2 and 3 present a framework for designing winter deicing maintenance strategies in a data-driven and results-oriented way. Chapter 2 focuses on developing the model and a constructive solution approach, while Chapter 3 develops a framework in which the approaches in Chapter 2 can be used to budget for deicing activities. While the developed solution techniques offer improvements over currently existing strategies, we note that simply building the corresponding MIP model and attempting to solve it with an off-the-shelf solver, such as CPLEX or Gurobi, offered better solutions than our constructive heuristic. This was often the case even in the event of a timeout on the part of the solver. While we still feel these contributions are important as they allow resource-lacking municipalities to be able to develop approaches that do not rely on expensive commercial software licenses, there are still modifications that may be useful to consider to improve the presented approach.

One immediate consideration that may be made to the heuristic approach presented in Chapter 2 is in the first phase when the priority queue is built. Currently, while we are cognizant of the fact that the edges treated to improve the first-stage flow will impact the marginal costs of the second-stage flow, our approach does not account for the fact that the reverse is also true: the edges treated to improve the second-stage flow will impact that marginal costs of the first-stage flow. Perhaps because of this interplay, there were some instances in which the heuristic failed at finding paths to route additional traffic, despite the fact that there were some paths available that could be treated given the resource levels in that instance. It seems apparent that this interplay has the capability to impact solution quality.

With this in mind, an obvious fix is to not only update the second-stage dual graph with the information from the first-stage shortest path, but also to update the first-stage dual graph with the information from the second-stage shortest path. Naturally, some questions immediately come to mind. First, note that if the first-stage dual graph is changed, this will likely change the first-stage shortest path. This, in turn, will change the second-stage dual graph, which could change the second-stage shortest path. However, we are attempting to update the first-stage dual graph by appealing to the second-stage shortest path. One way to ensure these changes are limited could be by fixing certain aspects of the path chosen. However, this could become computationally expensive. Are there other ways of addressing this?

A second concern would be that of convergence criteria. Since the marginal costs of the edges will not be monotonically increasing or decreasing under these updates, it is not readily apparent that performing these iterations will result in first- and second-stage shortest paths and dual graphs which eventually converge. What additional work must be done to ensure that an iterative approach such as this will converge?

An additional concern that should be addressed is the way the traffic is currently modeled. At the moment, the objective is to maximize the throughput of the network. Perhaps a better metric to consider would be the likelihood of traffic accidents (which would be minimized, naturally). Since traffic accidents tend to be correlated with traffic flows, with more traffic corresponding to fewer accidents, this metric may introduce non-linearities within the problem which will need to be accounted for.

5.2 Automated Mobility Districts

Chapter 4 considered routing automated shuttles within an automated mobility district as well as planning the fleet of vehicles used. We designed a mixed integer programming model which, in theory, is capable of routing the fleet of vehicles. We were also able to show the impact of various vehicle parameters on the quality of solutions obtained, as well as the size of the fleet that would be needed to satisfy the requests given the various parameters.

One aspect of the current approach that could be improved is the size of the current model. We note that the usage of the phase "in theory" in the previous paragraph was intentional, as any naive implementation of the provided mathematical model for an instance on the scale of a city seems unlikely to be solvable given current hardware limitations. It seems vanishingly unlikely, then, that current software approaches would be able to make much headway on instances like this, even if they were able to fit in memory. Because of this, additional work must be done, likely either through reformulations and/or decomposition techniques.

One potentially promising avenue was somewhat utilized and alluded to in the text. It was mentioned that it was often useful to think of a route as a set of requests to be fulfilled, rather than a series of nodes visited along with the corresponding pickups, deliveries, and times at which the visits occur. With this in mind, perhaps a better exact technique would be to think of this problem as more of a clustering (or graph coloring) problem, where the clusters represent the clustering of the requests. In this framework, it seems likely that considerable decompositions could be achieved since the resources necessary to compute the objective value of any given route is relatively low and the routes readily decompose individually, allowing large swathes of the variables and constraints to be ignored.

In addition to this, we note that the instances from Greenville, SC were built utilizing both on-demand requests as well as the requests on the fixed-route shuttles. The current approach does not account for fixed-route shuttles. However, practically (if not actually) every municipality which provides some form of public transportation, AMD or otherwise, utilizes fixed routes. Modifying the presented model to account for this seems like a reasonable next step.

5.3 Future Work

In addition to the questions asked in the previous sections, we note that one aspect of these problems that became apparent is the need for high-quality data to be able to properly utilize these techniques. In the context of AMDs, being able to reliably forecast future demand and route vehicles accordingly saw significant savings in terms of on-the-road travel time for the vehicles. Further, in order to utilize our deicing resources more effectively, it is imperative that we are able to forecast weather and traffic patterns in a reliable manner.

Bibliography

[1] (2006). *A Stochastic Optimization Model for Highway Project Selection and Programming under Budget Uncertainty*. 43

[2] (2017). *Modeling Road Vulnerability to Snow Using Mixed Integer Optimization*. IISE Annual Conference and Expo. 41, 42, 43, 44, 50

[3] Adams, E. (2017). How Long, Really, Until Self-Driving Cars Hit the Streets? - The Drive. 56

[4] Albert, R. and Barabási, A.-L. (2002). Statistical mechanics of complex networks. *Rev. Mod. Phys.*, 74:47–97. 30, 50

[5] Applegate, D. L., Bixby, R. E., Chvatal, V., and Cook, W. J. (2007). *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*. Princeton University Press, Princeton, NJ, USA. 5

[6] Asamer, J. and Reinthaler, M. (2010). Estimation of road capacity and free flow speed for urban roads under adverse weather conditions. In *13th International IEEE Conference on Intelligent Transportation Systems*, pages 812–818. 11

[7] Baldacci, R., Toth, P., and Vigo, D. (2010). Exact algorithms for routing problems under vehicle capacity constraints. *Annals of Operations Research*, 175(1):213–245. 5

[8] Bertsimas, D. and Tsitsiklis, J. (1997). *Introduction to Linear Optimization*. Athena Scientific, 1st edition. 2

[9] Boost (2018). Boost C++ Libraries. http://www.boost.org/. Last accessed 12JUN2018. 29, 76

[10] Brooker, A., Gonder, J., Wang, L., Wood, E., Lopp, S., and Ramroth, L. (2015). FASTSim: A Model to Estimate Vehicle Efficiency , Cost and Performance. *SAE Technical Paper*, (April):21–23. 74

[11] Christofides, Nicos (1976). The vehicle routing problem. *R.A.I.R.O. Recherche opérationnelle*, 10:55–70. 5

[12] Clarke, G. and Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4):568–581. 5, 26

[13] Cordeau, J.-F., Laporte, G., Savelsbergh, M., and Vigo, D. (2007). *Vehicle Routing*, volume 14, pages 195–224. 5

[14] Dantzig, G., Fulkerson, R., and Johnson, S. (1954). Solution of a large-scale traveling-salesman problem. *Journal of the Operations Research Society of America*, 2(4):393–410. 2, 3

[15] Dantzig, G. B. and Ramser, J. H. (1959). The truck dispatching problem. *Management Science*, 6(1):80–91. 3, 4

[16] Desrochers, M. and Verhoog, T. (1991). A new heuristic for the fleet size and mix vehicle routing problem. *Computers & Operations Research*, 18(3):263 – 274. 6

[17] et al., J. J. H. (1991). Highway deicing - comparing salt and calcium magnesium acetate. Technical Report 235, Transportation Research Board. 9, 31, 40

[18] et al, J. R. (2010). Maine winter roads: Salt, safety, environment and cost. Technical Report 10-06, Margaret Chase Smith Policy Center, The University of Maine. 9

[19] Fagella, D. (2017). Self-Driving Car Timeline for 11 Top Automakers. 56

[20] FHWA (2017). Fhwa awards $4 million grant to south carolina's greenville county for automated taxi shuttles. *Press Release: FHWA Awards $4 Million Grant to South Carolina's Greenville County for Automated Taxi Shuttles, 10/4/2017, Federal Highway Adminstration*. 56, 70

[21] Fisher, M. L. and Jaikumar, R. (1981). A generalized assignment heuristic for vehicle routing. *Networks*, 11(2):109–124. 6

[22] Fulkerson, D. R. (1959). Increasing the capacity of a network: The parametric budget problem. *Management Science*, 5(4):472–483. 19, 21, 49, 51

[23] Ghiani, G., Improta, G., and Laporte, G. (2001). The capacitated arc routing problem with intermediate facilities. *Networks*, 37(3):134–143. 9, 41

[24] Glover, F. (1989). Tabu search—part i. *ORSA Journal on computing*, 1(3):190–206. 7, 63, 65

[25] INFORMS (2016). Ups 2016 edelman. 4

[26] John R. Birge, F. L. (2011). *Introduction to Stochastic Programming*. Springer. 43

[27] Karmarkar, N. (1984). A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395. 2, 20

[28] Karp, R. M. (1972). *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, Boston, MA. 2

[29] Kinable, J., van Hoeve, W.-J., and Smith, S. (2016). Optimization models for a real-world snow plow routing problem. In *Integration of AI and OR Techniques in Constraint Programming*, pages 229–245. 9, 12

[30] Krajzewicz, D., Bonert, M., and Wagner, P. (2006). The open source traffic simulation package sumo. *RoboCup 2006 Infrastructure Simulation Competition*, 1:1–5. 74

[31] Krokhmal, P., Palmquist, J., and Uryasev, S. (2002). Portfolio optimization with conditional value-at-risk objective and constraints. *Journal of risk*, 4:43–68. 43

[32] Laporte, G. (1986). Generalized subtour elimination constraints and connectivity constraints. *The Journal of the Operational Research Society*, 37(5):509–514. 5, 59, 62

[33] Laporte, G. (2009). Fifty years of vehicle routing. *Transportation Science*, 43(4):408–416. 5

[34] Lin, Y.-K. (2004). Reliability of a stochastic-flow network with unreliable branches nodes, under budget constraints. *IEEE Transactions on Reliability*, 53(3):381–387. 43

[35] McMasters, A. W. (1972). Optimal capacity expansion in a flow network. Technical Report NPGS55MG72091A, United States Naval Postgraduate School. 19, 20, 49

[36] MEI-KO, K. (1962). Graphic programming using odd or even points. *Chinese Math.*, 1:273–277. 9, 41

[37] Miller, C. E., Tucker, A. W., and Zemlin, R. A. (1960). Integer programming formulation of traveling salesman problems. *J. ACM*, 7(4):326–329. 5

[38] of Agriculture, U. S. D. (2004). Snowmelt. In *National Engineering Handbook. Part 630, Hydrology*, chapter 11. 10

[39] O'Neill, R. P., Castillo, A., and Cain, M. B. (2012). The iv formulation and linear approximations of the ac optimal power flow problem. *Published online at http://www. ferc. gov/industries/electric/indus-act/market-planning/opf-papers/acopf-2-iv-linearization. pdf.* 2

[40] P. Erdős, A. R. (1959). On random graphs. *Publicationes Mathematicae*, 6:290 – 297. 30, 50

[41] Perrier, N., Langevin, A., and Campbell, J. F. (2006a). A survey of models and algorithms for winter road maintenance. part i: system design for spreading and plowing. *Computers & Operations Research*, 33(1):209 – 238. 9, 41

[42] Perrier, N., Langevin, A., and Campbell, J. F. (2006b). A survey of models and algorithms for winter road maintenance. part ii: system design for snow disposal. *Computers & Operations Research*, 33(1):239 – 262. 9, 41

[43] Perrier, N., Langevin, A., and Campbell, J. F. (2007a). A survey of models and algorithms for winter road maintenance. part iii: Vehicle routing and depot location for spreading. *Computers & Operations Research*, 34(1):211 – 257. 9, 41

[44] Perrier, N., Langevin, A., and Campbell, J. F. (2007b). A survey of models and algorithms for winter road maintenance. part iv: Vehicle routing and fleet sizing for plowing and snow disposal. *Computers & Operations Research*, 34(1):258 – 294. 9, 41

[45] Rego, C., Gamboa, D., Glover, F., and Osterman, C. (2011). Traveling salesman problem heuristics: Leading methods, implementations and latest advances. *European Journal of Operational Research*, 211(3):427 – 441. 5

[46] Renaud, J. and Boctor, F. F. (2002). A sweep-based algorithm for the fleet size and mix vehicle routing problem. *European Journal of Operational Research*, 140(3):618 – 628. 6

[47] Roh, H.-J., Sharma, S., and Sahu, P. K. (2016). Modeling snow and cold effects for classified highway traffic volumes. *KSCE Journal of Civil Engineering*, 20(4):1514–1525. 11

[48] Ryan, D. M., Hjorring, C., and Glover, F. (1993). Extensions of the petal method for vehicle routing. *Journal of the Operation Research Society*, 44:289–296. 6

[49] Snelder, M. and Calvert, S. (2016). Quantifying the impact of adverse weather conditions on road network performance. *European Journal of Transport and Infrastructure Research*, 16(1). 11

[50] TRB (2000). *Highway Capacity Manual*. Transportation Research Board, Washington, DC. 11

[51] Young, S. E., Hou, Y., Garikapati, V., Chen, Y., Zhu, L., Renewable, N., and Pkwy, D. W. (2017). Initial Assessment and Modeling Framework Development for Automated Mobility Districts. In *ITS World Congress*, pages 1–13, Montreal, Canada. 56, 57

[52] Zhu, L., Garikapati, V., Chen, Y., Hou, Y., Aziz, H. M. A., and Young, S. (2018). *Quantifying the Mobility and Energy Benefits of Automated Mobility Districts Using Microscopic Traffic Simulation*. 57

Vita

Tony Kent Rodriguez was born in Kingsport, Tennessee to Evelyn Mullins. He attended Volunteer High School in Church Hill, TN. Upon graduation, Tony spent 2 years in various places within Colorado doing volunteer work. Afterwards, Tony returned to east Tennessee, where he eventually enrolled at East Tennessee State University. In the spring of 2012, Tony earned a Bachelor of Science in Mathematics. Receiving a National Science Foundation GK-12 Fellowship to fund his graduate studies at East Tennessee State University, Tony would go on to receive a Master of Science in Mathematics in the spring of 2014. In the fall of that same year, he would enroll in the University of Tennessee, Knoxville's PhD program in the Industrial and Systems Engineering department under the guidance of James Ostrowski. He was supported in part by the University of Tennessee's Chancellor's Fellowship. During his time at the University of Tennessee, Tony also collaborated with Oak Ridge National Laboratory's Modeling and Simulation Group, where he would find the questions that were addressed in his PhD dissertation. Tony completed his PhD in Industrial Engineering in May 2019.