ABSTRACT

Title of dissertation: ON THE ROUTING AND LOCATION OF MOBILE FACILITIES

Russell David Halper
Doctor of Philosophy, 2010

Dissertation directed by: Professor Subramanian Raghavan
AMSC Program
R.H. Smith School of Business

Mobile facilities play important roles in many applications, including health care, public services, telecommunications, and humanitarian relief logistics. While mobile facilities operate in different manners, it is generally considered important for a decision maker to be capable of efficiently deploying mobile facilities. This dissertation discusses two problems on the use of mathematical models and algorithms for determining efficient deployments of mobile facilities.

First we discuss the mobile facility routing problem (MFRP), which effectively models the operations of a wide class of mobile facilities that have significant relocation times and cannot service demand during transit. Chapter 2 discusses the single MFRP (SMFRP), which is to determine a route for a single mobile facility to maximize the demand serviced during a continuous-time planning horizon. We present two exact algorithms, and supporting theoretical results, when the rate demand is generated is modeled using piecewise constant functions. The first is a dynamic program that easily extends to solve cases where the demand functions take on more

general forms. The second exact algorithm has a polynomial worst-case runtime.

Chapter 3 discusses the MFRP, which addresses the situation when multiple mobile facilities are operating in an area. In such a case, mobile facilities at different locations may provide service to a single event, necessitating the separation of the events generating demand from the locations mobile facilities may visit in our model. We show that the MFRP is NP-hard, present several heuristics for generating effective routes, and extensively test these heuristics on a variety of simulated data sets.

Chapter 4 discusses formulations and local search heuristics for the (minisum) mobile facility location problem (MFLP). This problem is to relocate a set of existing facilities and assign clients to these facilities while minimizing the movement costs of facilities and clients. We show that in a certain sense the MFLP generalizes the uncapacitated facility location and p-median problems. We observe that given a set of facility destinations, the MFLP decomposes into two polynomially solvable subproblems. Using this decomposition observation, we propose a new, compact IP formulation and novel local search heuristics. We report results from extensive computational experiments.

ON THE ROUTING AND LOCATION OF MOBILE FACILITIES

by

Russell David Halper

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2010

Advisory Committee:
Professor Subramanian Raghavan, Chair/Advisor
Professor Bruce Golden
Professor Brian Hunt
Professor Elise Miller-Hooks, Dean's Representative
Professor Konstantina Trivisa

Acknowledgments

The journey that led to this dissertation would not have been possible without the generous support and encouragement of many people.

First and foremost, I would like to thank my advisor Dr. S. "Raghu" Raghavan for the countless hours he has spent with me over the past several years working on this research. Although my name appears as the sole author, this dissertation would not have been possible without his many contributions. While I doubt I can ever find the proper words to adequately express my sincere gratitude, his insight, guidance, and encouragement have been invaluable throughout my graduate career.

I am grateful to have benefitted throughout graduate school from working with a number of outstanding mentors. Dr. Eric Harder, Dr. Brian Hunt, and Dr. Jim Yorke taught me a great deal about conducting and publishing research while working on our paper, "Stability of TCP Dynamics in Large Data Networks". While the process of writing my first paper was long and challenging, they pushed me to accomplish what I never could have done on my own. Dr. Richard Wong and the members of the UPS Operations Research Group taught me much about applying operations research (O.R.) techniques in industry during my internship at UPS. Rich has been a tremendous resource and provided excellent guidance throughout my graduate career. Interning with Dr. Marcelo Torres and the Load Balancing Group at Akamai Technologies also proved to be an invaluable experience that I will continue to benefit from for years to come. Marcelo has proven to be an outstanding mentor, a good friend, and perhaps most importantly, a true soccer fan. Dr. Bruce

Golden has taught me many valuable insights throughout my graduate career in his courses and during numerous personal conversations. Dr. Michael Ball taught me the "lighter" side of O.R. while working on publishing our paper, "Scramble Teams for the Pinehurst Terrapin Classic". As head of the AMSC program, Dr. Konstantina Trivisa has been a constant source of encouragement. Both she and Alverda McCoy have helped me tremendously to navigate through the AMSC Program. Dr. Elise Miller-Hooks has my gratitude for graciously taking the time out of her busy schedule to serve on my committee.

I am fortunate to have many close and trustworthy friends who provided entertainment, encouragement, and emotional support throughout graduate school. Matt and Caitlin Hoffman have proven to be true friends over the past several years. Matt has been a great roommate throughout graduate school. Experiencing Matt and Caitlin's relationship develop has been one of the highlights of my graduate career. Jim Keiser was always there to provide inspiration and encouragement through the more difficult phases of graduate school. Justin Draa and the Draa family have been beyond supportive throughout my graduate career, and have helped me to remember the community I grew up in back home in California. I would also like to thank Lisa Greenfield, Gahl Crane, Matt Harnack, Margaret Rice, Juliana Belding, Leigh Anne Scarborough, Beth McLaughlin, Kevin Wilson, Damon Gulczynski, Carter Price, Amy Finkbiner, and the members of the Radicals soccer team. The many memorable times we have spent together over the past several years have made graduate school far more tolerable.

I am grateful for the love of my family. Since moving to Maryland, the Harrises

have been beyond warm and welcoming. Between lunches with Andy, beer making with Rick, and numerous (and delicious) home cooked family meals, it has been a true pleasure getting to know them better over the past several years. I would like to thank my grandmother, Ruth Halper, who has sent me a constant supply of biscotti and rugula that fueled much of this dissertation. Last but in no way least, I would like to thank my mom, Cathy Weiner, who was always there to help me through the emotional trials and tribulations of graduate school, perhaps more than anyone else.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

List of Abbreviations

AMSC	Applied Math and Scientific Computation Program
COLT	Cell-Site-On-Light-Trucks
COW	Cell-Site-On-Wheels
D.N.L.	Did Not Load
IP	Integer Program
IDMIP	Infinite Dimensional Mixed Integer Program
LP	Linear Program
MC	"Mathematically Challenging" Data Set
MCLP	Maximum Covering Location Problem
MFLP	Maximum Total Movement Mobile Facility
MFRP	Mobile Facility Routing Problem
MIP	Mixed Integer Program
	Location Problem
O.O.M	Out of Memory
R	"Realistic" Data Set
RT	Runtime
$\mathbb{R}$	The Set of Real Numbers
SMFLPA	Single Mobile Facility Longest Path Algorithm
SMFRP	Single Mobile Facility Routing Problem
SRDP	Schedule Resolution Dynamic Program

1. INTRODUCTION

Mobile facilities are used to provide many different services in a diverse set of fields, including humanitarian relief logistics, telecommunications, health care, warehouse and distribution center location, military logistics and national security, and public services. The term "mobile facility" has no single definition. Rather, the term "mobile facility" can be used to refer to any number of different facilities, each having their own operational characteristics. Typically, a mobile facility is some type of portable facility or vehicle capable of being repositioned to provide a service at one or more locations. The manner in which mobile facilities provide service and may be repositioned varies substantially between different types of mobile facilities. Accordingly, there is a growing body of work studying a number of diverse problems dealing with mobile facility location. This dissertation adds to this discourse, presenting novel research on two problems on the efficient deployment of mobile facilities.

One popular use of mobile facilities is in humanitarian relief logistics. In this field, mobile facilities are often used to provide services to people afflicted by manmade and natural disasters. For example, mobile communications facilities can be brought in to provide communications capabilities to responders and to individuals affected by the disaster. In addition, mobile kitchens, mobile health care

clinics, and even mobile laundry facilities are sometimes employed in such situations. Relief organization also may model the problem of determining distribution points of relief supplies during a disaster as a mobile facility problem. In such a context, the facility moved in a model could represent relief supplies or equipment that are to be relocated from a warehouse to a distribution point. Modeling problems in humanitarian relief logistics as problems in relocating mobile facilities also gives the flexibility to plan for the relocation of facilities as the demand for service changes temporally and spatially.

Another area of humanitarian relief logistics where mobile facilities are used is in providing some types of services to populations in developing countries and in rural areas. For example, mobile medical clinics and mobile vaccination clinics are used to provide health care to individuals who would not otherwise have easy access to care. This lack of access may be because these individuals live in rural areas, far from the nearest hospital. Alternatively, these individuals may be in developing countries where access to health care is limited by the availability of care, or because of limited transportation options.

Mobile facilities are also used in other areas of health care to provide many types of services to individuals who might otherwise not be able to access care, or to encourage people to seek care by making access to care more convenient. Mobile health clinics, mobile mammograms, mobile hearing testing stations, and Red Cross blood collection buses are all examples of mobile facilities in common use.

Mobile facilities are deployed by many governments and nonprofit organizations in more urban settings. For example, mobile post offices are used to provide

postal service in busy urban areas that are far from a traditional, brick-and mortar post office. Mobile libraries are another example of mobile facilities used in urban areas. Researchers have studied the re-deployment of ambulances [28, 39] and other public service vehicles [34, 42] to provide sufficient levels of service when other public service vehicles are responding to an event. In this context, each ambulance, or other service vehicle, may be considered a mobile facility that is capable of traveling to a location to provide a service upon request. Brotcorne et al. [9] give a survey on past research in ambulance location and relocation models.

In private industry, a number of problems may be modeled as problems in deploying mobile facilities. For example, a company may have warehouse or distribution centers that it may want to relocate over some planning horizon to minimize costs. Another example is given a number of factories, the problem of determining the location of a single distribution center for each factory. In this context, the distribution center may be considered abstractly as a mobile facility that must travel from the factory to the location of the distribution center. Cellular telephone service providers often deploy mobile facilities called portable cellular base stations, such as Cell-Sites-on-Wheels (COWs) and Cell-Sites-on-Light-Trucks (COLTs). Portable cellular base stations are capable of providing cellular phone service while stationary at a location and can also be quickly moved between locations. However, portable cellular base stations are unable to provide cellular telephone service while in transit. Such a company may wish to determine a route for a fleet of portable cellular base stations over the course of a day to maximize the service provided by these portable cellular base stations.

1.1 Modeling Mobile Facility Operations

Many different considerations come into play when modeling the operations of mobile facilities. This modeling should be driven by the operation of the particular class of mobile facilities in use. These considerations include the objective function of the model, aspects of facility relocation, the type of planning horizon and its length, facility capacities, and methods of providing service.

Since mobile facilities can be repositioned, it is important to use a planning horizon that appropriately reflects the realities of the operations of the particular type of mobile facility being deployed. Three types of planning horizons can be used to model mobile facility operations. When mobile facilities are to be simultaneously relocated from an initial location to a destination location, a single period planning horizon can be used. These models are sometimes referred to as relocation models and have been used, for example, to reposition ambulances [28] and other utility vehicles [34] to maintain coverage constraints while one or more of these vehicles are responding to a call. Settings where mobile facilities may be relocated more than once can be modeled with either a discrete-time (i.e., multi-period) planning horizon, or with a continuous-time planning horizon. Discrete-time planning horizons are often used to model problems where the time it takes to relocate a facility is insignificant in relation to the length of the planning horizon. For example, the relocation of warehouses by a large company may have a planning horizon of many years, while warehouses (or their contents) may be relocated relatively quickly. In such cases, the time to relocate a warehouse may be relatively small compared to

the length of the planning horizon. Discrete-time models may also be used when travel times are significant but facilities can only be relocated during a common period of time, during which no service is provided. This may be the case in applications where a mobile facility must stay at one location during the day, and are only allowed to be relocated only at night.

Conversely, continuous-time planning horizons provide the ability to model the effects of relocation times with a high degree of accuracy. This may make a continuous-time planning horizon more appropriate when modeling problems where the relocation times are significant in relation to the length of the planning horizon. Examples of this include the routing of some types of mobile facilities, such as mobile post offices or portable cellular base stations, around an urban area during the course of a day, and problems where facilities must track continuously moving clients, as in some problems in computer networking [8]. In such contexts, it must also be considered whether a mobile facility can provide service during relocation.

Facility location problems often seek to satisfy one of two objectives: minimize the cost of servicing all demand or maximize the demand serviced. The cost of operation is often a function of the type of facility used, the relocation of the mobile facilities, and the distance between customers and facilities. The appropriate choice of the objective should be driven by the application at hand. In situations where the demand serviced can be equated with revenue, one may also seek to maximize the difference of the revenue generated and the cost of operation. A discrete-time model may lend itself to applications where all demand must be covered, since this constraint would be satisfied if all demand is covered during each period of the model.

Alternatively, a continuous-time planning horizon may allow the highest degree of accuracy when maximizing the amount of demand serviced by mobile facilities.

Another important aspect of modeling mobile facility operations is the demand model used. In some settings, demand for service may accumulate while a mobile facility is not providing service. In other settings, such as in cellular communications, demand for service may be lost when service is not available. In a discrete-time model, the demand for service during each time period can be expressed as a quantity and the capacity of each mobile facility can be modeled as the quantity of demand the mobile facility can service during that period. Alternatively, in a continuous-time model, demand for service can be modeled as a function describing the rate demand is generated at each time in the planning horizon, and the capacity of a mobile facility, if it exists, may be a limit on the maximum rate that it can service demand.

1.2 Overview of Dissertation Research

This dissertation presents novel results on two operational problems dealing with mobile facilities. In Chapters 2 and 3, the mobile facility routing problem (MFRP) is presented. This problem considers a large class of mobile facilities that service demand while at a location and can be relocated over the course of a planning horizon to attempt to service as much demand as possible. Examples of such mobile facilities are portable cellular base stations and mobile food service stations. In Chapter 2, we consider the single MFRP (SMFRP). The SMFRP is to determine a

route for a single such mobile facility to maximize the total demand serviced. This problem is set in a continuous-time planning horizon, where the travel times are significant in relation to the length of the planning horizon. There is a set of locations where each mobile facility may be positioned. For each location, there is a demand profile specifying the rate each mobile facility can service demand at each time t in the planning horizon. Two exact algorithms for solving the SMFRP are presented for the case when the demand profile is represented by a piecewise constant function. The first algorithm is named the schedule resolution dynamic program (SRDP). Given a sequence of locations, the SRDP computes an optimal route visiting a subsequence of those locations in order. We show how to choose a sequence of locations that guarantees the SRDP always finds the optimal route, and prove results that enable the execution of the SRDP to be significantly sped up. The second algorithm is named the single mobile facility longest path algorithm (SMFLPA). We show that there exists a polynomial set of times when each mobile facility may either depart from or arrive at each location. The route of a mobile facility can then be viewed as a path through a directed, acyclic graph. The SMFLPA finds the optimal route by computing the longest path through this graph. While both algorithms execute quickly, the SMFLPA has a worst case runtime that is polynomial in the inputs of the problem. In addition, we discuss how these two algorithms may be adapted to situations when the demand profile is represented by more general functions. While both algorithms may be extended, the SRDP may be more easily extended than the SMFLPA.

Chapter 3 considers the MFRP in the case of routing multiple mobile facilities.

In the MFRP, the locations where each mobile facility may be positioned are separated from the events that generate demand. This allows the modeling of scenarios where mobile facilities at different locations can provide service to a single event. This occurs in many application, such as when positioning portable cellular base stations or mobile medical facilities. Each event has a demand profile specifying the rate it is generating demand for service at each time t in the planning horizon. A mobile facility at a location is able to service events nearby. We show the MFRP is NP-complete and present heuristics for computing efficient routes. Each of these heuristics use the SMFLPA as a building block in their design. The performance of these heuristics is evaluated through extensive computational testing on a variety of simulated data sets. Some of these data sets are intended to model realistic scenarios, while others are intended to be mathematically challenging.

Chapter 4 discusses the (minisum) mobile facility location problem (MFLP). This problem is one of a class of movement problems proposed by Demaine et al. [20]. It is set in a graph where facilities and clients are located at subsets of the vertices. The MFLP seeks to find destination vertices for each client and each facility such that the destination of each client is also the destination of at least one facility while minimizing the total weighted movement of all clients and facilities. We present an improved integer programming (IP) formulation [51] for the MFLP that allows a commercial solver like CPLEX to solve larger instances. We then show that given the set of vertices to be occupied by facilities, the MFLP decomposes into two polynomially solvable subproblems. Using this decomposition, we describe two novel classes of local search heuristics. Next, we introduce a new framework for

the MFLP that allows the MFLP to model more general cost structures, where the relocation costs of facilities and clients are not necessarily proportional to distance traveled. In this more general framework, the MFLP generalizes both the p-median and uncapacitated facility location problem. Finally, we report on a variety of computational experiments comparing the performance of the IP formulations and local search heuristics presented in this chapter.

Chapter 5 presents some concluding remarks and possible directions of future research.

2. THE SINGLE MOBILE FACILITY ROUTING PROBLEM

2.1 Introduction

Mobile facilities are used in many application domains, ranging from cellular telephone coverage to humanitarian relief logistics. For example, cellular telephone service providers deploy portable cellular base stations, such as Cell-Site-on-Light-Trucks (COLTs) and Cell-Site-on-Wheels (COWs), to provide cellular telephone coverage to events generating demand for service at a higher rate than an existing network of fixed base stations can provide for, or when an existing network of fixed base stations is not operational. These portable cellular base stations can be positioned at a location and provide service from there to cellular customers without any need for existing infrastructure nearby. To provide service, portable cellular base stations must be stationary at a location. They cannot service cellular phone calls while in transit. Over 100 COLTs and COWs were deployed to the Gulf Coast of the United States after Hurricane Katrina disabled the cellular networks in the region [30]. COLTs and COWs have also been deployed to provide additional coverage for large events such as Superbowl XL [48] and the 2009 Presidential Inauguration of Barack Obama [37]. These mobile facilities may be quickly relocated to provide service where it is most needed. Consequently in settings where the demand for

service changes over time, these mobile facilities can potentially be used to provide service to a large region more effectively than an equal number of fixed facilities with an equivalent capacity. The challenge this creates is how to effectively deploy such mobile facilities.

Similar mobile facilities are also used in other application domains. In some contexts, mobile facilities are used to provide services to dense urban areas where the cost of establishing a permanent fixed facility is prohibitive, or the demand for services is sporadic. For example, the U.S. Postal Service, Royal Mail [46], and the Hong Kong Post [29] deploy mobile post offices in some urban areas to provide services for customers beyond the delivery and pick-up of mail. A mobile post office may be sent to a location, allowing customers to purchase services without having to travel to a more distant, traditional post office. Similar to portable cellular base stations, mobile post offices may only provide these services while stationary at a location. No services may be provided while in transit. Mobile post offices follow a fixed schedule, allowing customers to plan for their arrival. The demand for these services also varies over time. For example, demand may be higher in commercial areas during the morning and late afternoon, while demand in residential areas may be relatively low during the work-day. A decision maker wishing to schedule such mobile facilities faces the challenge of determining a schedule that will provide as much service as possible. Another similar type of mobile facility are the U.S. Postal Inspection Service Mobile Mail Screening Stations ([43], [47]), which are truck based facilities that screen mail for security threats. These mobile facilities screen mail while at a location, but may be moved from one location to another

as threat levels change. A third type of mobile facility deployed in urban areas are trailer mounted radar speed monitors, which are placed by the side of the road to inform passing motorists of their speed. Law enforcement agencies use these mobile facilities to encourage motorists to obey speed limits. Over a planning horizon, an operator could wish to deploy such mobile facilities to maximize the number of drivers observed, the number of drivers observed exceeding the speed limit, or to areas where pedestrian interaction with traffic is high.

Mobile facilities are also used to provide humanitarian relief. These mobile facilities give a relief organization the ability to provide aid to populations dispersed in large, remote regions and in dense urban areas. For example, the Red Cross has mobile blood collection vehicles that are deployed to collect blood donations. Mobile medical clinics are used to provide care to rural areas [1] and in developing countries. Mobile vaccination clinics are also used in developing countries. In the U.S., mobile kitchens are used by the Salvation Army to serve meals to individuals in need. In each of these settings, operators of these mobile facilities would like ideally to service all demand. However in practice, limited budgets and resources can force operators instead to maximize the amount of services provided.

The advantages of using mobile facilities are three-fold. Mobile facilities can be used to augment the capacity of established fixed facilities, to provide additional service when demand levels vary significantly over time. Mobile facilities can be employed to provide service over large regions, such as rural areas, where demand may be sparse and the implementation of fixed facilities may be cost prohibitive. Mobile facilities may also be employed to provide services when existing infrastruc-

ture has been disabled, such as after a manmade or natural disaster. Operators of these mobile facilities face the difficult decision of how to utilize them to maximize the service provided.

In all the above applications, the operational settings in which these mobile facilities are used are quite similar. The rate that service is demanded at a location changes over time. The mobile facility provides a service while stationary, but can also be quickly transported between locations. No service can be provided while the mobile facility is in transit between locations. Furthermore, in each of these settings, the operational objective is to maximize the service provided.

This chapter introduces the Single Mobile Facility Routing Problem (SMFRP) of determining a route for a single mobile facility to maximize the amount of demand serviced. The next chapter considers the situation with multiple mobile facilities. In the SMFRP, there is a discrete set of locations where a mobile facility may be positioned to provide service. (These could be, for example, where appropriate permits have been obtained to place the facilities.) Each location has a demand profile that specifies the rate at which the mobile facility can service demand at that location at each time in the planning horizon. The SMFRP assumes the locations where a mobile facility may be located, the demand profiles of each location, and the travel times between locations are known ahead of time and nonstochastic. In this chapter, we introduce two exact algorithms for finding the optimal solution to an instance of the SMFRP.

The remainder of this chapter will proceed as follows. We will give a formal introduction to the SMFRP in Section 2.2. In Section 2.3, we introduce the Schedule

Resolution Dynamic Program, which given a sequence of locations, finds the best route that visits a subsequence of the sequence of locations in order. We show that by using an appropriate choice of a sequence, this algorithm produces an optimal route for the SMFRP. However, although this algorithm typically executes quickly, it appears to have an exponential worst case runtime. In Section 2.4, we give theoretical results characterizing properties of optimal routes in the SMFRP. In Section 2.5, using the charactization of optimal routes from Section 2.4, we present a second exact algorithm for computing an optimal route for an instance of the SMFRP that runs in polynomial time. The second algorithm has a natural interpretation as a longest path problem on an acyclic graph, leading to a much simplier implementation. A computational comparison of these two algorithms is presented in Section 2.6. In Section 2.7, we show how the methodologies in this chapter may be easily extended to solve several variants of the SMFRP. Section 2.8 contains concluding remarks for the chapter.

2.2 Problem Description

The objective of the SMFRP is to find a route for a single mobile facility that services the maximum amount of demand. The SMFRP takes as inputs a set of predetermined locations, L, where the mobile facility may be stationed to provide service, the travel times, $TT_{ll'}$ between each pair of locations $l, l' \in L$, and a known, non-stochastic demand profile for each location $l \in L$ over a planning horizon $[0, T]$. We assume that the travel times satisfy the triangle inequality and include any

time necessary to prepare the mobile facility for transport and to set up the mobile facility at a new location, in addition to the time the mobile facility is in transit between locations. The demand profile for location l consists of a non-stochastic *moment demand function*, $f_l(t)$, which describes the *rate* demand is being generated at location l at time t.[1] Since $f_l(t)$ is the rate demand is being generated at location l at time t, if a mobile facility is at location l from time σ to time τ, the total amount of demand serviced by that mobile facility is

$$\int_{\sigma}^{\tau} f_l(s)ds.$$

The objective of the SMFRP is to create a route for a single mobile facility that allows the mobile facility to service the maximum possible amount of demand during the planning horizon. A route consists of a sequence of *stops* $(l_n, \sigma_n, \tau_n)_{n=0}^{N}$, where l_n is the location visited during stop n, σ_n is the arrival time at stop n, and τ_n is the departure time from stop n. When the mobile facility departs stop n of the route at time τ_n, we specify that it must arrive at location l_{n+1} at time $\sigma_{n+1} = \tau_n + TT_{l_n l_{n+1}}$ and immediately begin providing service. We assume the mobile facility arrives at location l_0 at time 0 and departs location l_N at time T. (We describe in Section 2.7.1 how the model may be easily modified if the mobile facility must begin and end the planning horizon at a specific depot.) Consequently, the demand serviced

[1] More generally, demand may be considered to be generated at a discrete set of event points. A mobile facility at a location is able to service demand from nearby event points. Such is the case when we discuss the routing of multiple mobile facilities in Chapter 3. Given a single mobile facility, it is easy to compute the rate demand can be serviced at each location l by the mobile facility at each time t (i.e., $f_l(t)$).

by the mobile facility on the route may be written as,

$$\int_0^{\tau_0} f_{l_0}(s)ds + \sum_{n=1}^{N-1} \int_{\sigma_n}^{\tau_n} f_{l_n}(s)ds + \int_{\sigma_N}^{T} f_{l_N}(s)ds. \tag{2.2.1}$$

This formulation of the SMFRP technically permits routes containing a stop (l_n, σ_n, τ_n) where the mobile facility departs the location the moment it arrives (i.e., $\sigma_n = \tau_n$). The following lemma demonstrates how to remove such a stop n and create a new route that services no less demand and has either $\sigma_{n+1} = \tau_{n-1} + TT_{l_{n-1}l_{n+1}}$, $\sigma_{n+1} = 0$, or $\tau_{n-1} = T$. This allows us to restrict our attention to sequences of stops where a mobile facility stays at each stop for a strictly positive amount of time.

Lemma 2.2.1. *Suppose $(l_n, \sigma_n, \tau_n)_{n=0}^{N}$ is a route and $\sigma_{n_0} = \tau_{n_0}$. Then a new route servicing no less demand can be created by applying one of the following three modification:*

1. *If $n_0 = 0$, then remove Stop 0 from the route and set $\sigma_1 = 0$.*

2. *If $n_0 = N$, then remove Stop N from the route and set $\tau_{N-1} = T$.*

3. *Otherwise, remove Stop n_0 from the route and set $\sigma_{n_0+1} = \tau_{n_0-1} + TT_{l_{n_0-1}l_{n_0+1}}$.*

Furthermore, an optimal route $(l_n, \sigma_n, \tau_n)_{n=0}^{N'}$ exists where $\sigma_n < \tau_n$ for each $n = 0, \ldots, N'$.

Proof. Since $\sigma_{n_0} = \tau_{n_0}$, zero demand is serviced by the mobile facility at Stop n_0. Thus removing Stop n_0 from each route in any of the three modifications does not reduce the amount of demand serviced. Applying modification 1 causes the mobile facility to arrive earlier at Stop 1. The departure time for Stop 1, and the arrival and

departure times for Stops 2 through N remain the same. Thus, an equal or greater amount of demand is serviced along this updated route. Similarly, modification 2 causes the mobile facility to depart Stop $N-1$ later, thus an equal or greater amount of demand is serviced. Suppose modification 3 is implemented. Let σ be the arrival time at Stop n_0+1 before the modification and $\tilde{\sigma}$ be the arrival time at Stop n_0+1 after the modification. Then,

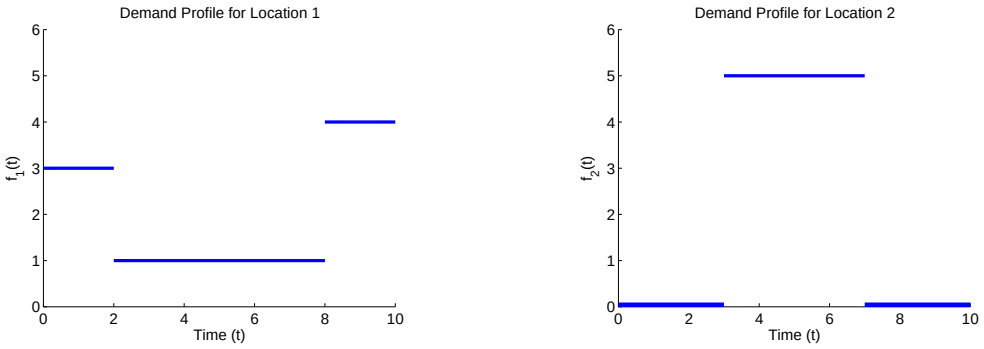
$$\tilde{\sigma} = \tau_{n_0-1} + TT_{l_{n_0-1}l_{n_0+1}}$$

$$<= \tau_{n_0-1} + TT_{l_{n_0-1}l_{n_0}} + TT_{l_{n_0}l_{n_0+1}} = \sigma_{n_0} + TT_{l_{n_0}l_{n_0+1}} = \tau_{n_0} + TT_{l_{n_0}l_{n_0+1}} = \sigma.$$

Thus, modification 3 causes the mobile facility to arrive at Stop n_0+1 earlier than before the modification, meaning an equal or greater amount of demand is serviced at Stop n_0. Consequently, no less demand is serviced along the route. Finally, given an optimal route of length N, a sequence of at most $N-1$ of these operations will produce a new route $(l_n, \sigma_n, \tau_n)_{n=0}^{\tilde{N}}$ servicing an equal or greater amount of demand. Thus, this new route is also optimal. □

The mobile facility does not service any demand while in transit between locations. Thus, an efficient route must balance the desire to provide service to locations while they are generating demand for service at a high rate with the amount of time the mobile facility must spend in transit.

We will assume each moment demand function, $f_l(t)$, is piecewise constant. We make this assumption to allow us to analyze the runtime of the two proposed algorithms. Since many types of functions can be approximated arbitrarily closely by piecewise constant functions (such as continuous functions), this assumption is



(a) (b)

Fig. 2.1: Examples of demand profiles for two locations, 1 and 2. In this example, the planning horizon is $[0, 10]$.

not very restrictive. (In Section 2.7.3, we discuss how the SMFRP can be solved when the moment demand functions take on more general forms.) Formally, $f_l(t)$ is a nonnegative, piecewise constant function that may assume nonzero values only during the planning horizon $[0, T]$. For each location l, let M_l be the number of discontinuities of the moment demand function $f_l(t)$. Let $q_1^l = 0$, $q_{M_l}^l = T$, and for $2 \leq j \leq M_l - 1$, let q_j^l be the j-th discontinuity of the function $f_l(t)$. (We will consider 0 and T "discontinuities", regardless of nearby values of $f_l(t)$.) For a location l, we refer to q_j^l as *critical time j of location l* and we refer to the interval of time $[q_j^l, q_{j+1}^l)$ as *step j of location l*. Finally let g_j^l be the value of $f_l(t)$ during step j of location l.

2.2.1 An Example of the SMFRP.

Examples of the demand profiles for two locations are given in Figures 2.1(a) and 2.1(b). Suppose the travel time between the two locations is two (i.e., $TT_{12} =$

$TT_{21} = 2$). One possible route for the mobile facility could start at Location 1 at time 0 and remain there until time 2. The mobile facility would service demand at rate 3 during these 2 units of time, servicing a total of 6 units of demand. The mobile facility could then depart Location 1 at time 2 to travel to Location 2. It would arrive at Location 2 at time 4 and begin servicing demand at rate 5. The mobile facility could continue to service demand at rate 5 until time 7, servicing a total of 15 units of demand. The mobile facility could then depart Location 2 and return to Location 1 at time 9 and service demand at rate 4 until time 10, the end of the planning horizon. This services an additional 4 units of demand from Location 1. A total of 25 units of demand are serviced along this route. As we will see, this is the optimal route for this example.

2.2.2 Related Work

The goal of the SMFRP is to find a path for a single mobile facility to maximize the demand covered in a limited period of time. In this sense, the SMFRP bears a resemblance to the orienteering problem. The orienteering problem was introduced by Tsiligride in [45]. Given a collection of locations, each with a prize of a known value, the orienteering problem is to find a path that allows the maximum value of prizes to be collected during a given period of time.

However, there are several distinct differences between the SMFRP and the orienteering problem. Firstly, the amount of demand serviced during a stop in a solution to the SMFRP is dependent on the arrival and departure time of the mobile facility. Thus it differs from the orienteering problem in that value of the prize

collected at a stop in the orienteering problem is independent of the time the vehicle arrives. Kantor and Rosenwein [32] studied the orienteering problem with time windows, where the prize at each location may only be collected during a specific time window. However, this still differs from our model as the value of the prizes collected in the orienteering problem with time windows is not dependent on the duration of time spent at each stop. In general, a solution to the orienteering problem should seek to minimize the amount of time spent at each location. Conversely, a solution to the SMFRP may possibly service more demand from staying longer at some stops, depending on the rate demand is being generated.

Bespamyatnikh et al. [8] also studied a problem in locating a single mobile facility in a continuous time planning horizon. Given a set of customers following continuous trajectories through space, Bespamyatnikh et al. studied how to compute a trajectory for a facility that minimizes a number of different objectives. Unlike the SMFRP where facilities must be at a location to service demand, facilities in this problem service demand while moving continuously.

2.3 The Schedule Resolution Dynamic Program and Computing an Optimal Route

A natural first question to ask when considering how to route a single mobile facility is, given a sequence of locations $(l_1, l_2, \ldots, l_{N'})$, what is the best route for a mobile facility that may only visit these locations in order? To answer this question, an arrival time and departure time must be determined for every stop the mobile

facility makes. Furthermore, while it is tempting to determine the route visiting every location in the sequence $(l_1, l_2, \ldots, l_{N'})$, it might be possible to service more demand by visiting only a subsequence $(l_{n_1}, \ldots, l_{n_N})$, where $1 \leq n_1 < \cdots < n_N \leq N'$. Indeed, the mobile facility may be able to service more demand by skipping a location in the sequence generating low levels of demand, or a location in the sequence located far from the other locations. With this in mind, we have developed the schedule resolution dynamic program (SRDP) that, *given a sequence of locations* $(l_1, l_2, \ldots, l_{N'})$, finds the best route where the locations visited along the route are a subsequence $(l_{n_1}, \ldots, l_{n_N})$, with $1 \leq n_1 < \cdots < n_N \leq N'$. (See [7] background on dynamic programming.) The SRDP does so by computing a *stop strategy function* for each sequence member l_n that describes what the mobile facility should do if it is at location l_n of the sequence at time t, as well as the amount of demand that can be serviced by executing this decision.

Define a *candidate sequence* to be a sequence of locations $(l_1, l_2, \ldots, l_{N'})$. A candidate sequence is allowed to contain multiple copies of the same location although we prohibit a location to be repeated consecutively in a sequence, which is nonrestrictive. For example, given three locations 1, 2 and 3, the sequence (1, 2, 1, 3) is a valid candidate sequence, while the sequence (1, 2, 2, 3) is not a valid candidate sequence. A mobile facility following the candidate sequence (1, 2, 1, 3) would be allowed to visit locations, 1, 2, and 3 in order, or locations 2, 1, and 3 in order, but it could not ever follow a route that travels from location 3 to location 2. (It may be conceptually easier to imagine that each member candidate sequence is a distinct location for the remainder of this section. However, the results below hold

when locations are allowed to be repeated nonconsecutively in a sequence) A route is said to be *in a candidate sequence* if the order locations are visited along that route is a subsequence of the candidate sequence. We will show how, given a candidate sequence, to find a route in the candidate sequence servicing the maximum possible amount of demand.

For each location l, define the *remaining demand function* $F_l(t)$ by the formula

$$F_l(t) = \int_t^T f_l(s)ds.$$

Thus, $F_l(t)$ gives the total amount of demand that will be serviced by the mobile facility if it is stationed at location l from time t to time T, the end of the planning horizon. Notice that $F_l(t)$ is a continuous, piecewise linear, decreasing function with slope $-g_j^l$ on the step $[q_j^l, q_{j+1}^l)$. Consequently, critical times of $f_l(t)$ are also critical times of $F_l(t)$.

The *schedule resolution dynamic program* (SRDP) determines an optimal route in a candidate sequence. Given a candidate sequence $(l_1, \ldots, l_{N'})$, the SRDP computes for each $n = 1, \ldots, N'$ a *stop strategy function* $r_n(t)$. The stop strategy function for location l_n, $r_n(t)$, dictates the optimal routing decision that should be made to capture the maximum possible demand, given that the mobile facility is at location l_n at time t and must follow a route in the candidate sequence. More specifically, at time t the stop strategy function either tells the mobile facility to remain at location l_n, or to leave immediately and travel to a specific location in the set $\{l_{n+1}, \ldots, l_{N'}\}$, and how much demand can be captured by following this strategy. In order to describe this routing decision for a mobile facility at location l_n at time

t, the stop strategy function $r_n(t)$ is composed of two coordinate functions, $r_n^1(t)$ and $r_n^2(t)$ (i.e., $r_n(t) = (r_n^1(t), r_n^2(t))$). The function $r_n^1(t)$ describes the maximum amount of demand the mobile facility can capture if it is at location l_n at time t and follows the strategy prescribed by the SRDP for the candidate sequence. It will be a piecewise linear, decreasing function that takes on values in $\mathbb{R}^+ \cup \{0\}$. The function $r_n^2(t)$ takes on values in $\{n, \ldots, N'\}$ and indicates which location the mobile facility at location l_n at time t should travel to in order to service the amount of demand specified by $r_n^1(t)$. If $r_n^2(t) = n$, then the mobile facility is to remain servicing location l_n at time t. Otherwise, $r_n^2(t) = n'$ for some $n' > n$, which indicates the mobile facility is to immediately depart location l_n for location $l_{n'}$.

Choose an $n < N'$ and assume we know $r_{n'}(t)$ for $n' = n+1, \ldots, N'$. The stop strategy function for location l_n will be computed from $F_{l_n}(t)$ and the stop strategy functions $r_{n'}(t)$ for $n' = n+1, \ldots, N'$. If a mobile facility following the candidate sequence is at location l_n at time t and is immediately to leave location l_n for a subsequent location in the candidate sequence, $l_{n'}$, it will arrive at time $t + TT_{l_n l_{n'}}$. Naturally, given the option, the mobile facility should choose to travel to the location $l_{n'}$ for which $r_{n'}^1(t + TT_{l_n l_{n'}})$ is the largest when it arrives. We define

$$max_n(t) = \max_{n'>n}\{r_{n'}^1(t + TT_{l_n l_{n'}})\}. \tag{2.3.1}$$

$$argmax_n(t) = \text{argmax}_{n'>n}\{r_{n'}^1(t + TT_{l_n l_{n'}})\}. \tag{2.3.2}$$

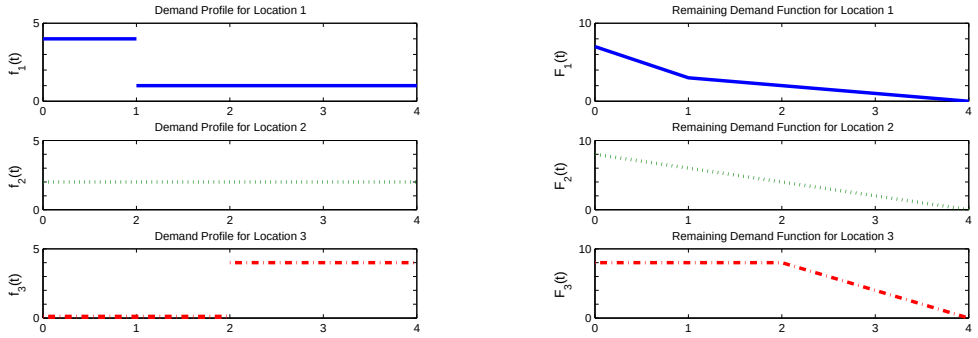
In the case of ties in 2.3.1 and 2.3.2, we choose the location later in the sequence. If the mobile facility is at location l_n at time t and must immediately leave to begin servicing a location later in the candidate sequence, then $max_n(t)$ specifies

the most demand the mobile facility can capture after time t following a route in the candidate sequence, and $argmax_n(t)$ specifies the index of location in the candidate sequence the mobile facility should travel to in order to capture this much demand. Since these two functions describe what the mobile facility should do if it must leave location l_n at time t, we refer to the pair $(max_n(t), argmax_n(t))$ collectively as the *departure strategy function* for location l_n.

Notice both the stop strategy functions and the departure strategy functions take on values in $(\mathbb{R}^+ \cup \{0\}) \times \{1, 2, \ldots, N'\}$. We define a *critical time* of a stop strategy function $r_n(t)$ as a time t when either $r_n^1(t)$ or $r_n^2(t)$ is non-differentiable. Similarly, we define a *critical time* of a departure strategy function as a time t when either $max_n(t)$ or $argmax_n(t)$ is non-differentiable. We adopt the further convention that 0 and T are always critical times of every stop strategy function and departure strategy function. While the critical times of the remaining demand function $F_{l_n}(t)$ are the times when $F_{l_n}(t)$ changes slope, the critical times of $r_n(t)$ or $(max_n(t), argmax_n(t))$ are when either $r_n^1(t)$ or $max_n(t)$ changes slope, or when $r_n^2(t)$ or $argmax_n(t)$ changes value.

2.3.1 An Example of Finding an Optimal Route in a Candidate Sequence.

Before giving a formal description of the SRDP, it is conceptually useful to see an example of computing the stop strategy functions for a candidate sequence. Doing so will provide a better understanding of the decisions the SRDP makes during execution. The following example computes the stop strategy functions exactly as would the SRDP.



(a) (b)

Fig. 2.2: Above, Panel (a) gives examples of moment demand functions for three locations. The corresponding remaining demand functions for each location are given in Panel (b). The graphs are color coded by locations.

Suppose we have three locations, 1, 2, and 3, a planning horizon of length four ($T = 4$), and it takes one unit of time to travel between any pair of locations. Let Location 1 generate demand at rate 4 during time interval $[0, 1)$ and generate demand at rate 1 during $[1, 4)$. Let Location 2 generate demand at rate 2 during $[0, 4)$. Finally, let Location 3 generate demand at rate 0 during $[0, 2)$ and generate demand at rate 4 during $[2, 4)$. The moment demand functions for each location are given in Figure 2.2(a) and the corresponding remaining demand functions for each location are given in Figure 2.2(b). We have assigned a different line style to each location that are used in these graphs. Location 1 is plotted using a solid line, Location 2 is plotted using a dotted line, and Location 3 is plotted using a line of alternating dashes and dots. These line styles will be used to distinguish these three locations in all plots in this subsection.

Suppose the candidate sequence is $(l_1 = 1, l_2 = 2, l_3 = 3)$. We will work

backwards through the sequence, first determining the stop strategy function for Location 3. Since Location 3 is the last location in the candidate sequence, a mobile facility at Location 3 and following the candidate sequence cannot leave for any other location. Doing so would violate the order of the locations in the candidate sequence. Thus, this mobile facility must remain servicing Location 3 until the end of the planning horizon. Consequently, the amount of demand this mobile facility could service after time t is simply the amount of demand generated at Location 3 between time t and time T, which is precisely $F_3(t)$. Thus, the stop strategy function for Location 3 is given by $r_3^1(t) = F_3(t)$ and $r_3^2(t) = 3$ for every time t in the planning horizon. Figure 2.3 shows the stop strategy function for Location 3.

Once the stop strategy function for Location 3, the last stop in the candidate sequence, has been determined, the next step is to compute a strategy for the next (previous) location in the candidate sequence, Location 2. To do so, we first compute the departure strategy function for Location 2, $(max_2(t), argmax_2(t))$. Recall if the mobile facility is at Location 2 at time t and must immediately depart for a location later in the candidate sequence, then $max_2(t)$ will specify the maximum amount of demand the mobile facility can capture after time t, and $argmax_2(t)$ specifies the location the mobile facility should travel to in order to service this amount of demand. In this example, there is only one location after Location 2 in the candidate sequence, namely Location 3. Thus, if the mobile facility following the candidate sequence is at Location 2 at time t and is to leave, it must travel to Location 3. Since it takes one unit of time to travel from Location 2 to Location 3, the mobile facility would arrive at Location 3 at time $t + 1$. Consequently, we can

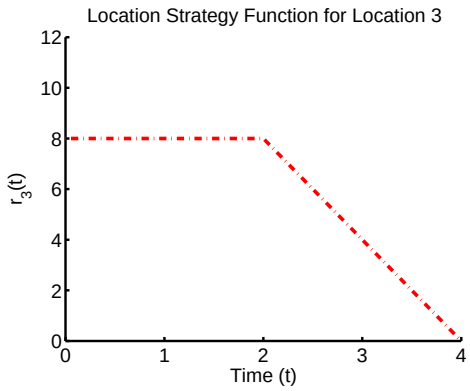
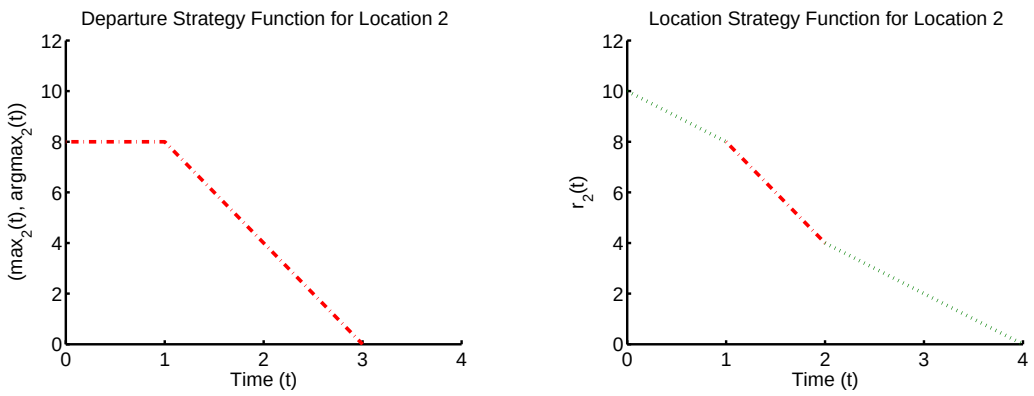


Fig. 2.3: Since location 3 is the last location in the candidate sequence, the stop strategy function is $r_3(t) = (F_3(t), 3)$.

define $max_2(t) = r_3^1(t+1)$ and $argmax_2(t) = 3$ for all t. Figure 2.4(a) displays the departure strategy function for Location 2.

Once the departure strategy function for Location 2 has been computed, we must compute $r_2(t)$, the stop strategy function for Location 2. In essence, we will determine for each time t if the mobile facility should stay servicing demand at Location 2, or if it should follow the departure strategy function for Location 2 and leave. We compute $r_2(t)$ by working backwards through time, starting at the end of the planning horizon ($T = 4$), and making a computation for each critical time of $F_2(t)$ and $(max_2(t), argmax_2(t))$. Between two consecutive critical times of these two functions, both $F_2(t)$ and $max_2(t)$ have constant slope. This allows the stop strategy function to be computed during the interval of time between the two consecutive critical times in one computation.

Firstly, since the mobile facility cannot depart Location 2 after time 3 and arrive at Location 3 at before time 4, the SRDP defines $r_2^1(t) = F_2(t)$ and $r_2^2(t) = 2$ for $3 \leq t \leq 4$. Next, the latest critical time of either $F_2(t)$ or $(max_2(t), argmax_2(t))$



(a) (b)

Fig. 2.4: Panel (a) shows $(max_2(t), argmax_2(t))$. Panel (b) shows the stop strategy functions for Location 2. The function $max_2(t)$ in Panel (a) and $r_2^1(t)$ in Panel (b) are represented by height, while $argmax_2(t)$ in Panel (a) and $r_2^2(t)$ Panel (b) are represented by line style.

before time 3 is considered. This is the critical time of the departure strategy function $(max_2(t), argmax_2(t))$ at time $t = 2$. We compute $F_2(t)$ for $2 \leq t < 3$. Notice that between time 2 and 3, both $F_2(t)$ and $max_2(t)$ have constant slope. The remaining demand function for Location 3, $F_3(t)$, intersects $max_2(t)$ at time 3, and is greater than or equal to $max_2(t)$ for $t \in [2, 3)$. Thus, more demand will be serviced by remaining at Location 2 between time 2 and time 3. Consequently, we define $r_2(t) = (F_2(t), 2)$ for $t \in [2, 3)$.

Now that $F_2(t)$ has been computed for $2 \leq t \leq 4$, we choose the latest critical time of either $F_2(t)$ or $(max_2(t), argmax_2(t))$ before time 2, and then compute $F_2(t)$ from that critical time until time 2. In this example, the next the critical time to consider is the critical time of $(max_2(t), argmax_2(t))$ at time 1. During the time interval $[1, 2)$, $max_2(t)$ is greater than $F_2(t)$. However, we must consider the rate

the mobile facility may service demand at Location 2 during $[1, 2)$ compared to the rate the mobile facility would service demand upon arrival at Location 3, if it were to leave Location 2 during $[1, 2)$. When this rate is higher at Location 2, the mobile facility should choose to remain at Location 2 since the option exists to depart for Location 3 at a later time. In this case, during the time interval $[1, 2)$, the slope of $max_2(t)$ is steeper than (i.e., less than) the slope of $F_2(t)$. In other words, the rate demand can be serviced at Location 2 during $[1, 2)$ is less than the rate demand could be serviced upon arrival at Location 3 if the mobile facility were to immediately depart Location 2 for Location 3. Thus, the mobile facility should immediately depart Location 2 for Location 3. Consequently, the SRDP will define $r_2(t) = (max_2(t), argmax_2(t))$ for $t \in [1, 2)$.

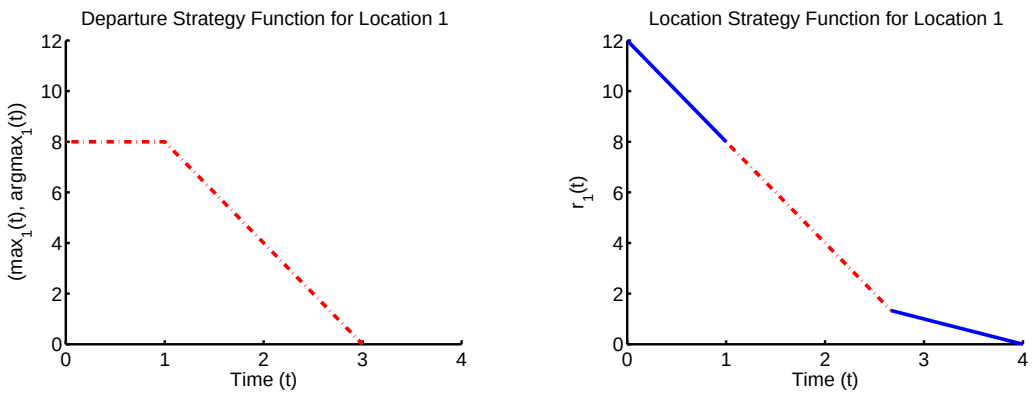
The next and final critical time to consider when computing $F_2(t)$ is at $t = 0$. Although $max_2(t)$ is greater than the remaining demand function for Location 2, $F_2(t)$, during $[0, 1)$, demand may be serviced at a faster rate by staying at Location 2 at any time t in $[0, 1)$, than at Location 3 at time $t+1$ (the time the mobile facility would arrive if it departed Location 2 at time t). Since the option to depart later from Location 2 for Location 3 always exists, more demand will be serviced by staying at Location 2 during $[0, 1)$. However, we cannot simply define $r_2(t) = (F_2(t), 2)$ for $t \in [0, 1)$. Doing so would result in $r_2^1(t)$ having a discontinuity at time 1 and describing less demand than would be serviced when following the actual route. In fact, doing so would make $r_2^1(.9) < r_2^1(1)$. Logically, since the stop strategy functions define what the mobile facility should do if it is at a location at any given time, they should be non-increasing over time. To appropriately define $r_2(t)$ during $[0, 1)$ let

$\texttt{offset} = r_2^1(1) - F_2(1)$. Then, we may define $r_2^1(t) = F_2^1(t) + \texttt{offset}$. This prevents a discontinuity of $r_2^1(t)$, and accurately describes the amount of demand the mobile facility may service if it were at Location 2 at a time t in $[0, 1)$.

Figure 2.4(b) displays the stop strategy function for Location 2. The height of the graph is $r_2^1(t)$. The location indicated by $r_2^2(t)$ is represented on the graph by the corresponding line style. In particular, the graph is dotted where $r_2^2(t) = 2$, and the graph alternates between dashes and dots where $r_2^2(t) = 3$.

Finally, the SRDP computes the strategy for Location 1. This is done in much the same way as for Location 2. The key difference comes in the computation of the departure strategy function. Specifically, the option exists when departing Location 1 to travel to either Location 2 or Location 3. It takes one unit of time to travel from Location 1 to either Location 2 or Location 3. Since the option exists, the mobile facility departing Location 1 at time t should always travel to the location n, with $n = 2, 3$, where $r_n^1(t + 1)$ is greatest. Consequently, at a given time t, $max_1(t)$ must be the maximum of $r_2(t+1)$ and $r_3(t+1)$. In this example, $r_3^1(t+1)$ is always greater than $r_2^1(t + 1)$. Thus $max_1(t) = r_3(t + 1)$ and $argmax_2(t) = 3$ for all t. Figure 2.5(a) shows the departure strategy function for Location 1.

To compute the stop strategy function for Location 1, $r_1(t)$, we begin again at time 4 and again work backwards through the critical times of the functions $F_1(t)$ and $(max_1(t), argmax_1(t))$. Since the mobile facility cannot depart Location 1 at or after time 3 and arrive at another location before time 4, $r_1(t) = (F_1(t), 1)$ for t in $[3, 4]$. The next critical time considered is at time 1, and is a critical time of both $F_1(t)$ and $max_1(t)$. During the interval of time $[1, 3)$, $F_1(t)$ has a greater



(a) (b)

Fig. 2.5: Panel (a) shows $(max_1(t), argmax_1(t))$. Panel (b) shows the stop strategy function for Location 1. The functions $max_1(t)$ in Panel (a) and $r_1^1(t)$ in Panel (b) are represented by height, while $argmax_1(t)$ in Panel (a) and $r_1^2(t)$ Panel (b) are represented by line style.

slope (i.e., is less steep) than $max_1(t)$. Thus, at a time t in $[1, 3)$, the strategy of the mobile facility should determined by the greater of these two functions. In this case, $F_1(t)$ and $max_1(t)$ intersect at time 2.67. Thus, we define $r_1(t) = (F_1(t), 1)$ during $[2.67, 3)$, when $F_1(t)$ is greater than or equal to $max_1(t)$, and define $r_1(t) = (max_1(t), argmax_1(t))$ during $[1, 2.67)$. The final step is to compute $r_1(t)$ during $[0, 1)$. Just as in the computation of $r_2(t)$, $F_1(t)$ is steeper than $\max_1(t)$ during $[0, 1)$, indicating the mobile facility should stay at Location 1 during $[0, 1)$. Thus, set $\texttt{offset} = \max_1(1) - F_1(1)$ and define $r_1(t) = (F_1(t) + \texttt{offset}, 1)$ for $0 \leq t < 1$. Figure 2.5(b) shows the stop strategy function for Location 1.

The optimal route the mobile facility may take while following the candidate sequence can now be determined. Since at time $t = 0$, $r_1^1(0)$ is greater than $r_2^1(0)$ and $r_3^1(0)$, the mobile facility can service the most demand if it begins at Location

1. It will stay at Location 1 while $r_1^2(t) = 1$. In this case, this occurs at time $t = 1$, when $r_1^2(1) = 3$. Consequently, at time 1 the mobile facility will leave Location 1 and travel to Location 3, arriving there at time 2. Following $r_3(t)$, the mobile facility will then provide service at Location 3 until time 4, the end of the planning horizon. A total of twelve units of demand will be serviced, four units from Location 1 and eight units from Location 3. In other words, the amount of demand serviced along the route is $r_1^1(0) = 12$. Notice Location 2 is never visited, despite being in the candidate sequence. As the reader may observe, any route in the candidate sequence that visits Location 2 will result in less than twelve units of demand being serviced.

2.3.2 Description of the SRDP

The SRDP computes departure strategy functions for each location in the sequence using the same basic method as in the example above. Given a candidate sequence $(l_1, \ldots, l_{N'})$, the SRDP begins by first creating the stop strategy function for last location in the candidate sequence, $l_{N'}$. The SRDP then works backwards through the candidate sequence, location by location, first generating the departure strategy function $(max_n(t), argmax_n(t))$, and then computing the stop strategy function $r_n(t)$ from $F_{l_n}(t)$ and $(max_n(t), argmax_n(t))$. The stop strategy function $r_n(t)$, for $n = 1, \ldots, N' - 1$, is computed by working backwards through the critical times of $F_n(t)$ and $(max_n(t), argmax_n(t))$.

It is conceptually helpful to make a few observations about the route of a mobile facility following a candidate sequence $(l_1, \ldots, l_{N'})$.

1. If the mobile facility is at location $l_{N'}$, the last location in the candidate sequence, then the mobile facility must stay at that location until the end of the planning horizon. Consequently, we must have $r^1_{N'}(t) = F_{l_{N'}}(t)$ and $r^2_{N'}(t) = N'$ for all $t \in [0, T]$

2. Assuming the travel time between a pair of locations l and l' is nonzero, there is a point in time when the mobile facility will not be able to leave location l and arrive at location l' before the end of the planning horizon. Thus, for each $n \in \{1, \ldots, N'\}$, it may be assumed that $r^1_n(t) = F_{l_n}(t)$ and $r^2_n(T) = n$ for all $t \geq \max_{n'=n+1,\ldots,N'} T - TT_{l_n l'_n}$.

3. Given that a mobile facility is leaving location l_n at time t for a location later in the candidate sequence, $argmax_n(t)$ is the index of the location in the candidate sequence the mobile facility should travel to in order to service the most demand.

4. If at time t, $f_{l_n}(t) > f_{l_{n'}}(t + TT_{l_n l_{n'}})$, then the mobile facility should not travel to location $l_{n'}$ at time t, since the mobile facility may continue to service demand at a higher rate at its current location than at the time it will arrive at location $l_{n'}$, and because the mobile facility always has the option of leaving for location $l_{n'}$ at a later time. In particular, the mobile facility should only depart location l_n if the slope of $F_n(t)$ is greater than (less steep than) the slope of $max_n(t)$.

Noting these observations, the steps of the SRDP can be described as follows:

Step 0: (Initialization) Define $r_{N'}(t) := (F_{l_{N'}}(t), N')$ for all $t \in [0, T]$, and set $n := N' - 1$.

Step 1: (Departure Strategy Step) Compute the departure strategy function for location l_n, $(max_n(t), argmax_n(t))$, in the candidate sequence from the stop strategy functions $r_{n'}(t)$, for $n' = n+1, \ldots, N'$ from Equations 2.3.1 and 2.3.2.

Step 2: (Location Strategy Step) From $(max_n(t), argmax_n(t))$ and $F_{l_n}(t)$, compute the stop strategy function for location l_n, $r_n(t)$, as described below. If $n = 1$, terminate. Otherwise set $n := n - 1$ and return to Step 1.

For a fixed n, Step 2 begins by defining the function $r_n^1(t) = F_{l_n}(t)$ and $r_n^2(T) = n$ for $t > \max_{n'=n+1,\ldots,N} T - TT_{l_n,l'_n}$, which may be done by Observation 2. As in our example we will need to define an additional variable `offset`, which is initialized to 0. Then the SRDP works backwards through through the critical times of $F_{l_n}(t)$ and $(max_n(t), argmax_n(t))$ to determine the stop strategy function. Since both $F_{l_n}(t)$ and $max_n(t)$ have constant slope during the interval between each consecutive pair of critical times considered, the SRDP may compute the stop strategy function for this entire interval of time in one computation. At each time $t \in [0, T]$, either $r_n^1(t)$ will assume the value $max_n(t)$ and $r_n^2(t) = argmax_n(t)$, or $r_n^1(t)$ will assume the value of $F_{l_n}(t)$ + `offset` and $r_n^2(t) = n$. Suppose the stop strategy function $r_n(t)$ has been computed between critical time t_1 and T. Let t_0 be the latest critical time of either $F_{l_n}(t)$ and $(max_n(t), argmax_n(t))$ before time t_1. One of two decisions is made for this critical time:

Case 1 ($r_n^2(t_1) = n$)*:* Check to see if there is a time $\tau \in [t_0, t_1)$ when $max_n(t)$ intersects $F_{l_n}(t) + \texttt{offset}$. If τ exists, define $r_n(t) = (F_{l_n}(t) + \texttt{offset}, n)$ for $t \in [\tau, t_1)$ and $r_n(t) = (max_n(t), argmax_n(t))$ for $t \in [t_0, \tau)$. Otherwise, define $r_n(t) = (F_{l_n}(t) + \texttt{offset}, n)$ for $t \in [t_0, t_1)$

Case 2 ($r_n^2(t_1) > n$)*:* Compare the slopes of $F_{l_n}(t)$ and $max_n(t)$ during the time interval (t_0, t_1). (Note that the slope of both functions is constant on this interval by the choice of t_0 and t_1.) If the slope of $max_n(t)$ is less than (steeper than) the slope of $F_{l_n}(t)$, then define $r_n(t) = (max_n(t), argmax_n(t))$ for $t \in [t_0, t_1)$. Otherwise, first set $\texttt{offset} = max_n(t_1) - F_{l_n}(t_1)$ and then define $r_n(t) = (F_{l_n}(t) + \texttt{offset}, n)$ for $t \in [t_0, t_1)$.

Step 2 continues to work backwards through the critical times of both $F_{l_n}(t)$ and $(max_n(t), argmax_n(t))$ until the stop strategy function $r_n(t)$ has been defined for the entire planning horizon, $[0, T]$.

At the end of the SRDP, locations strategy functions have been defined for each location in the candidate sequence. An optimal route may now be determined from these stop strategy functions. To determine the route, the mobile facility must begin at the location l_n for which $r_n(0)$ is greatest. The mobile facility stays at that location until the stop strategy function $r_n(t)$ dictates that the mobile facility should leave for some later location $l_{n'}$ in the candidate sequence. When the mobile facility arrives at location $l_{n'}$, it will then follow the instructions of $r_{n'}^2(t)$ and service demand from that location until this stop strategy function dictates that the mobile facility should depart to a location later in the candidate sequence. The route then

continues in this manner until the end of the planning horizon.

Theorem 2.3.1. *Let $(l_1, \ldots, l_{N'})$ be a candidate sequence. Suppose that a mobile facility is at location l_n in the sequence at time t and has the option to either remain servicing location l_n, or depart to a location of its choosing in the set $\{l_{n+1}, l_{n+2}, \ldots, l_{N'}\}$. Then a strategy for capturing the maximum possible amount of demand during $[t, T]$ is given by $r_n(t)$.*

Proof. Since a mobile facility at location $l_{N'}$ in the candidate sequence cannot leave location $l_{N'}$ to travel to any other location, the optimal strategy is given by $r_{N'}(t) = (F_{l_{N'}}(t), N')$ for all $t \in [0, T]$. The SRDP defines this in Step 0.

The remainder of the proof follows by backwards induction. Fix $n < N'$ and suppose the stop strategy functions $r_{n+1}(t), \ldots, r_{N'}(t)$ define an optimal strategy. The SRDP begins calculating the stop strategy function $r_n(t)$ by defining the function $r_n^1(t) = F_{l_n}(t)$ and $r_n^2(T) = n$ for $t > \max_{n'=n+1,\ldots,N} T - TT_{l_n,l'_n}$, which may be done by Observation 2. Now let B be the collection of critical times of $F_{l_n}(t)$ and $(max_n(t), argmax_n(t))$. Then B is the set of critical times considered by the SRDP. Fix a time $t_{cur} \in [0, T]$ and assume that for $t \in [t_{cur}, T]$, $r_n(t)$ is the best strategy for the mobile facility at location l_n of the candidate sequence at time t. Let t_{next} be the next critical time the SRDP considers from B after t_{cur}. Then $t_{next} < t_{cur}$ and t_{next} is the first critical time before time t_{cur} of either $F_{l_n}(t)$ or $(max_n(t), argmax_n(t))$. Between time t_{next} and time t_{cur}, both $F_{l_n}(t)$ and $max_n(t)$ have no critical times and thus have constant slopes. Furthermore, $argmax_n(t)$ is constant over the interval $[t_{next}, t_{cur})$. If the mobile facility is to leave the n-th location in the candidate se-

quence at time t for a subsequent location in the candidate sequence, it should leave for the location indicated by $argmax_n(t)$.

Suppose $r_n^2(t_{cur}) = n'$ for some $n' \geq n+1$. While the mobile facility has the option to leave stop n at any time $t \in (t_{next}, t_{cur}]$, once it leaves it may never return. Consequently, at time t it is more advantageous to stay at location l_n while location l_n is generating demand at a faster rate at time t than the rate location $l_{argmax_n(t)}$ will be generating demand when the mobile facility arrives at time $t + TT_{l_n,l_{n'}}$. This is true regardless of the height of $max_n(t)$. Thus, the decision of the SRDP to stay at location l_n during $[t_{next}, t_{cur})$ when the slope of $F_{l_n}(t)$ is less than (i.e., steeper than) the slope of $max_n(t)$ is optimal. Conversely, when this is not the case, the decision of the SRDP to leave immediately for location of the candidate sequence indicated by $argmax_n(t)$ is optimal.

Similarly, if $r_n(t_{cur}) = n$, then for any time $t \in [t_{next}, t_{cur})$, $F_n(t) + \texttt{offset}$ is the amount of demand the mobile facility will pick up if it stays at location n during $[t, t_{cur})$ and then follows the computed optimal strategy during $[t_{cur}, T]$. (Recall, `offset` is the difference in the height between $F_n(t)$ and $max_n(t)$ at the earliest critical time $t^* \geq t_{cur}$ when $F_n(t)$ is less than, and has a greater (less steep) slope than $max_n(t)$ for some interval of time $(t^*, t^* + \epsilon)$, and the slope of $F_n(t)$ is less than (steeper than) $max_n(t)$ for some interval of time $(t^* - \epsilon, t^*)$.) This is optimal for $t \in [t_{next}, t_{cur})$ so long as $F_n(t) + \texttt{offset} \geq max_n(t)$. When this inequality does not hold, the mobile facility should leave for the location indicated by $l_{argmax_n(t)}$. This is precisely the decision made by the SRDP. Consequently, the stop strategy function $r_n(t)$ gives the decision the mobile facility should make to service the maximum

possible demand if it is at location l_n of the candidate sequence at time t. It follows that the maximum possible amount of demand may be serviced by the mobile facility following the candidate sequence $(l_1, \ldots, l_{N'})$ by starting at location l_n at time 0 for which $r_n^1(0)$ is largest. □

The SRDP creates strategy functions for a particular candidate sequence that allow the mobile facility to service the most possible demand while following a route in that candidate sequence. However, there may exist routes not in the candidate sequence. Consequently given a particular candidate sequence, a route following the stop strategy functions produced by the SRDP does not necessarily service the maximum possible (globally optimal) amount of demand over all routes. Consequently, careful consideration must be given when choosing a candidate sequence for the SRDP.

It is possible to choose a candidate sequence that guarantees the SRDP will produce an optimal route, servicing the maximum possible demand over all routes. Suppose there exists an upper bound K on the maximum number of stops the mobile facility can make during the planning horizon $[0, T]$. Choose a permutation of all the locations and create candidate sequence by repeating this permutation K times. We call this special type of candidate sequence a *master candidate sequence*. Since the mobile facility may visit at most K locations during the planning horizon, any feasible route is in this master candidate sequence. Consequently, an optimal route may be found by running the SRDP on this master candidate sequence.

For example, suppose there are three locations, $1, 2$, and 3, and that $K =$

3. Choosing the permutation $[3, 1, 2]$ generates the master candidate sequence $(3, 1, 2, 3, 1, 2, 3, 1, 2)$. The sequence of locations visited in any route of length $K = 3$ or less is a subsequence of this master candidate sequence. This master candidate sequence is not unique since it depends on the chosen permutation. If the permutation $[1, 2, 3]$ was chosen instead, the master candidate sequence would have been $(1, 2, 3, 1, 2, 3, 1, 2, 3)$.

Such an upper bound K is easy to find. For example, K may be chosen to be the greatest integer less than the length of the planning horizon divided by the minimum travel time between two locations (i.e., $K = \lfloor \frac{T}{\min_{l,l' \in L} TT_{ll'}} \rfloor$). When two locations are close to each other, this may lead to a unnecessarily long candidate sequence. In Subsection 2.3.5, we will discuss methods for reducing the runtime of the SRDP on a master candidate sequence.

2.3.3 Our Implementation of the SRDP

The implementation of the SRDP is important to ensure the creation of a unique route servicing the maximum amount of demand for a candidate sequence. For example, two functions may have the same height at time t when calculating $(max_n(t), argmax_n(t))$ giving a choice of which location to visit next. Similarly, there may be points in time when equal amounts of demand can be serviced by either staying at a location or departing for a subsequent location. We have adopted a couple of tie breaking rules to handle such situations. Suppose a mobile facility is at location l_n in the candidate route $(l_1, \ldots, l_{N'})$. Our tie breaking rules are the following:

1. Suppose, for some interval of time, the mobile facility will service the same amount of demand by either staying at location l_n, or departing for some subsequent location in the sequence. Then we assume the mobile facility will stay at location l_n for that interval of time.

2. Suppose when calculating the departure strategy function for location l_n in a candidate sequence, for some interval of time, two or more of the functions $r_{n'}(t + TT_{l_n,l_{n'}})$, for $n' > n$, have the same height. Then we define $argmax_n(t)$ to be the greatest of all such n'. (i.e., $argmax_n(t)$ indicates the location among the tied locations appearing latest in the sequence.)

These are appropriate tie breaking rules to adopt in an operational setting. The first convention says the mobile facility is not going to travel unless strictly more demand can be captured by visiting the new location. In a practical setting, this reduces wear and tear on the mobile facility. The second convention helps to reduce the runtime of the SRDP by potentially reducing the number of critical times in each stop strategy function and departure strategy function.

To compute $max_n(t)$ in Step 2, the SRDP must compute the maximum of n piecewise linear functions. In our implementation, this is done using recursion. Given two piecewise linear functions $f_1(t)$ and $f_2(t)$ with V_1 and V_2 pieces, we compute $f_{\max}(t) = \max\{f_1(t), f_2(t)\}$ in $O(V_1 + V_2)$ time. To do this we start at time T and work backwards through the critical times of $f_1(t)$ and $f_2(t)$ as follows:

Step 0: Set $t_{cur} := T$. Let t^1_{next} and t^2_{next} be, respectively, the greatest critical times of $f_1(t)$ and $f_2(t)$ less than time T. Set $f_{\max}(T) := \max\{f_1(T), f_2(T)\}$

Step 1: Set $f_{\max}(t) := \max\{f_1(t), f_2(t)\}$ for $\max\{t^1_{next}, t^2_{next}\} \leq t < t_{cur}$.

Step 2: Set $t_{cur} := \max\{t^1_{next}, t^2_{next}\}$. If $t_{cur} = 0$, terminate. Otherwise for each $i = 1, 2$, if $t^i_{next} = t_{cur}$, set t^i_{next} to be the next greatest critical time of $f_i(t)$. Return to Step 1.

During each iteration of Step 1, both functions $f_1(t)$ and $f_2(t)$ are constant on $\max\{t^1_{next}, t^2_{next}\} \leq t < t_{cur}$. Thus Step 1 is executed in constant time. Similarly, by storing the critical times of $f_1(t)$ in a sorted array or linked list, the new values t_1 and t_2 may be found in constant time.

Given $n > 2$ functions $f_1(t), f_2(t), \ldots, f_n(t)$, to find the maximum our implementation recursively computes $F_1(t) = \max\{f_1(t), \ldots, f_{\lfloor \frac{n}{2} \rfloor}(t)\}$ and $F_2(t) = \max\{f_{\lfloor \frac{n}{2} \rfloor + 1}(t), \ldots, f_n(t)\}$. After that we compute $\max\{F_1(t), F_2(t)\}$.

2.3.4 Runtime of the SRDP

Given a candidate sequence, the stop strategy function for a location is derived not only from the demand profile of that location and the stop strategy function of the next location in the candidate sequence, but also from the stop strategy functions of every subsequent location in the route. Consequently, the worst case runtime of the SRDP is likely exponential. That being noted, in practice the SRDP executes quickly. Additionally, we have not been able to produce any example where the SRDP achieves a worst case runtime that is not polynomial. Furthermore, there are several practical reasons why we believe that an exponential runtime shouldn't be expected to occur in practice.

Suppose the SRDP is run using a candidate sequence $(l_1, \ldots, l_{N'})$. Recall, the number of critical times of the remaining demand function, $F_{l_n}(t)$, of location l_n is denoted as M_{l_n}. For each $n = 1, 2, \ldots N'$, let K_n be the number of critical times of the departure strategy function $(max_n(t), argmax_n(t))$, and let P_n be the number of critical times of the stop strategy function $r_n(t)$. Since during execution of the SRDP, each computation made compares critical times from the remaining demand function and the departure strategy function, or critical times of the stop strategy functions of two or more locations, the runtime of the SRDP is dependent on how the number of critical times grows. At present, we do not have a polynomial bound on the number of critical times during execution of the SRDP.

First, we present a bound on the number of critical times in the maximum of I piecewise linear functions.

Proposition 2.3.2. *Given I piecewise linear functions, $f_k(t) : [0, T] \to \mathbb{R}^+ \cup \{0\}$ for each $i = 1, \ldots, I$ let V_i be the number of pieces of $f_i(t)$. Let $F(t) = \max_{1 \le i \le I} f_i(t)$. Then $F(t)$ has at most $I \sum_{i=1}^{I} V_i$ critical times.*

Proof. Define the ordered set of critical times,

$$P = \{0 = p_0 < p_1 < \cdots < p_J = T \mid p_j \text{ is a critical time of some } f_i(t),\ i = 1, \ldots I\}.$$

Since 0 and T are a critical times of each function $f_i(t)$, we have $|P| \le \sum_{i=1}^{I} V_i$.

Let B be the set of critical times of $F(t)$. Then B can be partitioned into two disjoint subsets, $B_1 = B \cap P$ and $B_2 = B \backslash P$. Since $B_1 \subset P$, we may write $|B_1| \le \sum_{i=1}^{I} V_i$. The critical times of $F(t)$ in the set B_2 must result from the intersection of two or more piecewise linear functions at a time that is not in P.

For each i, no critical times of $f_i(t)$ lie in any interval of the form (p_j, p_{j+1}). Thus, each $f_i(t)$ is a line segment during (p_j, p_{j+1}). Consequently, for any fixed i, $F(t) = f_i(t)$ for at most one continuous subinterval of (p_j, p_{j+1}). Thus at most $I-1$ critical times of $F(t)$ may lie in the interval (p_j, p_{j+1}). As the number of intervals of the form (p_j, p_{j+1}) is one less than $|P|$, it follows that $|B_2| \leq (I-1)\sum_{i=1}^{I} V_i$. Consequently, $|B| \leq I\sum_{i=1}^{I} V_i$. □

Since each departure strategy function is computed as a maximum of stop strategy functions shifted to the left by a travel time $TT_{l_n,l_{n'}}$, an immediate consequence of Proposition 2.3.2 is a bound on the number of critical times of the departure strategy function computed in Step 1 of the SRDP. This bound is in terms of the number of critical times of the stop strategy functions. In particular, for each $n = 1, \ldots, N$, the departure strategy function $(max_n(t), argmax_n(t))$ has at most $(N-n)\sum_{i=n+1}^{N} P_i$ critical times.

Conversely, for a given n, the maximum number of critical times in the stop strategy function $r_n(t)$ can be expressed in terms of the number of critical times of $F_{l_n}(t)$ and the departure strategy function $(max_n(t), argmax_n(t))$. During Step 2 of the SRDP, each critical time of $r_n(t)$ is produced from one of three methods:

1. A critical time of $r_n(t)$ may be a critical time of $F_{l_n}(t)$. We define B_n^1 to be the set of such critical times in $r_n(t)$.

2. A critical time of $r_n(t)$ may be a critical time of the departure strategy function $(max_n(t), argmax_n(t))$. We define B_n^2 to be the set of such critical times in $r_n(t)$ that are not also in B_n^1.

3. A critical time of $r_n(t)$ may be produced by the intersection of $F_{l_n}(t) + \texttt{offset}$ and $max_n(t)$. We define B_n^3 to be the set of such critical times that are not also in B_n^1 or B_n^2.

The following proposition relates the number of critical times of $r_n(t)$ to the number of critical times of $F_{l_n}(t)$ and $(max_n(t), argmax_n(t))$.

Proposition 2.3.3. *For each $n = 1, \ldots N$, the stop strategy function $r_n(t)$ has at most $M_{l_n} + K_n - 1$ critical times.*

Proof. Fix an n. Recall the variable `offset` from the SRDP. The SRDP may only generate a critical time of $r_n(t)$ that is neither a critical time of the remaining demand function $F_{l_n}(t)$ nor a critical time of the departure strategy function $(max_n(t), argmax_n(t))$ when, for some time t_0,

$$\begin{aligned} max_n(t_0) &= F_{l_n}(t_0) + \texttt{offset}_0 \\ max_n(t) &< F_{l_n}(t_0) + \texttt{offset}_0 \qquad \text{for } t \in (t_0 - \epsilon, t_0) \text{ and,} \\ max_n(t) &> F_{l_n}(t_0) + \texttt{offset}_0 \qquad \text{for } t \in (t_0, t_0 + \epsilon), \end{aligned}$$

where $\texttt{offset}_0$ is the value of the variable `offset` that would be used by the SRDP when computing $r_n(t)$ at time t_0. Define t_1 to be the first time after t_0 ($t_1 > t_0$) when $max_n(t_1) = F_{l_n}(t_1) + \texttt{offset}_1$, where $\texttt{offset}_1$ is some possibly different value the variable `offset` took during the computation of $r_n(t)$ for time t_1. Then the slope of $F_{l_n}(t)$ is less than (steeper than) the slope of $max_n(t)$ for each $t \in (t_1 - \delta, t_1)$, for $\delta > 0$ sufficiently small. The variable `offset` does not change when computing $r_n(t)$ for the interval (t_0, t_1). Consequently, there must be a time $t^* \in (t_0, t_1)$ where

the slope of $max_n(t)$ at t^* is greater than (less steep) the slope of $r_{l_n}(t)$ at t^*. Thus, there must be a critical time of $max_n(t)$ in the interval (t_0, t_1).

Consequently, for each critical time in B_n^3, there is a critical time of the departure strategy function $(max_n(t), argmax_n(t))$ that is not a critical time of $r_n(t)$. Furthermore, 0 may be counted as a critical time at most once. Consequently, $r_n(t)$ can have at most $M_{l_n} + K_n - 1$ critical times. □

Since $r_{N'}(t) = (F_{N'}(t), N')$, it follows that $P_{N'} = M_{l_{N'}}$. Furthermore, since $l_{N'}$ is the last location in the candidate sequence, $(max_{N'-1}(t), argmax_{N'-1}(t)) = r_{N'}(t + TT_{l_{N'-1}l_{N'}})$. Thus, $K_{N'-1} \leq P_{N'}$. From Proposition 2.3.2 we have that for each $n = 1, \ldots, N'$,

$$K_n \leq (N - n) \sum_{n'=n+1}^{N'} P_{n'} \tag{2.3.3}$$

Furthermore more, Proposition 2.3.3 states that

$$P_n \leq M_{l_n} + K_n - 1. \tag{2.3.4}$$

Combining Equations 2.4.3 and 2.4.3, we get that,

$$P_n \leq M_{l_n} + (N' - n) \sum_{n'=n+1}^{N'} P_{n'}. \tag{2.3.5}$$

Substituting Equation 2.3.5 for P_{n+1} into Equation 2.3.5 for P_n and substituting $(N' - n)$ for $(N' - (n + 1))$, we get

$$P_n \leq M_{l_n} + (N' - n)M_{l_{n+1}} + (N' - n)^2 \sum_{n'=n+2}^{N'} P_{n'} + (N' - n) \sum_{n'=n+2}^{N'} P_{n'}.$$

Continuing this expansion, we get,

$$
\begin{aligned}
P_n &\leq M_{l_n} + (N'-n)M_{l_{n+1}} + (N'-n)^2 M_{l_{n+2}} + (N'-n)M_{l_{n+2}} \\
&\quad + (N'-n)^3 \sum_{n'=n+3}^{N'} P_{n'} + 2(N'-n)^2 \sum_{n'=n+3}^{N'} P_{n'} + (N'-n) \sum_{n'=n+3}^{N'} P_{n'}, \\
&\ldots \\
&\leq M_{l_n} + \sum_{n'=n+1}^{N'-1} \left[\sum_{k=0}^{n'-n-1} \binom{n'-n-1}{k} (N'-n)^{n'-n-k} \right] M_{l_{n'}} + \sum_{k=1}^{N'-n} (N-n)^k M_{l_{N'}}.
\end{aligned}
\tag{2.3.6}
$$

Equation 2.3.6 provides an upper bound on the number of critical times in the stop strategy function $r_n(t)$ that is exponential in the length of the candidate sequence. In particular, the number of critical times in the stop strategy function for location l_n is $O(\sum_{k=0}^{N'-n}(N'-n)^k M_{l_{n+k}})$. Since every critical time must be considered during execution of the SRDP, this suggest that it is possible that the worst case runtime of the SRDP is exponential.

Despite the upper bound for the number of critical times in a stop strategy function given by 2.3.6, there are several reasons why the SRDP seems to execute quickly. Firstly, in all of our computational experiments, the number of critical times in the location strategy does not appear to grow exponentially. Furthermore, critical times of the departure strategy functions are critical times of later departure strategy functions translated forward in time by the travel time between two locations. Each time a departure strategy function is created, it often occurs that critical times of later stop strategy functions are too early in the planning horizon to cause the creation of a critical time of a departure strategy function. As a consequence,

the number of critical times in the stop strategy functions and departure strategy functions do not appear to grow exponentially in practice.

2.3.5 Reducing the Runtime of the SRDP on a Master Candidate Sequence

Given an upper bound K on the number of locations the mobile facility can visit during the planning horizon, we may create a master candidate sequence by taking any permutation of the locations in L and repeating it K times. Since the order of locations visited in any possible route is a subsequence of this master candidate sequence, the SRDP will produce the globally optimal route from this master candidate sequence. One example of such a K is the length of the planning horizon divided by the minimum travel distance between two locations. However, when two locations are very close together, this can produce a very long master candidate sequence. Since the running time of the SRDP depends on the length of this sequence, we would like to ensure K is as small as possible. The next proposition demonstrates that it is possible to always choose a candidate sequence with a length that is polynomial in the inputs of the SMFRP.

Proposition 2.3.4. *Let $K = \sum_{l \in L} M_l$. Then an optimal route exists where the mobile facility makes no more than K stops.*

Proof. Suppose not and we can produce a feasible route, $(l_n, \sigma_n, \tau_n)_{n=0}^{N}$, with more than K stops. By Lemma 2.2.1, we may also assume that for each stop n in the route, $\sigma_n < \tau_n$. Then there must exist an $n < N$ such that τ_n is not a critical time of $f_{l_n}(t)$ and σ_{n+1} is not a critical time of $f_{l_{n+1}}(t)$. Since the route is optimal, it

must be that $f_{l_n}(\tau_n) = f_{l_{n+1}}(\sigma_n)$. (If not, then a small perterbation of τ_n and σ_n would produce a route servicing more demand.) Let p be the first critical time of $f_{l_n}(t)$ after time τ_n, and let q be the first critical time of $f_{l_{n+1}}(t)$ after time σ_{n+1}. Set $\tau_n = \min\{p, q - TT_{l_n l_{n+1}}, \tau_{n+1} - TT_{l_n l_{n+1}}\}$ and $\sigma_{n+1} = \min\{p + TT_{l_n l_{n+1}}, q, \tau_{n+1}\}$. Then the mobile facility either departs stop n at the critical time p, arrives at stop $n+1$ at critical time q, or arrives at stop $n+1$ at the departure time τ_{n+1}. In the later case, stop $n+1$ can then be removed from the route while maintaining optimality by Lemma 2.2.1. Apply this modification to all stops n such that τ_n is not a critical time of $f_{l_n}(t)$ and σ_{n+1} is not a critical time of $f_{l_{n+1}}(t)$. This produces an optimal route where each time the mobile facility travels, it either departs at a critical time or arrives at a critial time. Thus, the number of stops in the route can be no more than K.

□

Proposition 2.3.4 provides a method for determining a K that is polynomial in the size of the problem. While choosing such a master candidate sequence guarantees the SRDP will produce an optimal route, there is an obvious problem with choosing $K = \sum_{l\ L} M_l$. Specifically, K can still be quite large and yield a long master candidate sequence. Without being careful, running the SRDP on such a master candidate sequence can take an unreasonably long amount of time. However, it is possible to significantly reduce the amount of time that the SRDP takes to run on a master candidate sequence.

A master candidate sequence repeats a permutation of all locations K times.

The master candidate sequence may be written as

$$(l_1, \ldots, l_{|L|}, l_{|L|+1}, \ldots, l_{2|L|}, \ldots, l_{(K-1)|L|}, \ldots, l_{K|L|}).$$

Notice that the sequence members $l_i, l_{i+|L|}, \ldots, l_{i+(K-1)|L|}$ refer to the same location for each $i \in \{1, 2, \ldots, |L|\}$. As a result, when determining the departure strategy function for location l_n, it is not always necessary to consider every subsequent location in the master candidate sequence. Proposition 2.3.5 describes which locations it is necessary to consider.

Proposition 2.3.5. *When running the SRDP on a master candidate sequence of length $K|L|$, it suffices to define $max_n(t) = \max_{n'=n+1,\ldots,n+|L|-1}\{r_{n'}(t + TT_{l_n,l_{n'}})\}$ and $argmax_n(t) \in \{n+1, \ldots, n+|L|-1\}$ for each $n = 1, \ldots, (K-1)|L|$ in Step 1 of the SRDP.*

Proof. We proceed by reverse induction. Choose an $n \in \{1, 2, \ldots, (K-1)|L|\}$ and assume the proposition is true for all $n' > n$. Furthermore, assume that $r^1_{n'}(t) \geq r^1_{n'+|L|}(t)$ for all $t \in [0, T]$ and for $n' > n$. (Intuitively, this may be seen as true since as location $l_{n'} = l_{n'+|L|}$ for each $n' \in \{n+1, n+2, \ldots, (K-1)|L|\}$, the mobile facility at location $l_{n'}$ of the candidate sequence at time t may always make the same decision as if the mobile facility was at location $l_{n'+|L|}$ of the candidate sequence at time t.) For each time $t \in [0, T]$, given the choice between departing for location $l_{n'}$ and location $l_{n'+|L|}$ in the master candidate sequence, the SRDP could specify that the mobile facility would depart for location $l_{n'}$ without servicing less demand. Therefore, computing $\max_n(t)$ as described in this proposition does not result in stop strategy functions that would allow less demand to be serviced than

the stop strategy functions that would be created by defining $\max_n(t)$ as before, provided we show that $r^1_n(t) \geq r^1_{n+|L|}(t)$. However, by assumption $r_{n'}(t) \geq r_{n'+|L|}(t)$ for every t. Thus, $max_n(t) \geq max_{n+|L|}(t)$. Therefore, the SRDP would compute $r^1_n(t) \geq r^1_{n+|L|}(t)$. □

Proposition 2.3.5 allows the SRDP to determine the stop strategy functions for a master candidate sequence by reducing the number of functions being compared each time Step 1 of the SRDP is executed. The strategy functions produced by the SRDP may still be used to produce an optimal route. While it can be assumed that K is an upper bound on the number of locations visited in an optimal route, typically that bound is not tight and the route may be expected to make significantly less than K stops. The following proposition shows that the SRDP is often able to terminate early knowing that the optimal solution may be derived from stop strategy functions computed so far.

Proposition 2.3.6. *When running the SRDP on the master candidate sequence, if for some n it is found that $r_{n+i}(t) = r_{n+|L|+i}(t)$ for $i = 0, \ldots, |L| - 1$, then an optimal route exists starting at a location $l_{n'}$ in the candidate sequence for some $n' \geq n$. Thus, the SRDP may be terminated early.*

Proof. By the previous proposition, $r_n(t)$ may be derived from only $F_{l_n}(t)$ and $r_{n'}(t)$ for $n' = 1, \ldots, |L| - 1$. Suppose the SRDP is in the process of computing $r_n(t)$ and $r_{n+i}(t) = r_{n+i+|L|}$ for $i = 0, 2, \ldots, |L| - 1$. Then the inputs the SRDP used to compute $r_n(t)$ are the same as the inputs the SRDP used to compute $r_{n+|L|}(t)$. Since the SRDP is deterministic it would find $r^1_n(t) = r^1_{n+|L|}(t)$ for all $t \in [0, T]$. Thus no

improved solution has can be found and the SRDP can terminate early. □

2.4 Characteristics of Optimal Routes

The SRDP is a method for finding an optimal route for a single mobile facility. However, the SMFRP has a special property that we may exploit to more efficiently compute an optimal route. While, multiple optimal routes may exist for an instance of the SMFRP, there exists at least one optimal route with this special property. Theorem 2.4.1 establishes that for a given instance of the SMFRP there exists an optimal route where each time the mobile facility travels from one location for another, it either departs or arrives at a time that is a critical time of the moment demand function of the two locations.

Theorem 2.4.1. *There exists an optimal route $(l_n, \sigma_n, \tau_n)_{n=0}^{N}$ where at least either the departure time τ_n is a critical time of $f_{l_n}(t)$, or the arrival time σ_{n+1} is a critical time of $f_{l_{n+1}}(t)$, for each $n = 0, 1, \ldots, N-1$.*

Proof. By Lemma 2.2.1, we may assume that $\sigma_n < \tau_n$ for $n = 0, 1, \ldots, N$. The amount of demand serviced as a function of the departure times may be written as,

$$\Delta(\tau_0, \ldots, \tau_N) = \int_0^{\tau_0} f_{l_0}(t)dt + \cdots + \int_{\tau_{n-1}+TT_{l_{n-1}l_n}}^{\tau_n} f_{l_n}(t)dt + \int_{\tau_n+TT_{l_n l_{n+1}}}^{\tau_{n+1}} f_{l_{n+1}}(t)dt + \cdots + \int_{\tau_{N-1}+TT_{l_{N-1}l_N}}^{\tau_N} f_{l_N}(t)dt. \tag{2.4.1}$$

Suppose the sequence $(l_n, \sigma_n, \tau_n)_{n=0}^{N}$ defines an optimal route and there exists an n such that τ_n is not a critical time of $f_{l_n}(t)$ and σ_{n+1} is not a critical time of $f_{l_{n+1}}(t)$. To prove the theorem it suffices to show that this route may be modified to create

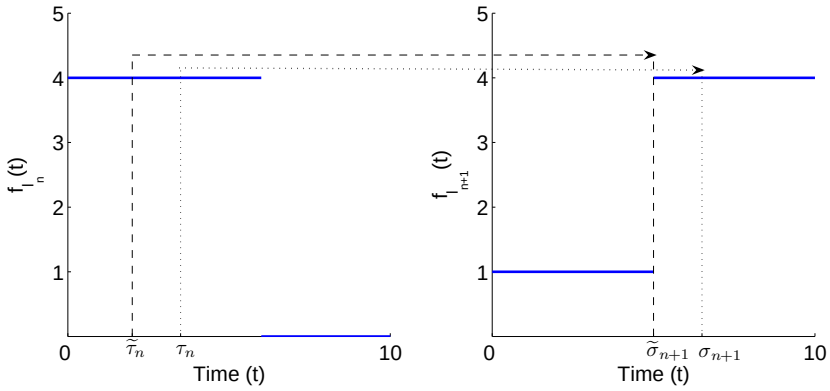


Fig. 2.6: An example where the mobile facility neither departs a location or arrives at a location at a critical time. Here, the mobile facility departs location l_n at time τ_n and arrives at location l_{n+1} at time σ_{n+1}. Notice that in this example, since $f_{l_n}(\tau_n) - f_{l_{n+1}}(\sigma_{n+1}) = 0$, an equal amount of demand is captured if the mobile facility departs location l_n at time $\tilde{\tau}_n$ and arrives at location 2 at the critical time, $\tilde{\sigma}_{n+1}$.

a new route servicing an equal amount of demand that either leaves location l_n at a critical time of $f_{l_n}(t)$, arrives at location l_{n+1} at a critical time of $f_{l_{n+1}}(t)$, or skips stop n entirely. With this in hand, the repeated application of such modifications will produce an optimal route that satisfies the theorem.

Because neither τ_n nor σ_{n+1} are a critical times of $f_{l_n}(t)$ and $f_{l_{n+1}}(t)$ respectively, it follows that the piecewise constant function $f_{l_n}(t)$ is constant in a neighborhood of τ_n and $f_{l_{n+1}}(t)$ is constant in a neighborhood of σ_{n+1}. Consequently, $\Delta(\tau_0, \ldots, \tau_N)$ is differentiable in a neighborhood of τ_n and,

$$\frac{\partial}{\partial \tau_n}\Delta(\tau_0, \ldots, \tau_N) = f_{l_n}(\tau_n) - f_{l_{n+1}}(\tau_n + TT_{l_n l_{n+1}}) \tag{2.4.2}$$

$$= f_{l_n}(\tau_n) - f_{l_{n+1}}(\sigma_{n+1}) \tag{2.4.3}$$

Because the route is assumed optimal, it follows that

$$f_{l_n}(\tau_n) - f_{l_{n+1}}(\sigma_{n+1}) = 0. \tag{2.4.4}$$

Since $f_{l_n}(t)$ and $f_{l_{n+1}}(t)$ are piecewise constant functions, Equation (2.4.4) holds true for the interval of time around τ_n where both $f_{l_n}(t)$ and $f_{l_n}(t+TT_{l_n l_{n+1}})$ remain constant. Figure 2.6 shows an example of such a route. If Equation (2.4.4) were not true, some small perturbation of τ_n would yield a route that services more demand. (For example, if $f_{l_n}(\tau_n) - f_{l_{n+1}}(\sigma_{n+1}) > 0$, then changing the departure time from τ_n to $\tau_n + \epsilon$, for some sufficiently small $\epsilon > 0$, will increase the amount of demand serviced along the route.) Thus by Equation (2.4.4), the rate demand may be serviced at location l_n when the mobile facility departs at time τ_n is equal to the rate demand may be serviced at location l_{n+1} when the mobile facility arrives at time σ_{n+1}.

Let $\tilde{p}$ be the latest critical time of $f_{l_n}(t)$ before time τ_n and let $\tilde{q}$ be the latest critical time of $f_{l_{n+1}}(t)$ before time σ_n. Since both $f_{l_n}(t)$ and $f_{l_{n+1}}(t + TT_{l_n l_{n+1}})$ are piecewise functions, it follows from Equation 2.4.4 that $f_{l_n}(t) = f_{l_{n+1}}(t + TT_{l_n l_{n+1}})$ for all t in the interval $\max\{\tilde{p}, \tilde{q} - TT_{l_n l_{n+1}}\} \leq t \leq \tau_n$. Thus, if the mobile facility departs from stop n at time $\tilde{\tau}_n = \max\{\tilde{p}, \tilde{q} - TT_{l_n l_{n+1}}, \sigma_n\}$ instead of time τ_n and otherwise follows the original route as specified, the mobile facility will service an equal amount of demand. Consequently, this modified route remains optimal.

If $\tilde{\tau}_n = \tilde{p}$ or $\tilde{\tau}_n = \tilde{q} - TT_{l_n l_{n+1}}$, then either the mobile facility departs Stop n or arrives at Stop $n+1$ at a critical time and we are done. If $n \geq 1$ and $\tilde{\tau}_n = \sigma_n$, then the mobile facility may instantaneously depart Stop n the moment it arrives

and capture an equal amount of demand. By Lemma 2.2.1, we may modify the route, creating an optimal route without stops where the mobile facility departs the instant it arrives. Finally, if $\tilde{\tau}_n = 0$, then the mobile facility must be at the first stop in the route, so $n = 0$. In this case, a new optimal route may be defined by starting at location l_1 at time 0 and following the remainder of the route. In either case, a new optimal route may be defined that skips stop n, proving the Theorem. □

Consider the example in Figure 2.1 consisting of two locations separated by two units of travel time. Suppose an optimal route satisfying Theorem 2.4.1 exists and somewhere in the route the mobile facility travels from Location 1 to Location 2. Then the mobile facility must either depart Location 1 at a critical time of Location 1 that allows it to arrive at Location 2 by time 10 (i.e., it must leave at a time in the set $\{0, 2, 8\}$), or it must depart Location 1 so that it arrives at Location 2 at a critical time of Location 2 before time 10 (i.e., it must leave at a time in the set $\{1, 5, 8\}$). Furthermore, the proof of Theorem 2.4.1 shows that a mobile facility may be assumed never to depart a stop the instant it arrives. Therefore, the mobile facility will never leave Location 1 for Location 2 at time 0. Similarly, it may be assumed that the mobile facility will not leave Location 1 for Location 2 at time 8, since that would cause the mobile facility to arrive at Location 2 at time 10, the end of the planning horizon. Thus, the mobile facility can be assumed to depart Location 1 for Location 2 at a time in the set $\{1, 2, 5\}$ and arrive at Location 2 at a time in the set $\{3, 4, 7\}$ in an optimal route.

Similarly, if the mobile facility travels from Location 2 to Location 1 in the

route, then it may be assumed to depart Location 2 at a time in the set $\{3, 6, 7\}$ and arrive at Location 1 at a time in the set $\{5, 8, 9\}$. Note that the optimal route described in Section 2.2.1 satisfies the results of Theorem 2.4.1.

An immediate consequence of Theorem 2.4.1 is that there exists a discrete set of times when the mobile facility can be expected to depart from or arrive at each location. Furthermore, the size of this set is polynomial in the number of critical times of each location. In particular, the set of times S_l when the mobile facility can be assumed to depart from or arrive at location l in an optimal solution is,

$$S_l = (\cup_{l' \neq l} \{q_j^{l'} + TT_{l'l} | j = 0, 1, \dots M_{l'},\ q_j^{l'} + TT_{l'l} < T\}) \tag{2.4.5}$$

$$\cup\, (\cup_{l' \neq l} \{q_j^{l'} - TT_{ll'} | j = 0, 1, \dots M_{l'},\ q_j^{l'} + TT_{ll'} > 0\}) \tag{2.4.6}$$

$$\cup\, \{q_j^l | j = 0, 1, \dots, M_l\}. \tag{2.4.7}$$

The set S_l contains the $M_l + 1$ critical times of $f_l(t)$. Since 0 and T are common critical times of all moment demand functions, and since S_l only contains times in $[0, T]$, each other moment demand function $f_{l'}(t)$ can contribute at most $M_{l'}$ times to S_l. Thus,

$$|S_l| \leq 2 \sum_{l' \neq l} M_{l'} + M_l + 1. \tag{2.4.8}$$

2.4.1 IP Formulation

The results from Theorem 2.4.1 the allow SMFRP to be modeled as a compact integer program (IP). Let $S_l = \{s_0^l < s_1^l < \cdots < s_{|S_l|}^l\}$ denote the ordered set of times a mobile facility can either depart from or arrive at location l. For each location $l \in L$ and each $k \in \{0, 1, \dots, |S_l| - 1\}$, define the binary variable x_k^l to be

1 if the mobile facility is at location l during $[s_k^l, s_{k+1}^l)$ and zero otherwise. Let d_k^l be the demand captured by a mobile facility at location l during $[s_k^l, s_{k+1}^l)$. (i.e., $d_k^l = f_l(s_k^l)(s_{k+1}^l - s_k^l)$.) The IP can then be formulated as follows,

$$\text{maximize} \sum_{l \in L} \sum_{k=0}^{|S_l|-1} d_k^l x_k^l \tag{2.4.9}$$

$$x_k^l + x_{k'}^{l'} \leq 1 \qquad \text{for all } l, l' \in L,\ k, k' \in \{0, \ldots, |S_l| - 1\} \tag{2.4.10}$$

$$\text{such that } s_{k+1}^l + TT_{ll'} \geq s_{k'}^{l'} \text{ and } s_{k'+1}^{l'} + TT_{l'l} \geq s_k^l$$

$$x_k^l \in \{0, 1\} \qquad \text{for all } l \in L,\ k = 0, 1, \ldots, |S_l| - 1 \tag{2.4.11}$$

Above, the objective function is given by 2.4.9 and records the amount of demand captured by the mobile facility if it follows the route prescribed by the binary variables x_k^l. Constraint 2.4.10 ensures the mobile facility does not visit two locations during two time periods if it is not possible to leave one location at the end of one time period and arrive and the other location at or before the start of the other time period. Constraint 2.4.11 species the variable as binary.

2.5 Finding an Optimal Route for a Mobile Facility in Polynomial Time

Theorem 2.4.1 gives a discrete set of times when a mobile facility may be assumed to depart from a location or arrive at a location each time the mobile facility travels in an optimal route. Using this result, it is possible to find the optimal solution for a single mobile facility in polynomial time. To do so, we define a directed acyclic graph, which we call the routing graph, and find the longest path between

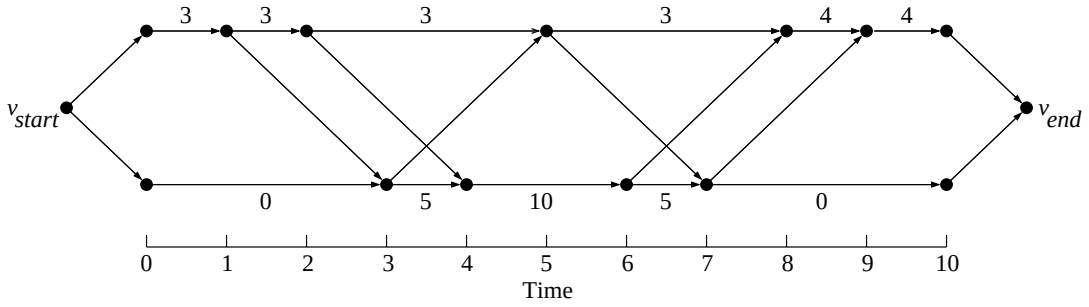


Fig. 2.7: The routing graph for routing a single mobile facility from the event points and locations from Figure 2.1. Nodes v_{start} and v_{end} are explicitly labeled. There is one node in the upper row for each time in S_1 and one node in the bottom row for each time in S_2. A timeline is drawn below for reference. The node for each critical time are above that point on the timeline. Each arc with nonzero length is labeled. Unlabeled arcs have length zero.

a designated source and sink node in the graph. Again let $S_l = \{s_0^l < s_1^l < \dots\}$ denote the ordered set times a mobile facility can either depart from or arrive at location l. For each location l and each time $s_k^l \in S_l$, we create a vertex v_k^l in the graph. If $s_k^l < T$, an arc is created from vertex v_k^l to vertex v_{k+1}^l and given a weight equal to the amount of demand that the mobile facility could service if it was at that location l during the time period $[s_k^l, s_{k+1}^l]$. For each pair of times s_k^l and $s_{k'}^{l'}$, if the mobile facility could leave location l at time s_k^l and arrive at location l' at time $s_{k'}^{l'}$, and either time s_k^l or $s_{k'}^{l'}$ are critical times of $f_l(t)$ and $f_{l'}(t)$ respectively, then an arc with weight 0 is created from vertex v_k^l to vertex $v_{k'}^{l'}$. A source vertex v_{start} is added along with arcs with weight 0 from v_{start} to each v_0^l for each $l \in L$. Finally a sink node v_{end} is added along with arcs with weight 0 from each node $v_{|S_l|}^l$ to v_{end} for each $l \in L$.

Since each arc originates from a vertex representing an earlier time than the time represented by the vertex at which the arc terminates, no cycles exist in the graph. Thus, a path through this network from v_{start} to v_{end} can be thought of as representing a route for the mobile facility. Each arc on the path from a vertex v_k^l to vertex v_{k+1}^l represents the mobile facility providing service at location l from time s_k^l to time s_{k+1}^l. The remaining arcs on the path from a vertex v_k^l to a vertex $v_{k'}^{l'}$, with $l \neq l'$, represent periods when the mobile facility is traveling between locations l and l'. The length of the path equals the amount of demand the mobile facility can capture along this route. Figure 2.7 gives the routing graph for the example SMFRP scenario presented in Figure 2.1, where the two locations are two units of travel time apart.

The optimal route can be computed by finding the longest path through the network from v_{start} to v_{end}. We do this using a dynamic program that we call the *single mobile facility longest path algorithm* (SMFLPA). For each vertex v in the graph, we compute the length of the longest path from that vertex to v_{end}, denoted as $\texttt{length}(v)$, and the first arc that should be traversed on that path, denoted as $\texttt{next}(v)$.

The Single Mobile Facility Longest Path Algorithm:

Step 0: Set $\texttt{length}(v_{end}) := 0$. Create a sorted list of the remaining vertices in the network in decreasing order of the time associated with each vertex and with v_{start} at the end of the list.

Step 1: Remove the vertex v at the top of the list. Set

$$\texttt{length}(v) := \max_{(v,v')}\{\texttt{weight}(v,v') + \texttt{length}(v')\}$$

and set $\texttt{next}(v)$ to equal the vertex v' for which this maximum is achieved. Ties can be broken arbitrarily.

Step 2: If the list is empty, terminate. Otherwise, repeat Step 1.

The routing graph is a directed acyclic graph containing $2+\sum_{l\in L}|S_l|$ vertices. Vertex v_{end} has no outgoing arcs and each other vertex has at most $|L|$ outgoing arcs. Therefore the routing graph has at most $|L| + |L|\sum_{l\in L}|S_l|$ arcs. Since the routing graph is a directed acyclic graph, the sorted list of vertices in Step 0 can be computed using a topological sort, which has a running time that is linear in the number of vertices of the graph plus the number of edges of the graph. Therefore, the running time of Step 0 is $O(|L|\sum_{l\in L}|S_l|) = O(|L|^2\sum_{l\in L}M_l)$. Each time each time the maximum is computed in Step 1, the above algorithm makes at most $|L|$ calculations, one for each arc exiting vertex v. Step 1 is executed $1+\sum_{l\in L}|S_l|$ times. Thus, executing the remaining parts of the SMFLPA takes running time $O(|L|\sum_{l\in L}|S_l|) = O(|L|^2\sum_{l\in L}M_l)$. Thus the single mobile facility dynamic program has running time $O(|L|^2\sum_{l\in L}M_l)$, which may be viewed as either linear in the number of arcs in the routing graph, or polynomial in the number of locations and critical times of the functions $f_l(t)$.

2.6 Computational Experience

In the previous two sections, we presented two algorithms for computing the optimal solution to the SMFRP. The SRDP executes quickly but does not have a guaranteed polynomial runtime, while the SMFLPA will always execute in polynomial time. Since both of these algorithms compute the optimal solution, in this section we evaluate the runtimes of the two algorithms when solving a variety of simulated data sets. In all data sets, locations are chosen as points in the plane and the travel time between two locations is the Euclidean distance between them

The first type of scenario simulates, at a high level, demand profiles that we might expect to occur in practice, such as in routing a portable cellular base stations for a given day. We refer to these scenarios as *realistic scenarios*. Each realistic scenario simulates part of a single day, beginning at 7AM and ending at midnight. The demand profile of each location in the scenario simulates the forecasted demand from a single event that might occur in practice, such as a sporting event, rush hour traffic, a county fair, a convention, etc. Next, we define a time window during which demand from that type of event is generated. For each event of a type, we choose at random a time t_{start} when demand generation begins and a later time t_{end} when demand generation ends inside that time window, although, we specify that each location must generate demand for service for at least 1 hour. (i.e., $t_{end} - t_{start} \geq 1$.) During that time period, $f_l(t)$ gradually increases to a maximum level of between 4 and 10, remains at that maximum level for a period of time, and then gradually decreases to zero. The rate of increase and decrease are also chosen at random.

Outside of the time period between t_{start} and t_{end}, demand for service at location l is generated at rate zero.

The demand profiles at the locations in a realistic scenarios share a specific structure. Namely, the demand is generated during a single interval of time and during that interval demand gradually increases to, remains at, and then decreases from a maximum rate of demand generation back down to zero. To see how these two algorithms preform in a less structured data set, we created a second type of scenario where the demand profile of each location displays less structure. We refer to these scenarios as *mathematically challenging scenarios*. The planning horizon of each of these scenarios begins at time 0 and ends at time 100. The demand profile for a location a mathematically challenging scenario is generated as follows. The number of pieces in the piecewise constant moment demand function $f_l(t)$ for a location l was chosen at random from a specified range. To generate a wide range of demand profiles, we specified that each location would have at least 5 and no more than 40 pieces. To find the length of each step of the piecewise constant function $f_l(t)$, we assign each step a random number between 0 and 1. These numbers are then normalized so that their sum equals the length of the planning horizon. Each normalized number is taken to be the length of the corresponding step. The rate demand is generated by that location (i.e., the height of $f_l(t)$) during that time is 0 with probability p, and otherwise chosen randomly between 0 and 10. We chose $p = 0.5$ to provide a reasonably large amount of time when each location does not require demand for service.

Table 2.1 gives runtime results for the two algorithms for finding the optimal

Data Set	Number of Locations	SMFLPA Runtime	SRDP Runtime	Ave. Demand Serviced
MC0	5	0.0006	0.0019	368.098
MC1	10	0.0281	0.0808	433.671
MC2	20	0.4612	3.5862	500.879
MC3	40	4.4695	26.5724	553.538
R0	5	0.0001	0.0001	45.784
R1	10	0.0001	0.0001	62.832
R2	20	0.0001	0.3300	74.767
R3	40	0.0219	3.8556	87.808

Tab. 2.1: Each row in this table gives the averaged results from 25 data sets generated with the same parameters. Rows containing results from realistic data sets are labeled with the letter 'R' followed by a number identifying the data sets. Similarly, the rows containing mathematically challenging data sets are labeled with the letters 'MC' followed by a number identifying the data sets. The second and third columns give the runtime in seconds of SMFLPA and the SRDP. The average demand serviced in each scenario is in the rightmost column.

solution to the SMFRP. In our computational experiments, the SMFLPA found the optimal solution for almost every data set in less time that the SRDP. The only exceptions were data sets where both algorithms were reported to execute in under 0.0001 seconds. These results confirm our analysis, and show that the SMFLPA is the quicker algorithm to solve the SMFRP. However, as we will discuss in Section 2.7, the SRDP does have value in that it may more easily be extended than the SMFLPA to more general cases when the demand functions are not piecewise constant.

2.7 Solving Variants of the SMFRP

The methodologies developed in this chapter can be extended to solve several variants of the SMFRP. Below we describe several such extensions.

2.7.1 The Addition of a Depot

We have created the SMFRP with the assumption that the mobile facility may start at any location at time 0 and end at any location at time T. Under this assumption if it is also the case that the mobile facility must start and end the day at a specific location or depot, the mobile must depart early enough to arrive at the first location in the route at time 0 and will not return to the depot until after leaving the last location in the route at time T. However, the model can be easily modified if the mobile facilities may not depart the depot until time 0 and return to the depot by time T. Let TT_{Dl} be the travel time from the depot D to location l and let TT_{lD} be the travel time from location l to the depot D. In this variant, the mobile facility would not be able to provide service to location l before time TT_{Dl} or after time $T - TT_{lD}$. To modify the instance of the SMFRP, replace each moment demand function $f_l(t)$ with,

$$\tilde{f}_l(t) = \begin{cases} f_l(t), & TT_{Dl} \leq t \leq T - TT_{lD}; \\ 0, & \text{otherwise.} \end{cases}$$

Solving the SMFRP using these modified demand profiles will create the best route when the mobile facility may only be at a location l between time TT_{Dl} and $T - TT_{lD}$. Consequently, this route is also an optimal route if the mobile facility must start at the depot D at time 0 and return by time T.

2.7.2 Applying the SMFRP in a Stochastic Environment

We considered the moment demand function for location l, $f_l(t)$, to represent the deterministic rate at which demand can be serviced by a mobile facility at

location l at time t. However, this model may also be applied in a somewhat stochastic scenario. Specifically, we may assume each step j of $f_l(t)$ represents a Poisson arrival process with average arrival rate h_j^l during the period of time $[p_j^l, p_{j+1}^l)$. The expected number of arrivals that a mobile facility at location l during step j would see is $h_j^l(p_{j+1}^l - p_j^l)$. Consequently, if a mobile facility is at location l from time σ to time τ, the expected number of arrivals that mobile facility will see is also

$$\int_\sigma^\tau f_l(s)ds.$$

In other words, the expected number of arrivals seen by the mobile facility in this stochastic interpretation of the SMFRP is equal to the amount of demand serviced in the SMFRP as initially described. Thus, the problem can be interpreted stochastically as maximizing the expected number of arrivals the mobile facility could service along a route.

2.7.3 Other Types of Moment Demand Functions

In this chapter, we have described two heuristics for generating routes for mobile facilities when the moment demand functions are piecewise constant. However, under certain conditions these heuristics may be extended to more general types of moment demand functions. Below we describe how to extend both heuristics when the moment demand functions are continuous. We first describe how to extend the SMFLPA when we know all times t such that $f_l(t) = f_{l'}(t + TT_{ll'})$ for each pair of location l and l'. We then describe how to extend the SRDP without knowing this collection of times t.

Suppose the mobile facility follows a route $(l_n, \sigma_n, \tau_n)_{n=0}^N$. The amount of demand serviced can be written as a function of the departure times $(\tau_n)_{n=0}^N$ by Equation 2.4.1. Consequently, when a mobile facility departures location l_n in an optimal solution, τ_n must satisfy the equation,

$$\frac{\partial}{\partial \tau_n} \Delta(\tau_0, \tau_1, \ldots, \tau_N) = 0.$$

Since,

$$\frac{\partial}{\partial \tau_n} \Delta(\tau_0, \tau_1, \ldots, \tau_N) = f_{l_n}(\tau_n) - f_{l_{n+1}}(\tau_n + TT_{l_n l_{n+1}}),$$

we have that,

$$f_{l_n}(\tau_n) = f_{l_{n+1}}(\tau_n + TT_{l_n l_{n+1}}).$$

In other words, if the mobile facility departs location l_n for location l_{n+1}, then the rate demand is being generated at the moment the mobile facility departs location l_n must be equal to the rate demand is being generated the moment it arrives at location l_{n+1}.

If for each pair of location l and l' all times t when $f_{l_n}(\tau_n) = f_{l_{n+1}}(\tau_n + TT_{l_n l_{n+1}})$ were known, the SMFLPA could be adapted to compute the optimal route. To do so, for each location l define the set S_l containing all times t such that $f_l(t) = f_{l'}(t + TT_{ll'})$ or $f_l(t) = f_{l'}(t - TT_{l'l})$. Construct the routing graph using the times in S_l as described in Section 2.5. Running the SMFLPA on this routing graph will find the optimal route.

The ability to extend the SMFLPA to find the optimal route when moment demand functions are continuous, and thus the runtime of the algorithm, depends on the ability to compute ahead of time all intersection points of $f_{l_1}(t)$ and $f_{l_2}(t + TT_{l_1 l_2})$

for each pair of locations l_1 and l_2. In a practical setting, this may not be possible. However, the SRDP may be extended without knowing these intersection points beforehand. To extend the SRDP, given two continuous functions $g(t)$ and $h(t)$ and a time t_1 we need to be able to compute the latest time t_0 when $g(t) \geq h(t)$ for $t_0 \leq t \leq t_1$ and $h(t) > g(t)$ for $t_0 - \epsilon < t < t_0$ for sufficiently small $\epsilon > 0$. The time t_0 has one of two meanings:

1. t_0 is the time when $h(t)$ crosses $g(t)$, and graph of $h(t)$ is higher than the graph of $g(t)$ to the left of t_0, and the graph of $g(t)$ lower than the graph of $h(t)$ between time t_0 and time t_1.

2. If $h(t) = g(t)$ for some period of time $t_0 < t < t_0 + \epsilon$, the t_0 is the time when the two graphs diverge, with $g(t) < h(t)$ for some period of time the left of t_0.

Most often, this can be computed numerically by finding the first intersection of $g(t)$ and $h(t)$ before time t_1. Consequently, it may be more appropriate to extend the SRDP to solve the SMFRP when moment demand functions take on more general forms.

Suppose the SRDP is given a candidate sequence $(l_1, \ldots, l_{N'})$ and each moment demand function f_l(t) is continuous instead of piecewise constant. To extend the SRDP, again define the remaining demand function for each location l by $F_l(t) = \int_t^T f_l(s)ds$. In this case, we do not have the predetermined set of critical times for the moment demand function of each location. Computing $(max_l(t), argmax_l(t))$ may still be accomplished by beginning at time T and selecting the location $l_{n'}$ after l_n in the candidate sequence that would allow the mobile facility to leave location

l_n at the latest time t and arrive at location $l_{n'}$ by time T (i.e., the location $l_{n'}$ that maximizes $T - TT_{l_n l_{n'}}$ with $n' > n$). Ties between two distinct locations $l_{n'}$ and $l_{n''}$ (i.e., $l_{n'} \neq l_{n''}$) may be broken by computing the first time $t_0 < T$ where $F_{l_{n'}}(t+TT_{l_n l_{n'}}) = F_{l_{n''}}(t+TT_{l_n l_{n''}})$ for $t \geq t_0$, and $F_{l_{n'}}(t+TT_{l_n l_{n'}}) \neq F_{l_{n''}}(t+TT_{l_n l_{n''}})$ for $t_0 - \epsilon < t < t_0$ for some sufficiently small $\epsilon > 0$. Ties between two entries in the candidate sequence, $l_{n'}$ and $l_{n''}$, referring to the same physical location (i.e., $l_{n'} = l_{n''}$) may be resolved arbitrarily.

Suppose $(max_n(t), argmax_n(t))$ have been determined from time t_1 through time T, and that $F_{l_{n'}}(t - TT_{l_n l_{n'}}) \geq F_{l_{n''}}(t - TT_{l_n l_{n''}})$ for all $n'' = n+1, \ldots, N$. Compute the latest time t_0 before time t_1 where for some $n'' > n$, the graph of $F_{l_{n''}}(t-TT_{l_n l_{n''}})$ crosses the graph of $F_{l_{n'}}(t-TT_{l_n l_{n'}})$ from above on the left, to below or equal to on the right. Then set $(max_n(t), argmax_n(t)) = (F_{l_{n'}}(t - TT_{l_n l_{n'}}), n')$ for $t_0 \leq t < t_1$. Iterating this process will produce $(max_n(t), argmax_n(t))$.

Just as before, the SRDP should initialize the stop strategy function for location l_n as $r_n^1(t) = F_{l_n}(t)$ and $r_n^2(t) = n$ for $t > \max_{n'=n+1,\ldots,N} T - TT_{l_n,l'_n}$. Again, initialize the variable `offset` to equal zero. Suppose the $r_n(t)$ has been determined from time t_1 through time T. Then the SRDP will make one of two decisions.

Case 1 ($r_n^2(t_1) = n$)*:* Compute the latest time t_0 before time t_1 when $max_n(t)$ crosses $F_{l_n}(t) + \texttt{offset}$ from above of the left to below on the right. If t_0 exists, define $r_n(t) = (F_{l_n}(t) + \texttt{offset}, n)$ for $t \in (t_0, t_1)$ and define for time t_0, $r_n(t_0) = (max_n(t_0), argmax_n(t_0))$. Otherwise, define $r_n(t) = (F_{l_n}(t) + \texttt{offset}, n)$ for $t \in [0, t_1)$

Case 2 ($r_n^2(t_1) > n$)*:* Find the latest time t_0 after time t_1 when the graph of slope of $F_{l_n}(t)$, (i.e., $f_{l_n}(t)$) crosses the graph of slope of $max_n(t)$ (i.e., $f_{l_{argmax_n(t)}}(t)$) from above on the left, to below on the right. (Note that the slope of both functions is constant on this interval by the choice of t_0 and t_1.) If such a time t_0 exists, define $r_n(t) = (max_n(t), argmax_n(t))$ for $t \in (t_0, t_1)$, assign $\texttt{offset} = max_n(t_0) - F_{l_n}(t_0)$, and define $r_n(t_0) = (F_{l_n}(t_0) + \texttt{offset}, n)$. If such a time t_0 does not exist, then define $r_n(t) = (max_n(t), argmax_n(t))$ for $t \in [0, t_1)$.

Iterating this decision process during Step 2 of the SRDP will produce the stop strategy functions that yield the optimal solution when the moment demand functions are continuous.

2.7.4 Addition of Relocation Costs

In some applications, the services provided by a mobile facility can be equated with revenue, such as when operating mobile post offices. The operators of the mobile facility in this context might wish to maximize the service provided or the revenue brought in by mobile facilities minus the cost of operating the mobile facilities, which can include relocation costs. The SMFLPA presented in Section 2.5 can be adapted to accommodate these relocation costs. To do so, for each arc from a node representing location l at some time to a node representing location l' at another time, set the weight of the arc to be the negative of the cost of relocating the mobile facility from location l to location l'. Each time the mobile facility relocates, the path representing the route of the mobile facility in the routing graph

must traverse an arc with weight equal to the negative cost of relocating. Thus, the length of a path in this modified routing graph will equal the service provided (or revenue generated) minus the relocation costs of moving the mobile facility. The longest path through this modified routing graph will produce the optimal route for this variant.

2.8 Conclusion

In this chapter we have introduce the SMFRP. The goal of the SMFRP is to maximize the amount of demand serviced by a single mobile facility over a planning horizon. We have shown that despite the seemingly complex nature of the SMFRP, the optimal route for the single mobile facility may be found in polynomial time when the demand functions are piecewise linear using the SMFLPA. This is a direct result of the fact that it may be assumed that the mobile facility either departs a location or arrives at a location at a critical time, each time the mobile facility travels between two locations. We have also presented a second method for solving the SMFRP by using the SRDP with a master candidate sequence that is not dependent on this result. Our theoretical results allow the SRDP to be modified to significantly reduce runtime. Our computational results demonstrate the SMFLPA executes more quickly than the SRDP when moment demand functions are piecewise constant. However, the SRDP may be more easily extended to situations where the moment demand functions take on more general forms. We have also shown that the SMFRP may be extended to several additional variants that may arise in applications.

3. THE MOBILE FACILITY ROUTING PROBLEM

3.1 Introduction

Mobile facilities are used to provide many types of services. In the previous chapter, we discussed different types of mobile facilities that share common operational characteristics, including portable cellular base stations, mobile post offices, trailer mounted radar speed monitors, mobile medical facilities, mobile kitchens, and mobile medical clinics. These mobile facilities only provide services while at a location, and can be rapidly transported between locations; however, no service may be provided by a mobile facility in transit. We discussed the situation when routing a single mobile facility (i.e., the SMFRP) and developed exact algorithms for it. While these algorithms find the optimal solution for routing a single mobile facility, additional factors must be considered when dealing with multiple mobile facilities operating in a region. Specifically, it is important to model interactions between the mobile facilities. For example, two portable cellular base stations positioned close to each other may both be capable of providing cellular phone coverage to the same customer. Consequently, when routing multiple mobile facilities it is important to separate the events generating demand for service from the locations where a mobile facility may be positioned to provide that service. Furthermore, by separating the

events generating demand from the locations where a mobile facility may be positioned to provide service, one must give consideration to the rate capacity of each mobile facility (i.e., the maximum rate a mobile facility may provide service). It is possible that a subset of events nearby a mobile facility may generate demand at a higher rate than the rate capacity of the mobile facility.

In this chapter we study the problem of effectively deploying a limited fleet of mobile facilities when demand for service changes over time. We call this problem the mobile facility routing problem (MFRP). The MFRP seeks to determine routes for a fleet of mobile facilities to maximize the amount of demand serviced in a continuous-time planning horizon. In the MFRP, there is a discrete set of locations where a mobile facility may be positioned to provide service and a discrete set of event points generating demand. These event points could be towns, events or individuals generating demand for service. While at a location, a mobile facility may service demand from a given subset of event points nearby. Mobile facilities may depart one location for another location at any time, although a mobile facility cannot service demand while in transit. In the MFRP, we assume the locations where a mobile facility may visit, the travel times between locations, the set of event points generating demand, and the demand for service at each event point over the entire planning horizon are known ahead of time. We show that the problem is NP-hard, and then describe several heuristics for generating effective routes for mobile facilities. We also provide some insight into the types of demand profiles for which mobile facilities are most effective when compared to the use of static (or non-movable) facilities.

The remainder of this chapter is organized as follows. We will give a formal introduction to the MFRP in Section 3.2 and show the problem is NP-hard. We then describe several heuristics for the MFRP in Section 3.3 which utilize the SMFLPA from Section 2.5 as a building block. Next we provide a large set of computational experiments in Section 3.4. Section 3.5 provides concluding remarks.

3.2 The Mobile Facility Routing Problem

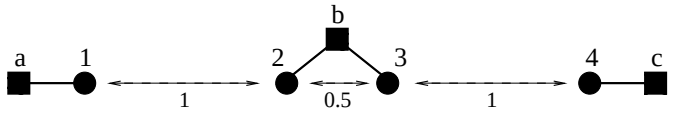
The MFRP is set in a continuous-time planning horizon, which starts at time 0 and ends at time T. The objective is to determine a set of routes, one for each mobile facility in a set M, that together maximize the amount of demand serviced. There is a discrete set of predetermined locations L, and a mobile facility may only provide service while at one of these locations. At any time during the planning horizon, a mobile facility may depart from its current location and travel to any other location in L. The time a mobile facility takes to travel from a location l to a location l' is denoted $TT_{ll'}$. Without loss of generality, we assume these travel times to satisfy the triangle inequality.

Demand for service is generated by a set of discrete event points E. For each event point $e \in E$, there is a nonnegative, real-valued *moment demand function* $d_e(t)$ that describes the *rate* demand is being generated by event point e at time t. As $d_e(t)$ describes the rate at which demand is being generated, the total demand generated by event e between times σ and τ is given by $\int_\sigma^\tau d_e(s)ds$.

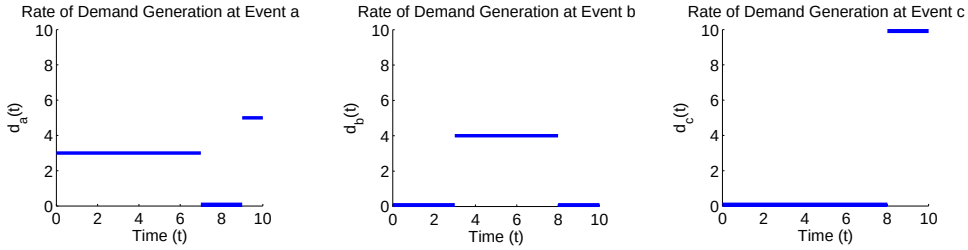
A mobile facility at a location $l \in L$ is capable of servicing demand from

a specified subset of events E_l. The subset E_l may consist of all events within a specified distance of location l, or possibly some other subset of events defined by service constraints particular to an application. However, each mobile facility has a rate capacity C, which is the maximum rate at which the mobile facility can service demand. For each location $l \in L$, we define $f_l(t)$ to be the cumulative rate of demand for service is being generated at time t by all events that can be covered from location l (i.e., $f_l(t) = \sum_{e \in E_l} d_e(t)$). Thus, from time σ to time τ a mobile facility at location l will service at most $\int_\sigma^\tau \min\{C, f_l(s)\} ds$ units of demand. A mobile facility could service less than this amount if demand from one or more of the events in E_l is also being serviced by another mobile facility during that time period.

A route for a mobile facility is a sequence of *stops*, $(l_n, \sigma_n, \tau_n)_{n=0}^N$, where l_n is the location visited during stop n, σ_n is the arrival time at stop n, and τ_n is the departure time from stop n. In other words, service is provided at location l_n during $[\sigma_n, \tau_n]$. We specify that a mobile facility leaving stop $n-1$ travels directly to stop n and immediately begins service upon arrival. Thus in any route, $\sigma_n = \tau_{n-1} + TT_{l_{n-1} l_n}$ for $n = 1, \ldots N$. We do not assume that the mobile facility must begin and end its route at a depot, but rather that each mobile facility always begins at stop 0 of its route at time 0 and ends at stop N at time T. The requirement that each mobile facility start and end at a depot is equivalent to adding the constraint that no demand may be serviced at a location l during $[0, TT_{Dl})$ and $(T - TT_{lD}, T]$ where TT_{Dl} and TT_{lD} are the travel times between the depot D and location l. This constraint may be enforced by modifying the definition of $f_l(t)$ so that $f_l(t) = \sum_{e \in E_l} d_e(t)$ for



(a)



(b)

Fig. 3.1: Panel (a) shows a configuration of the locations and event points. Event points are represented with squares and locations are represented with circles. The locations are positioned in a straight line with the travel times between neighboring locations shown below the dashed arrows. A solid line connects each location to each event point it can cover. Panel (b) displays the moment demand function, $d_e(t)$, for each event point in the configuration.

$t \in [TT_{Dl}, T - TT_{lD}]$ and $f_l(t) = 0$ otherwise.

We will assume $d_e(t)$ is a piecewise constant function that may only assume nonnegative values during the planning horizon $[0, T]$. Since any reasonable function for modeling $d_e(t)$, such as continuous functions, may be approximated arbitrarily closely by a piecewise constant function, this assumption is not very restrictive.

Figure 3.1 displays the locations and event points for an instance of the MFRP problem. This example contains four locations, $L = \{1, 2, 3, 4\}$, and three event points, $E = \{a, b, c\}$. Suppose each mobile facility in this example has a rate capacity of 10. A route for a single mobile facility, for example, could start at

location 1 and service demand from event point a from time 0 until time 1.5. At this moment the mobile facility could depart location 1 for location 3. It would arrive at location 3 at time 3 and could provide service from there for event point b until time 7. The mobile facility could then immediately depart location 3 and travel to location 4, servicing demand from event point c from time 8 until the end of the planning horizon. In this route, 40.5 units of demand are serviced. The mobile facility services demand from event point a at rate 3 during $[0, 1.5]$, servicing 4.5 $(= 3 \times 1.5)$ units of demand. It services demand from event point b at rate 4 during $[3, 7]$, servicing 16 units of demand. Finally, the mobile facility services demand at rate 10 from event point c during $[8, 10]$, servicing 20 units of demand. Note that this is an optimal route for a single mobile facility with a rate capacity of 10.

A second mobile facility with a rate capacity of 10 could then be sent on a route starting at location 1, providing service for event point a during $[1.5, 6]$. At time 6, it could travel to location 2, arriving at time 7 and servicing demand from event point b during $[7, 8]$. The mobile facility could then be sent back to location 1 to provide service to event point a during $[9, 10]$. In total, this second route services 22.5 units of demand.

Together, these two routes together define an optimal solution to this instance of the MFRP with two mobile facilities, each with a rate capacity of 10. However, this solution is not unique. An equal amount of demand would be serviced if the first mobile facility departs location 1 for location 3 at any time $t \in [0, 1.5]$ and the second mobile facility arrives at location 1 at time t instead of time 1.5.

In this example, the optimal route for a single mobile facility was also one of

the two routes in an optimal solution for two mobile facilities. However, in general this need not be the case.

3.2.1 Formulating the MFRP as an Infinite Dimensional Mixed Integer Program.

The MFRP may be formulated as an infinite dimensional mixed integer program (IDMIP). The program seeks to find functions $\tilde{h}_{mle}(t)$ that describe the rate demand from event point e is serviced by mobile facility m at location l at time t. Let $\tilde{x}_{ml}(t)$ be 1 if mobile facility m is at location l at time t, and 0 otherwise. Let $L_e = \{l : e \in E_l\}$ be the set of locations from where service may be provided to event point e. The MFRP can be formulated as follows:

$$\text{Maximize} \quad \sum_{m\in M}\sum_{l\in L}\sum_{e\in E_l}\int_0^T \tilde{h}_{mle}(s)ds \tag{3.2.1}$$

$$\text{subject to:} \quad \sum_{e\in E_l}\tilde{h}_{mle}(t) \le C\tilde{x}_{ml}(t) \qquad \text{for each } m\in M,\ l\in L,\ t\in[0,T] \tag{3.2.2}$$

$$\sum_{m\in M}\sum_{l\in L_e}\tilde{h}_{mle}(t) \le d_e(t) \qquad \text{for each } e\in E,\ t\in[0,T] \tag{3.2.3}$$

$$\tilde{x}_{ml}(t)+\tilde{x}_{ml'}(t') \le 1 \qquad \text{for each } m\in M,\ l\in L, t, t'\in[0,T], \text{ and } t-TT_{l'l} < t' < t+TT_{ll'} \tag{3.2.4}$$

$$\tilde{h}_{mle}(t) \ge 0 \qquad \text{for each } m\in M, l\in L, e\in E_l \text{ and } t\in[0,T] \tag{3.2.5}$$

$$\tilde{x}_{ml}(t) \in \{0,1\} \qquad \text{for each } m\in M,\ l\in L,\ t\in[0,T] \tag{3.2.6}$$

The objective function maximizes the amount of demand serviced. If mobile facility m is at location l at time t, Constraint (3.2.2) ensures that the rate capacity of mobile facility m is not violated. If mobile facility m is not at location l at time

t, then $\tilde{x}_{ml}(t) = 0$, which in turn forces $\tilde{h}_{mle}(t) = 0$ for each event point $e \in E_l$. Constraint (3.2.3) ensures that for each time t, the rate demand is serviced from event point e is no greater than the rate demand is generated by event point e. Constraint (3.2.4) enforces the travel times. If mobile facility m is at location l at time t, then this constraint says mobile facility m may not service demand from any other location l' at time t' if either mobile facility m could not leave location l' at time t' for location l and arrive by time t, or mobile facility m could not leave location l at time t and arrive at location l' by time t'. There are an infinite number of such constraints, since these constraints are defined for a continuum of times.

While there has been considerable work on infinite dimensional linear programs, to our knowledge, not much is known in terms of solution methods for IDMIPs. Consequently, the focus of this chapter is on discussing heuristics for solving the MFRP. We note however, that to compare the solution quality of our heuristics, we consider a particular discretization of the IDMIP for the MFRP that limits arrival and departure times to a finite set of times.

3.2.2 Related Work

Facility location is a well studied field within operations research. For background on facility location theory, see the excellent monographs by [19] and [36]. To the best of our knowledge, the MFRP has not been studied before. However, there has been considerable work in the literature on mobile facilities (i.e., a facility that can be relocated in the solution to a problem). We review some of the most closely related problems below.

One well studied problem is the dynamic facility location problem that seeks to locate facilities to service all demand during a multi-period planning horizon at minimum cost. (For references, see [3], [10], [23], [41], [49], and [50].) The multi-period planning horizon consists of a discrete set of periods. Facilities with sufficient capacity must be sited to service all demand during each of these periods. Typically, a cost is incurred to relocate, remove, or add a facility however, the problem does not give specific consideration to the time it takes to relocate a facility. This may be well suited for long planning horizons when the facilities are infrequently moved, or for scenarios where the time it takes to relocate a facility is relatively short compared to the planning horizon. The continuous-time planning horizon in the MFRP provides an ability to consider problems when the time to relocate a mobile facility may constitute a significant portion of the planning horizon, as is the case, for example, in the telecommunications setting.

Mobile facilities can be expensive to own and deploy. Consequently, fleets of mobile facilities can be of limited size. In such cases, the limited fleet of mobile facilities may lack the sufficient capacity to cover all demand. Furthermore, some demand may be unduely difficult to cover. For example, to cover an event point generating a small amount of demand for a short period of time that is far from all other demand points, a mobile facility would need to spend significant time traveling to cover that small amount of demand. In other applications, such as routing mobile post offices, demand not covered can be serviced by fixed facilities, even though this is less convenient to the customer. In this sense the MFRP resembles a dynamic version of the maximum covering location problem (MCLP). Given a set of demand

points, and a set of potential facility sites, the MCLP seeks to site a fixed number of facilities to cover the maximum possible amount of demand. Typically, demand is considered to be covered if it is within a given service radius of a facility. [19], chap. 4 is a good survey for background on the MCLP. Other work on the MCLP includes [11], [13], [33], and [40]. Below we show that the MCLP may be considered as a special case of the MFRP.

A number of publications have studied covering path problems. These problems can be separated into two classes. The first class contains those problems where all demand must be covered, such as [14], [15], and [17]. The second class contains those problems that seek to maximize coverage, sometimes as one objective in a multiobjective optimization problem, such as [16] and [18]. The problems in this second class resemble the MFRP in that they seek to determine a collection of paths that maximize demand coverage. However, unlike the MFRP, demand in these problems only need be covered at some point along the path. In the MFRP, the demand covered is dependent on the period of time the mobile facility spends at each location.

[8] and [21] studied a problem related to mobile facility location. Similar to the MFRP, this problem is set in a continuous-time planning horizon. Given a collection of continuously-moving points in the plane, the objective is to find paths for each of k continuously moving facilities so that at any time t, the distances from the points to the closest facility minimizes a given metric. Since facilities move continuously and are always providing service, the facilities in this problem differ from those used in the MFRP, where mobile facilities may only provide service while stationary.

Additionally, demand in the MFRP does not originate from moving customers and varies over time.

Furthermore, unlike the problem studied by [8] and [21] and the dynamic facility location problem, the value of the demand serviced by a mobile facility at a location l from an event point e in the MFRP is always independent of the location l (provided $e \in E_l$). In other words, if that demand was instead serviced by a different mobile facility and a second location l', the contribution of that demand to the objective function would remain unchanged. This is a result of the objective of the MFRP, which is to maximize the total demand serviced. The other problems referenced here in mobile facility location generally seek to minimize costs that may be dependent on both the location l from which the facility and the demand point e being serviced.

3.2.3 Computational Complexity of the MFRP

The maximum covering location problem (MCLP) can be viewed as a special case of the MFRP, thereby proving that the MFRP is NP-hard. Given a set of demand points J, each with demand d_j, a set of potential facility sites I, and the distances between each facility site and each demand point, the MCLP seeks to site a given number of facilities to cover the maximum possible amount of demand. A facility at a site is capable of providing coverage to all demand points within a given radius r of that site. The MCLP is well known to be NP-hard.

Theorem 3.2.1. *The MFRP is NP-hard.*

Proof. We give a polynomial reduction of the decision version of the MCLP (named Decision-MCLP) to the decision version of the MFRP (Decision-MFRP), which is to determine if the given facilities can be sited to cover at least X units of demand. Similarly, Decision-MFRP is to determine if the given mobile facilities can be routed to service at least X units of demand. Note, that it is easy to see that Decision-MFRP is in NP.

Given an instance of Decision-MCLP, we transform each facility site $i \in I$ into a location $l(i)$. The length of the planning horizon, T, is defined to be half the minimum travel time between two locations. As a result, each mobile facility will be able to visit at most one location during the planning horizon. Each demand point $j \in J$ is transformed into an event point $e(j)$. The event point $e(j)$ will generate demand at the constant rate of $\frac{d_j}{T}$ for the entire length of the planning horizon. Thus, over the planning horizon, event point $e(j)$ generates an equal amount of demand as demand point j. Event point $e(j)$ will be in the set $E_{l(i)}$ in Decision-MFRP if and only if demand point j could be covered by a facility at site i in Decision-MCLP. A mobile facility is created in Decision-MFRP for each facility in Decision-MCLP. The rate capacity of each mobile facility is defined to be $\sum_{j \in J} \frac{d_j}{T}$. With this rate capacity, each mobile facility would be capable of servicing all demand in the scenario, if it were serviceable from one location.

Any solution to the instance of Decision-MCLP can be realized as a solution to Decision-MFRP with an equal objective value. To create this solution, for each site i where a facility is placed in a given solution to Decision-MCLP, we place a mobile facility for the entire planning horizon at each location $l(i)$. Whenever demand from

a demand point j is being covered, we assign all demand from event point $e(j)$ for the entire planning horizon to the closest mobile facility.

Note that a mobile facility may visit at most one location in any solution to this instance of the MFRP. To transform a solution of this MFRP instance into a solution to the MCLP that covers at least as much demand, a facility is placed at site i if location $l(i)$ is visited by a mobile facility. If this places less than the total number of facilities, then position the remaining facilities at arbitrary, unused sites. If demand point j is covered in this solution, then by optimality event point $e(j)$ is being serviced by one or more mobile facilities during the planning horizon $[0, T]$. Thus the demand covered in this solution to the MCLP is no less than the demand serviced in the solution of the MFRP instance. Therefore this is a solution to the MCLP that covers at least as much demand. Consequently, Decision-MFRP is NP-complete and thus, the MFRP is NP-hard. □

3.3 Heuristics for the MFRP

In this section, we discuss heuristics for the MFRP with multiple mobile facilities, which is NP-hard. The SMFLPA introduced in Section 2.5 gives a method for calculating an optimal route for a single mobile facility in polynomial time. This algorithm was derived using Theorem 2.4.1, which says that each time a mobile facility relocates, it may be assumed to either depart from or arrive at a location at a critical time. Unfortunately, Theorem 2.4.1 does not extend to the general case with multiple mobile facilities. Recall the example problem in Figure 3.1 and the optimal

routes for two mobile facilities. In any optimal solution, without loss of generality, the second mobile facility leaves location 1 at time 6 and arrives at location 2 at time 7; however, neither is time 6 a critical time of location 1 nor is time 7 a critical time of location 2. This occurs because the first mobile facility departs location 3 at time 7. If one were to create a function representing the amount of demand generated by event b that is not serviced by the first mobile facility, then it may be observed there is a discontinuity at time 7 caused by the departure of the first mobile facility from location 3 at time 7. This discontinuity, which is not a critical time of $f_2(t)$, causes the route of the second mobile facility to depart location 1 at time 6 and arrive at location 2 at time 7. One may view time 6 as the critical time 8 of location 4 translated backward in time by $TT_{l_3 l_4}$ and $TT_{l_1 l_2}$. It is possible to construct an example where given $|M|$ mobile facilities, a facility leaves from or arrives at a location at a time that is a critical time of some location translated forward or backward (or a combination of both) of a critical time by the travel times between $|M|$ pairs of locations.

In general, the potential departure times for a mobile facility from a location l are dependent on the critical times of the locations in L and on the demand from the events in E_l serviced by the other mobile facilities. Each time a mobile facility relocates from one location to another in an optimal solution, it may be assumed that the mobile facility either departs or arrives either at a critical time or at a time when another mobile facility is arriving or departing from some, possibly different location. If not, by a similar argument to the proof of Theorem 2.4.1, the mobile facility could either leave earlier or later and service more demand. For the MFRP,

there seems to be no obvious way to generate a polynomially bounded set of discrete times when a mobile facility may depart from or arrive at a location.

It remains an open question whether an optimal solution to the MFRP always exists in which each time a mobile facility travels, it departs a location at a time that is a translation of a critical time forward and backward by a sequence of at most $|M|$ travel times. Even if this were true, the number of potential departure times would still be exponential in $|M|$, since the number of different sequences $(TT_{l_{n_k} l_{m_k}})_{k=0}^{|M|}$ is exponential in $|M|$. Furthermore, even if there was a method to reduce this set of potential departure times so that its size is polynomially bounded, the problem of determining the optimal routes for a fleet of mobile facilities is still extremely difficult. For example, in perhaps the simplest case where each mobile facility could arrive at each location only at time 0 and depart only at time T, the end of the planning horizon, we have observed that solving the MFRP is no easier than solving the MCLP.

Consequently, our strategy has been to develop several heuristics for generating routes for the MFRP, which we describe below. We then apply a local search method to improve the routes. As each heuristic executes, a record is kept of the demand not being serviced in the instance of the MFRP. We define the variable $\tilde{d}_e(t)$, for each $e \in E$, to be the rate at which demand that is not serviced is being generated at event point e at time t. Similarly, for each location $l \in L$, we define the variable $\tilde{f}_l(t)$ to be the cumulative rate demand that is not serviced is generated by the event points in E_l. Thus, $\tilde{f}_l(t) = \sum_{e \in E_l} \tilde{d}_e(t)$ for each $t \in [0, T]$. Furthermore, we define $\tilde{d}_{me}(t)$ to be the rate demand is being serviced by mobile facility m from event point

e at time t and $\tilde{f}_{ml}(t)$ to be the rate that mobile facility m is servicing demand from location l at time t.

We use the SMFLPA when generating routes. Given the subset of locations L_m that mobile facility m may visit, the SMFLPA can be used to generate a route that services the maximum possible amount of demand currently not serviced from these locations. In other words, the SMFLPA can be seen as a function that takes in the remaining serviceable demand at each location in L_m, the travel times between those locations, and the rate capacity of a mobile facility, and returns a sequence of locations for the mobile facility to visit, the arrival and departure times at each location in the sequence, and functions $\tilde{f}_{ml}(t)$ describing the amount of demand serviced by mobile facility m from location l at each time t.

3.3.1 Demand Assignment

After a route is created for a mobile facility using the SMFLPA, we still must specify which event points are being serviced while the mobile facility is at each stop on its route, and the rate demand is being serviced at each time t from each of these event points. The phase of our heuristics that determines the demand from each event point serviced by each mobile facility is called *demand assignment*. For stop n of the route, the demand assignment phase of each of our heuristics initially set $\tilde{d}_{me}(t) = 0$ for all $e \in E_{l_n}$ and all times t between the arrival time σ_n and the departure time τ_n. The event points in E_{l_n} are sorted according to one of four criteria, which we specify below. Taking the first event point e in the list, we define $\tilde{d}_{me}(t)$ at each time $t \in [\sigma_n, \tau_n]$ to be the minimum of the rate demand currently

not serviced is being generated at event point e (i.e., $\tilde{d}_e(t)$), and the unused rate capacity of the mobile facility. Accordingly, we then subtract $\tilde{d}_{me}(t)$ from $\tilde{d}_e(t)$. This process is repeated for each event point in the sorted list until either the list has been exhausted or the rate capacity of the mobile facility has been reached for the entire duration of the stop. The four sorting criteria we consider are:

Sort 1: Sort the event points in increasing order of the number of locations from which a mobile facility could provide service to the event point (i.e., $|L_e|$). For event points serviced by the same number of locations, sort them in increasing order of the amount of demand not serviced during Stop n.

Sort 2: Sort the event points in increasing order of the number of locations from which a mobile facility could provide service to the event point (i.e., $|L_e|$). For event points Serviced by the same number of locations, sort them in decreasing order of the amount of demand not serviced during Stop n.

Sort 3: Sort the event points in increasing order of the amount of demand not serviced during Stop n.

Sort 4: Sort the event points in decreasing order of the amount of demand not serviced during Stop n.

Intuitively, Sort 1 and Sort 2 first assign demand from those event points that can be serviced from fewer locations, making it less likely that the route of another mobile facility will be able to service demand from those event points. Sort 1 and Sort 3 attempt to first assign demand from those event points with small amounts of not

serviced demand by sorting the event points by increasing order of the amount of demand not serviced. This would leave more event points with large amounts of demand that is not serviced, which would hopefully allow other routes to be created having stops where larger amounts of demand is serviced. Alternatively, sorting the event points in decreasing order of the amount of demand not serviced in Sort 2 and Sort 4 would leave more event points with demand that is not serviced, hopefully allowing any route created to minimize travel time, during which a mobile facility services no demand.

3.3.2 Sequential Routing for the MFRP

The sequential routing heuristic generates a route for one mobile facility at a time. Upon initialization, this heuristic sets $\tilde{d}_e(t) = d_e(t)$ for each $e \in E$ and $\tilde{f}_l(t) = f_l(t)$ for each $l \in L$. Each subsequent stage of the heuristic generates a route for a single mobile facility considering all locations (i.e., $L_m = L$) using the SMFLPA. Thus, each time a route is generated, it will service the maximum possible amount of demand that was previously not serviced. Since the SMFLPA only determines locations for the mobile facility to visit and an arrival and departure time for each stop in the route, after each time a route is created using the SMFLPA the demand assignment phase is used to determine how demand will be assigned from event points to the stops in the route. Finally, the variables $\tilde{d}_e(t)$ and $\tilde{f}_l(t)$ are updated to reflect the demand serviced by the new route. The steps of the sequential routing heuristic are outlined below.

Step 0 (Initialization): For each $e \in E$, initialize $\tilde{d}_e(t) := d_e(t)$. For each $l \in L$,

initialize $\tilde{f}_l(t) := f_l(t)$.

Step 1 (Route Determination): Choose the next mobile facility to be routed. Run the SMFLPA, using the capacitated remaining serviceable demand functions $\tilde{f}_l^C(t) = \min\{C, \tilde{f}_l(t)\}$ for each location l, to determine the location visited, and the arrival and departure times for each stop in the route.

Step 2 (Demand Assignment): Compute the demand serviced from each event point for each stop in the route following the method for demand assignment described in Subsection 3.3.1.

Step 4 (Demand Update): Update $\tilde{d}_e(t)$ for each $e \in E$ and $\tilde{f}_l(t)$ for each $l \in L$ to reflect the demand that is covered by this route.

Step 5: If all mobile facilities have been assigned a route or all demand is serviced, terminate. Otherwise return to Step 1.

3.3.3 Generating Routes with an Insertion Heuristic

The second heuristic we present is an insertion heuristic. This heuristic associates a set of locations L_m with each mobile facility m. Each mobile facility m will only visit locations in L_m on its route. Each set L_m is initially empty. At each stage of the heuristic, a location $\tilde{l}$ is considered for insertion into each set L_m. When location $\tilde{l}$ is being considered for insertion into L_m, the SMFLPA is used to calculate the demand serviced along the best route for mobile facility m, assuming it may only visit the locations in L_m and location $\tilde{l}$ and assuming mobile facility m may only service demand it is already servicing, or demand that is currently not

serviced at these locations. If the route followed by at least one mobile facility may be improved by allowing it to visit location $\tilde{l}$, then location $\tilde{l}$ is then added to the set L_m for the mobile facility m whose route shows the greatest improvement by the addition of location $\tilde{l}$. The routes of the other mobile facilities remain unchanged during this iteration. The process is repeated until no more improvements may be found. The details of the insertion heuristic are outlined below.

Step 0 (Initialization): For each $e \in E$, $l \in L$, and $m \in M$, initialize $\tilde{d}_e(t) := d_e(t)$, initialize $\tilde{f}_l(t) := f_l(t)$, initialize the set L_m to be empty, initialize $\tilde{d}_{me}(t) := 0$, and initialize $\tilde{f}_{ml}(t) := 0$. Furthermore, initialize the set $\tilde{L} := L$.

Step 1 (Insertion Selection): Select the location $\tilde{l} \in \tilde{L}$ with the largest total amount of demand not serviced during the planning horizon. For each mobile facility m with $\tilde{l} \notin L_m$, we do the following. For each location $l \in L_m \cup \{\tilde{l}\}$, define the function $\delta_l(t) = \min\{C, \tilde{f}_l(t) + \tilde{f}_{ml}(t)\}$. Calculate how much demand could be serviced by mobile facility m by executing the SMFLPA using the functions $\delta_l(t)$ as the amount of demand that can be serviced from location l by this mobile facility and the travel times between the locations in $L_m \cup \{\tilde{l}\}$.

Step 2 (Route Selection): If it was found in Step 1 that the route of no mobile facility can be improved by allowing it to visit location $\tilde{l}$, then remove location $\tilde{l}$ from the set $\tilde{L}$ and goto Step 5. Otherwise, the route that may be most improved by visiting location $\tilde{l}$ is changed to follow the new route calculated in Step 1. In the case where this maximum improvement may be realized by inserting location l into several routes, we choose the route that was first

considered. Location $\tilde{l}$ is added to L_m.

Step 3 (Demand Assignment): Demand from event points is assigned to each stop in the route following the method for demand assignment described in Subsection 3.3.1. If no demand remains that may be serviced from location $\tilde{l}$ or $\tilde{l} \in L_m$ for every mobile facility m, then location $\tilde{l}$ is removed from $\tilde{L}$.

Step 4 (Demand Update): Update $\tilde{d}_e(t)$ for each $e \in E$ and $\tilde{f}_l(t)$ for each $l \in L_m$ to reflect the demand that is now covered or no longer covered by this new route.

Step 5: If $\tilde{L}$ is empty, terminate. Otherwise, return to Step 1.

3.3.4 Local Search for the MFRP

We now describe a local search algorithm that looks for improvements to routes generated by the above heuristics. A naive method for looking for improvements could examine solutions in a local search neighborhood defined by the following two types of exchanges:

- For a stop n in a route r, try changing the location visited in stop n to a different location, maintaining either the arrival and departure times for stop n, or the departure time from stop $n-1$ and the arrival time at stop $n+1$.

- Let r_1 and r_2 be a pair of routes. Delete stop n_{r_1} from route r_1, so that route r_1 travels from stop $n_{r_1}-1$ to stop $n_{r_1}+1$. (Thus, the demand that was serviced during stop n_{r_1} is not longer serviced.) Next, delete all of route r_2, and

recalculate route r_2 using the single mobile facility dynamic program. (This could improve route r_2 by allowing it to service some of the demand previously serviced during stop n_{r_1} on route r_1.) Finally, delete the remaining stops in route r_1 and recalculate route r_1 using the single mobile facility dynamic program.

Implementing such a local search procedure produces two distinct problems. The first type of exchange will never improve the routes generated by either heuristic. The insertion heuristic associates with each route a set of locations that may be visited, and then uses the SMFLPA to calculate a route that services the maximum amount of demand that is not serviced while visiting only the stops in that set. It terminates when for each route, no improvement may be found by allowing that route to visit another location. Consequently, each route generated by the insertion heuristic cannot be improved by changing the location visited in a stop to any other location. In the sequential heuristic, when each route was created, it serviced the maximum amount of demand that was not serviced at that time. Since demand being serviced at some stage of the sequential routing heuristic will remain serviced during the execution of the remaining steps of the sequential routing heuristic, the demand not serviced immediately prior to the generation of any route includes all demand that is not serviced by the final routes produced by the sequential heuristic. Consequently, after the sequential heuristic, if a stop in a route is replaced with a stop at a different location, the mobile facility will service no more demand after this exchange than it does on its route before the exchange.

Secondly the running time of a local search algorithm that considers every exchange of the second type blows up wildly. This is because the SMFLPA is executed each time the second type of exchange is considered. In our experience, searching over every exchange of the second type could take an unreasonably long time as most exchanges in the neighborhood are fruitless. Motivated by identifying types of neighborhoods that can yield improvements, we look at a similar special type of local search neighborhood where a lower bound on the improvement may be computed rapidly.

Suppose we have two distinct routes, route r_1, $(l_n^1, \sigma_n^1, \tau_n^1)_{n=0}^{N_1}$, and route r_2, $(l_n^2, \sigma_n^2, \tau_n^2)_{n=0}^{N_2}$, and that route r_1 was generated prior to route r_2 in the sequential routing heuristic. All demand serviced in route r_2 was not being serviced when route r_1 was generated. Consequently, route r_1 cannot be improved by servicing demand that is serviced along route r_2. However, it may be possible to improve route r_2 if the mobile facility following route r_2 was allowed to service some of the demand serviced in route r_1.

For each pair of routes in a solution to the MFRP generated by one of our heuristics, our local search algorithm looks for opportunities to improve the solution by removing a stop from route r_1, inserting it into route r_2, and filling in the gap in route r_1 with demand that is not serviced. Specifically, if stop $(l_{n_0}^1, \sigma_{n_0}^1, \tau_{n_0}^1)$ is removed from route r_1 and inserted into route r_2, we temporarily assume the arrival time and departure time of stop $(l_{n_0}^1, \sigma_{n_0}^1, \tau_{n_0}^1)$ are preserved. Since travel times are assumed to obey the triangle inequality, this produces a unique time t_0 when the mobile facility following route r_2 must deviate from route r_2 to arrive at location $l_{n_0}^1$

Fig. 3.2: An example of an exchange in a local search algorithm. Boxes above each time-line each represent a stop along a route. The solid black box represents a stop moved from route r_1 to route r_2. The demand lost in the exchange is shaded by horizontal and vertical crossing lines. The demand added in the exchange is shaded with diagonal crossing lines.

at time $\sigma^1_{n_0}$ and a unique time t_1 when the mobile facility returns to route r_2 after departing location $l^1_{n_0}$ at time $\tau^1_{n_0}$. The demand previously serviced along route r_2 between time t_1 and time t_2 is no longer serviced by this mobile facility.

Route r_1 no longer contains stop n_0, which allows the mobile facility following this route to stay longer at stops $n_0 - 1$ and $n_0 + 1$, possibly servicing more demand. To calculate the amount of additional demand serviced, we define the set S of potential departure times from stop $n_0 - 1$. By a similar argument to the proof of Theorem 2.4.1, we can assume the mobile facility will either depart stop $n_0 - 1$ at a discontinuity of $f_{l_{n_0-1}}(t)$ or $\tau^1_{n_0-1}$, or arrive at stop $n_0 + 1$ at a discontinuity of $f_{l_{n_0+1}}(t)$ or σ_{n_0+1}. Specifically, S will contain the discontinuities of $\tilde{f}_{l_{n_0-1}}(t)$ during the interval of time between $(\tau^1_{n_0-1}, \sigma^1_{n_0+1} - TT_{l_{n_0-1}l_{n_0+1}})$, the discontinuities of $\tilde{f}_{l_{n_0+1}}(t)$ during $(\tau^1_{n_0+1} + TT_{l_{n_0-1}l_{n_0+1}}, \sigma^1_{n_0})$, and the times $\tau^1_{n_0-1}$ and $\sigma^1_{n_0+1} - TT_{l_{n_0-1}l_{n_0+1}}$. Sort S and denote the sorted order as $S = \{s_0 < s_1 < \cdots < s_K\}$. To efficiently compute the best time to leave stop $n_0 - 1$ for stop $n_0 + 1$ after time τ_{n_0}, we initially assume the

mobile facility departs stop $n_0 - 1$ at time τ_{n_0-1}. We then work forward through the times s_k in S, computing the amount of demand that can be serviced if the mobile facility departs stop $n_0 - 1$ at time s_k and arrives at stop $n_0 + 1$ at time $s_k + TT_{l_{n_0-1}l_{n_0+1}}$. To make this computation for time s_k, it suffices to compute the additional demand serviced from stop $n_0 - 1$ between times s_{k-1} and s_k, and the amount of demand no longer serviced during stop $n_0 + 1$. (i.e., the demand serviced from stop $n_0 + 1$ between times $s_{k-1} + TT_{l_{n_0-1}l_{n_0+1}}$ and $s_k + TT_{l_{n_0-1}l_{n_0+1}}$.) This procedure gives an efficient method for computing the locally optimal time the mobile facility should depart stop $n_0 - 1$ for stop $n_0 + 1$. The steps of the procedure are as follows:

Step 0: Initialize $k := 1$, $D^- := 0$, $D^+ := \int_{s_0+TT_{l_{n_0-1}l_{n_0+1}}}^{\sigma^1_{n_0+1}} \tilde{f}_{l_{n_0}+1}(s)ds$. Then initialize $k^{max} := 0$ and $D^{max} := D^+$.

Step 1: Update the variable $D^+ := D^+ - \int_{s_{k-1}+TT_{l_{n_0-1}l_{n_0+1}}}^{s_k+TT_{l_{n_0-1}l_{n_0+1}}} \tilde{f}_{l_{n_0+1}}(s)ds$. Update the variable $D^- = D^- + \int_{s_{k-1}}^{s_k} \tilde{f}_{l_{n_0-1}}(s)ds$.

Step 2: If $D^- + D^+ > D^{max}$, set $D^{max} := D^- + D^+$ and $k^{max} := k$.

Step 3: If $k = K$, terminate. Otherwise, set $k := k + 1$.

At the end of this procedure, $s_{k^{max}}$ gives the locally optimal time that the mobile facility following should depart stop $n_0 - 1$ for stop $n_0 + 1$.

To see if this exchange produced an improvement, one may wish to look if the demand serviced along route r_1 and route r_2 after the exchange is greater than the demand serviced along the two routes before the exchange. This is equivalent to

computing the difference in the amount of demand serviced after the exchange that was not serviced before the exchange, and the amount of demand serviced before the exchange that is not serviced after the exchange. The additional amount of demand serviced after the exchange is the amount demand serviced in route r_1 between times $\tau^1_{n_0-1}$ and $\sigma^1_{n_0+1}$. The amount of demand that is no longer serviced after the exchange is the amount of demand previously serviced along route r_2 between times t_0 and t_1. (We note this may count the demand from an event point twice if it was covered by route 2 before the exchange and route r_1 after the exchange. However, this demand will cancel out in the final calculation.) The difference in these two is the net change in the amount of demand serviced. If this net change is positive, this is an improvement in the route. If an improvement is found, we delete route r_2 and stop n_0 from route r_1, leaving the remainder of route r_1 intact, and recompute route r_2 using the SMFLPA to service the maximum amount of remaining demand. After demand assignment to this new route, we delete the remaining stops in route r_1 and recompute route r_1 using the SMFLPA. However, this operation is computationally expensive. Thus, we only will perform such an operation after considering all possible exchanges and selecting the one that potentially yields the most improvement.

Notice since the arrival time and departure time of stop n_0 of route r_1 are preserved when it is inserted into route r_2, even if the difference in the demand serviced after the exchange is negative, an overall increase in the amount of demand service could be found by recomputing route r_1 and route r_2 using the SMFLPA in the same manner as if an improvement was found. For this reason, we define a

negative threshold value Δ and consider all exchanges where the change in the total demand serviced along the two routes after the exchange is greater than Δ.

To start the local search procedure, we define an empty tabu list $\mathcal{T}$ that will keep track of exchanges of a stop between routes that are not to be considered again. Each iteration of the local search algorithm searches over all such exchanges not in the tabu list of a stop between two routes and selects the exchange yielding the greatest improvement. If the greatest difference in the amount of demand serviced is less than Δ, we terminate. Otherwise, the two routes are recalculated as follows. We first delete the stop n_0 that was moved from route r_1 to route r_2 and delete all stops originally in route r_2 (i.e., $(l_n, \sigma_n^2, \tau_n^2)_{n=0}^{N_1}$), temporarily leaving the remainder or route r_1 intact. Next, we recompute route r_2 using the SMFLPA to generate a new route r_2 servicing the maximum amount of currently not serviced demand. Demand assignment is performed for the new route r_2 using the procedure described in Section 3.3.1. Now, to recalculate route r_1 we delete the remaining stops in route r_1 and then recompute route r_1 using the SMFLPA to service the maximum amount of demand currently not serviced. Demand assignment is performed for the new route r_1 using the procedure described in Section 3.3.1. If the total demand serviced along the new route r_1 and route r_2 is not more than before the exchange, we revert to the two routes before the exchange and add that exchange to $\mathcal{T}$. Otherwise, the new routes are kept. The local search algorithm continues to iterate until no further improvement can be found. The details of the local search procedure are outlined below.

Step 0: Initialize $r_1 := -1$, $r_2 := -1$, $n_0 := 0$, $\delta := \Delta$. Initialize $\mathcal{T}$ to be empty.

Step 1: For each pair of routes, a and b, and for each stop n in route a, do the following. Consider moving the stop from route a to route b, preserving its arrival and departure time. Compute the demand that may no longer be serviced along route b if stop n must be visited and store it as δ_1. Compute the locally optimal time the mobile facility following route a must leave stop $n-1$ for stop $n+1$. Store the additional demand serviced along route a at stops $n-1$ and $n+1$ as δ_2. If $\delta < \delta_2 - \delta_1$, set $r_1 := a$, $r_2 := b$, $n_0 := n$, and $\delta := \delta_2 - \delta_1$.

Step 2: If $\delta \leq \Delta$, terminate. Otherwise, delete stop n_0 from route r_1, leaving the remainder of route r_1 intact, and delete route r_2. Recalculate route r_2 using the SMFLPA and allowing the mobile facility to visit any location. Run demand assignment for each stop in the new route r_2. Next, delete the remaining stops in route r_1 and recalculate route r_1 using the SMFLPA and allowing the mobile facility to visit any location. Run demand assignment for each stop in the new route r_1. If this does not produce an improvement, add the exchange of stop $(l_{n_0}, \sigma^a_{n_0}, \tau^a_{n_0})$ from route r_1 to route r_2 to $\mathcal{T}$ and revert to the previous routes. Return to Step 1.

Given two routes generated by the sequential algorithm, route r_1 and route r_2, if route r_1 was generated after route r_2, no improvement will be found by trying to remove a stop from route r_1 and insert it into route r_2. This is because both route r_1 and route r_2 were generated to service the maximum amount of demand that

was not serviced at the time of their generation. Thus route r_2 will not improve by the addition of a stop from route r_1, and route r_1 will not improve by the addition of demand currently not serviced. Thus, when running the local search algorithm on routes generated from the sequential routing heuristic, we only consider pairs of routes where route r_1 was generated before route r_2.

3.4 Computational Results

In the previous section, we described two heuristics to generate solutions for the MFRP, and a local search procedure that attempts to improve a given solution. During execution, both these heuristics employ one of four sorting orders, which are defined in Section 3.3, to assign demand coverage. We have developed a variety of simulated data sets to test these heuristics. These data sets are generated with varying parameter values and demand profiles to represent a wide range of data. The one parameter we keep fixed is the rate capacity of each mobile facility, which we fix at 10. Fixing this parameter value does not restrict the scope of the data sets. Given some other data set with a different rate capacity C, the routes in the optimal solution and our heuristic solutions will not change if rate capacity of each mobile is changed to 10, and the rate demand is generated at each event point at time t is scaled by $\frac{10}{C}$. In every data set, all locations and events are positioned in a rectangular region of the plane. The travel time between each pair of locations is given by the Euclidean distance between them. A mobile facility at a location may provide service to events within a given distance of that location.

It is important to note the scope of demand profiles for the MFRP when simulating the demand in data sets. In particular, we are interested in studying the use of mobile facilities to provide service over a large geographic region with a dynamic demand profile. When demand is generated at a relatively constant rate for a large portion of the planning horizon, it is unlikely that mobile facilities could be used more efficiently than fixed facilities. In such a case, a static model may be more appropriate for determining fixed locations for the facilities. One of the greatest uses of mobile facilities is to provide service when demand levels are changing significantly over time. Thus, we do not wish to study scenarios where many event points generate demand at a relatively constant rate for a large percentage of the planning horizon. Similarly, we also do not wish to study scenarios with many locations where demand could be serviced at a relatively constant rate for the entire planning horizon. Keeping this in mind, we have created two distinct types of simulated scenarios for the MFRP.

The first type of scenario simulates, at a high level, demand profiles we might expect to occur in practice, such as in routing a fleet of portable base stations over a day. We refer to these scenarios as *"realistic" scenarios*. Each realistic scenario simulates part of a single day, beginning at 7 AM and ending at midnight. The demand profile of each event point in the scenario simulates the forecasted demand from a single event that might occur in practice, such as a sporting event, rush hour traffic, a county fair, a convention, etc. For each type of event point, we define a time window during which that type of event point may generate demand. For each event point e of that type, we choose at random a time t_{start} when demand

generation begins and a later time t_{end} when demand generation ends inside that window, although, we specify that each event point must generated demand for at least 1 hour (i.e., $t_{end} - t_{start} > 1$). During this time period, $d_e(t)$ gradually increases to a maximum level, remains at that maximum level for a period of time, and then gradually decreases to zero. The rate of increase and decrease are also chosen at random. Outside of the time period between t_{start} and t_{end}, no demand will be generated by event point e.

The demand profiles at the event points in a realistic scenarios share a specific structure. Namely, the demand is generated during a single interval of time, and during that interval demand gradually increases to, remains at, and then decreases from a maximum rate of demand generation. To see how our heuristics performs in a less structured data set, we created a second type of scenario where the demand profile of each event point displays less structure. We refer to these scenarios as *mathematically challenging scenarios*. The planning horizon of each of these scenarios begins at time 0 and ends at time 100. The demand profile for an event point is generated as follows. The number of pieces in the piecewise constant moment demand function $d_e(t)$ for an event point e was chosen at random from a specified range. To generate a wide range of demand profiles, we specified that each event point would have at least 5 and no more than 40 pieces. To find the length of each piece, we assign each piece a random number between 0 and 1. These numbers are then normalized so that their sum equals the length of the planning horizon. Each normalized number is taken to be the length of the corresponding step. The rate demand is generated by that event point (i.e., the height of $d_e(t)$) during that time is

	Sequential Routing Heuristic			
Data Set Type	**Sort 1**	**Sort 2**	**Sort 3**	**Sort 4**
R0M3L25E75	134.55	134.55	134.46	134.55
R1M5L25E75	197.39	197.57	197.22	197.52
R2M7L25E75	260.67	260.77	260.33	260.71
R3M10L25E75	316.49	316.77	316.29	315.22
R4M3L25E35	87.12	87.12	87.11	87.12
R5M5L25E35	133.25	133.34	133.19	133.04
R6M7L25E35	150.51	150.55	150.56	150.51
R7M10L25E35	169.80	169.77	169.78	169.77
R8M3L15E35	91.64	91.64	91.48	91.37
R9M5L15E35	134.19	134.21	134.11	134.17
R10M7L15E35	156.57	156.50	156.51	156.33
R11M10L15E35	170.55	170.55	170.23	170.42
R12M3L15E20	65.61	65.61	65.59	65.61
R13M5L15E20	88.10	88.10	88.10	88.00
R Average	**154.03**	**154.07**	**153.93**	**153.88**
MC0M3L25E75	2494.71	2494.96	2492.52	2492.00
MC1M5L25E75	4042.53	4040.43	4031.52	4024.26
MC2M7L25E75	5390.01	5394.17	5369.33	5374.52
MC3M10L25E75	7117.66	7122.76	7079.51	7086.65
MC4M5L10E20	2360.31	2360.58	2357.73	2357.74
MC5M10L25E50	6014.99	6019.46	5993.12	5996.95
MC6M5L15E20	2248.01	2247.33	2244.27	2243.67
MC7M10L15E20	3471.89	3469.71	3470.63	3466.91
MC8M5L10E20	2430.20	2430.29	2425.40	2426.83
MC9M5L10E15	1947.57	1947.70	1946.06	1946.75
MC Average	**3751.79**	**3752.74**	**3741.01**	**3741.63**

Tab. 3.1: Performance of the sequential routing heuristic with different sorting orders. Each row displays the averaged results from either 40 mathematically challenging data sets or 25 realistic data sets.

0 with probability p, and otherwise chosen randomly from a lognormal distribution.

In our results, the names of realistic scenarios will begin with the letter 'R' and the names of the mathematically challenging scenarios will begin with the letters 'MC', followed by a unique index for the scenario. The name then contains the letter 'M' and the number of mobile facilities, the letter 'L' and the number of locations, and finally the letter 'E' and the number of event points. For example, R0M5L25E75 refers to a realistic data set with 75 event points, 25 locations, and 5

Insertion Heuristic				
Data Set Type	**Sort 1**	**Sort 2**	**Sort 3**	**Sort 4**
R0M3L25E75	130.17	130.17	130.18	129.96
R1M5L25E75	193.70	193.65	193.44	193.56
R2M7L25E75	254.71	254.71	255.04	254.89
R3M10L25E75	311.42	311.92	310.12	310.55
R4M3L25E35	83.28	83.61	83.28	83.61
R5M5L25E35	129.45	129.57	129.22	130.11
R6M7L25E35	148.44	148.34	148.46	148.70
R7M10L25E35	169.49	169.38	169.61	169.06
R8M3L15E35	90.46	90.46	90.01	90.19
R9M5L15E35	131.88	131.81	131.40	131.80
R10M7L15E35	155.10	155.18	154.80	154.98
R11M10L15E35	170.13	170.20	169.87	170.36
R12M3L15E20	63.68	63.68	63.68	63.68
R13M5L15E20	86.44	86.49	86.69	86.92
R Average	**151.31**	**151.37**	**151.13**	**151.27**
MC0M3L25E75	2436.39	2436.55	2433.20	2427.83
MC1M5L25E75	3936.63	3934.95	3921.62	3907.47
MC2M7L25E75	5236.50	5234.01	5218.15	5217.11
MC3M10L25E75	6853.98	6855.66	6826.91	6832.50
MC4M5L10E20	2322.06	2322.04	2322.19	2319.02
MC5M10L25E50	5806.29	5804.62	5791.46	5792.32
MC6M5L15E20	2205.04	2204.13	2201.72	2198.37
MC7M10L15E20	3414.45	3415.97	3408.82	3410.38
MC8M5L10E20	2405.90	2405.41	2402.18	2397.19
MC9M5L10E15	1929.34	1933.73	1928.95	1932.74
MC Average	**3654.66**	**3654.71**	**3645.52**	**3643.49**

Tab. 3.2: Performance of the insertion heuristic with different sorting orders. Each row displays the averaged results from either 25 realistic scenarios or 40 mathematically challenging scenarios.

mobile facilities. The coverage radii for a mobile facility was (momentarily dropping the number of mobile facilities, locations, and events from the scenario names) 3.5 for data sets MC0-MC3, 4.5 for MC4-MC7 and MC9, 1 for MC8 and R0 through R7, and 1.5 for R8-R13. Computational results were compiled on a Dell Optiplex 740 with a AMD Athlon 64 X2 5000+ dual core processor with 3GB of RAM running Microsoft Windows XP. The heuristics were coded and compiled in Microsoft Visual C++ 2005.

3.4.1 The Sequential Routing Heuristic vs. the Insertion Heuristic

Tables 3.1 and 3.2 display the performance of the two route generation heuristics when different sorting orders are used with each heuristic on a variety of data sets. Each row displays the averaged results from 25 data sets generated with varying demand parameters. We make two key observations. First notice that regardless of the sorting order used in the demand assignment step, the sequential routing heuristic typically outperforms the insertion heuristic. The only data set where the insertion heuristic was consistently superior was MC7M10L15E20. Second, it appears that in both heuristics, using Sort 2 produces routes that on average service the most total demand. Consequently, we used Sort 2, which sorts events first in decreasing order of $|L_e|$ and secondly by the total demand not serviced that is generated by the event, in the demand assignment step for the remainder of our experiments.

Implementing the local search procedure with the routes generated from either the sequential routing heuristic or the insertion heuristic provided some improvement. Tables 3.3-3.6 display the demand serviced in solutions generated by the two route generation heuristics before and after the local search procedure, as well as the runtimes of the algorithms in seconds. The sequential routing heuristic typically finds higher quality solutions than the insertion heuristic, but has a longer runtime. While the insertion heuristic runs the SMFLPA more frequently than the sequential routing heuristic, the insertion heuristic found no improvement the vast majority of the time when trying to add a location to the collection of locations visited on a

route. In fact, routes generated by the insertion heuristic typically contained just one or two stops. Thus, each time the SMFLPA was run in the insertion heuristic, it was run on a substantially smaller network, providing an overall improvement in performance.

In some instances, the local search procedure found no improvement. Even after the local search procedure, the sequential routing heuristic outperforms the insertion heuristic. In addition, the runtimes of the two route generation heuristics with local search were comparable. The local search procedure also seems to provide proportionally more improvement for realistic data sets than it does for mathematically challenging data sets. This may be because the demand at each event point in a realistic data set is generated during a distinct interval of time, while the demand at an event point in a mathematically challenging data set could be generated at many different points in the planning horizon. This could produce more opportunities to find improvements by moving stops between routes for realistic data sets than between routes for mathematically challenging data sets.

3.4.2 Mobile Facilities vs. Fixed Facilities

When evaluating the quality of solutions produced by our heuristic, it is natural to ask the following question: how much more demand could be serviced by using mobile facilities than by placing an equivalent number of fixed facilities with equal rate capacity? To answer this question, we compare the demand serviced in our heuristic solutions to the demand serviced in an optimal placement of the same number of fixed facilities with equal rate capacity. We refer to the problem of

Sequential Routing Heuristic							
Data Set Type	**Sequential Heuristic** Demand Serviced	Runtime (s)	**With Local Search** Demand Serviced	Runtime (s)	**Improvement** Max.	Min.	Median
R0M3L25E75	134.55	50.18	135.30	199.24	3.41%	0.00%	0.00%
R1M5L25E75	197.57	18.85	199.11	135.60	4.43%	0.00%	0.31%
R2M7L25E75	260.77	38.03	264.37	369.58	5.15%	0.00%	1.16%
R3M10L25E75	316.77	116.88	320.88	1144.15	4.29%	0.00%	1.35%
R4M3L25E35	87.12	5.61	87.57	39.18	3.93%	0.00%	0.00%
R5M5L25E35	133.34	3.64	134.62	27.35	5.37%	0.00%	0.68%
R6M7L25E35	150.55	12.49	152.56	87.25	4.80%	0.00%	0.81%
R7M10L25E35	169.77	33.84	171.85	105.23	2.96%	0.00%	1.12%
R8M3L15E35	91.64	4.68	92.22	32.08	4.39%	0.00%	0.00%
R9M5L15E35	134.21	0.91	135.71	6.27	4.86%	0.00%	0.46%
R10M7L15E35	156.50	2.36	158.44	14.03	6.67%	0.00%	0.98%
R11M10L15E35	170.55	6.42	171.98	21.68	3.50%	0.00%	0.47%
R12M3L15E20	65.61	0.10	66.21	0.61	13.10%	0.00%	0.00%
R13M5L15E20	88.10	0.20	89.03	1.25	3.71%	0.00%	0.41%
Average	**154.07**	**21.01**	**155.70**	**155.96**	**5.04%**	**0.00%**	**0.55%**

Tab. 3.3: Performance of the sequential routing heuristic with and without local search on the realistic data sets. Each row displays the averaged results from 25 data sets. The maximum, minimum, and median improvement found by local search is given for each group of 25 data sets.

Sequential Routing Heuristic							
Data Set Type	**Sequential Heuristic** Demand Serviced	Runtime (s)	**With Local Search** Demand Serviced	Runtime (s)	**Improvement** Max.	Min.	Median
MC0M3L25E75	2494.96	142.61	2495.18	164.06	0.93%	0.00%	0.00%
MC1M5L25E75	4040.43	24.13	4046.92	27.93	1.53%	0.00%	0.00%
MC2M7L25E75	5394.17	166.73	5409.85	229.41	2.42%	0.00%	0.01%
MC3M10L25E75	7122.76	116.72	7140.36	191.80	2.18%	0.00%	0.12%
MC4M5L10E20	2360.58	3.51	2377.57	5.48	3.70%	0.00%	0.59%
MC5M10L25E50	6019.46	128.57	6042.36	304.20	2.62%	0.00%	0.21%
MC6M5L15E20	2247.33	8.48	2259.29	20.11	3.77%	0.00%	0.36%
MC7M10L15E20	3469.71	8.14	3493.89	39.36	3.13%	0.00%	0.73%
MC8M5L10E20	2430.29	6.06	2450.00	11.58	5.12%	0.00%	0.53%
MC9M5L10E15	1947.70	3.33	1962.74	9.51	5.94%	0.00%	0.39%
Average	**3752.74**	**60.83**	**3767.82**	**100.34**	**5.04%**	**0.00%**	**0.29%**

Tab. 3.4: Performance of the sequential routing heuristic with and without local search on the mathematically challenging data sets. Each row displays the averaged results from 40 data sets. The maximum, minimum, and median improvement found by local search is given for each group of 40 data sets.

Insertion Heuristic							
Data Set Type	**Insertion Heuristic** Demand Serviced	Runtime (s)	**With Local Search** Demand Serviced	Runtime (s)	**Improvement** Max.	Min.	Median
R0M3L25E75	130.17	0.17	135.08	307.34	12.56%	0.00%	3.94%
R1M5L25E75	193.65	0.29	199.11	191.78	7.24%	0.00%	1.99%
R2M7L25E75	254.71	0.31	264.20	677.30	9.62%	0.00%	3.24%
R3M10L25E75	311.92	0.30	320.82	1283.30	6.55%	0.23%	2.27%
R4M3L25E35	83.61	0.05	87.73	61.25	16.26%	0.00%	4.60%
R5M5L25E35	129.57	0.08	134.43	16.12	12.27%	0.00%	3.15%
R6M7L25E35	148.34	0.07	153.11	41.26	8.70%	0.00%	2.95%
R7M10L25E35	169.38	0.05	171.53	32.36	5.24%	0.00%	0.93%
R8M3L15E35	90.46	0.04	92.26	10.36	9.75%	0.00%	0.16%
R9M5L15E35	131.81	0.05	135.62	28.83	8.38%	0.00%	2.32%
R10M7L15E35	155.18	0.04	158.32	10.24	10.90%	0.00%	1.12%
R11M10L15E35	170.20	0.04	172.11	14.18	3.61%	0.00%	0.85%
R12M3L15E20	63.68	0.01	66.43	8.83	31.88%	0.00%	2.35%
R13M5L15E20	87.14	0.01	88.89	10.62	12.29%	0.00%	0.95%
R13M5L15E20	86.49	0.01	88.22	10.62	12.29%	0.00%	0.95%
Average	**151.37**	**0.11**	**155.64**	**185.81**	**11.09%**	**0.02%**	**2.20%**

Tab. 3.5: Performance of the insertion heuristic with and without local search on the realistic data sets. Each row displays the averaged results from 25 data sets. The maximum, minimum, and median improvement found by local search is given for each group of 25 data sets.

Insertion Heuristic							
Data Set Type	**Insertion Heuristic** Demand Serviced	Runtime (s)	**With Local Search** Demand Serviced	Runtime (s)	**Improvement** Max.	Min.	Median
MC0M3L25E75	2436.55	0.49	2461.06	47.36	8.25%	0.00%	0.00%
MC1M5L25E75	3934.95	1.06	4015.27	14.29	7.27%	0.00%	1.84%
MC2M7L25E75	5234.01	1.73	5380.02	161.76	9.72%	0.00%	2.17%
MC3M10L25E75	6855.66	2.80	7119.38	182.76	11.34%	0.00%	3.76%
MC4M5L10E20	2322.04	0.13	2370.97	8.52	7.57%	0.00%	1.66%
MC5M10L25E50	5804.62	2.00	6031.09	360.15	17.62%	0.04%	2.67%
MC6M5L15E20	2204.13	0.17	2259.30	21.31	9.61%	0.00%	1.62%
MC7M10L15E20	3415.97	0.35	3502.45	51.77	5.51%	0.00%	1.57%
MC8M5L10E20	2405.41	0.10	2457.52	9.23	6.33%	0.00%	1.18%
MC9M5L10E15	1933.73	0.10	1966.64	8.44	7.87%	0.00%	0.93%
Average	**3654.71**	**0.89**	**3756.37**	**86.55**	**9.11%**	**0.00%**	**1.74%**

Tab. 3.6: Performance of the insertion heuristic with and without local search on the mathematically challenging data sets. Each row displays the averaged results from 40 data sets. The maximum, minimum, and median improvement found by local search is given for each group of 40 data sets.

placing these fixed facilities to maximize the demand serviced as the static problem.

In many applications, static facilities are cheaper to operate than mobile facilities, which may exhibit costs for relocation that may be large enough for consideration when generating routes. This is especially true when the demand serviced can be equated with revenue. In such situations, the relocation costs of mobile facilities can be considered when generating routes by modifying the routing graph as described in Section 2.7.4 each time a heuristic calls the SMFLPA. In such cases, an operator may consider the trade offs between deploying static and mobile facilities by comparing the demand serviced in the optimal placement of static facilities with the demand serviced and the relocation costs in the mobile facility routes generated by our heuristics with such modifications.

The optimal placement of fixed facilities may be found by solving a mixed integer program (MIP). Let F be the set of fixed facilities. Given a scenario, let $S = \{0 = s_0 < s_1 < \cdots < s_K = T\}$ be the collection of all critical times of all locations in L. For each $f \in F$ and $l \in L$, define the binary variable x_{fl} to be 1 if fixed facility f is positioned at location l and 0 otherwise. For each $f \in F$, $l \in L$, $e \in E$, and $k = 0, 1, \ldots, K-1$, define the nonnegative continuous variable d_{fek} to be the amount of demand serviced by fixed facility f from event point e during $[s_k, s_{k+1})$. Let D_{ek} be the total amount of demand generated by event point e during $[s_k, s_{k+1})$. The optimal static solution may be found by solving the following mixed

integer program:

$$\text{Maximize} \quad \sum_{f \in F} \sum_{e \in E} \sum_{k=0}^{K-1} d_{fek} \tag{3.4.1}$$

$$\text{subject to:} \quad \sum_{l \in L} x_{fl} = 1 \qquad \text{for each } f \in F \tag{3.4.2}$$

$$\sum_{e \in E} d_{fek} \leq C(s_{k+1} - s_k) \qquad \text{for each } f \in F, k = 0, \ldots, K-1 \tag{3.4.3}$$

$$\sum_{f \in F} d_{fek} \leq D_{ek} \qquad \text{for each } e \in E, k = 0, \ldots, K-1 \tag{3.4.4}$$

$$d_{fek} \leq D_{ek} \sum_{l \in L_e} x_{fl} \qquad \text{for each } f \in F,\ e \in E,$$

$$\text{and for } k = 0, \ldots, K-1 \tag{3.4.5}$$

$$d_{fek} \geq 0 \qquad \text{for each } f \in F, e \in E,$$

$$\text{and for } k = 0, \ldots, K-1 \tag{3.4.6}$$

$$x_{fl} \in \{0, 1\} \qquad \text{for each } f \in F, l \in L \tag{3.4.7}$$

The objective of the mixed integer program, given by Equation (3.4.1), is to maximize the amount of demand serviced. Constraint (3.4.2) assigns each facility to a single location l. Since the rate at which every event point generates demand is constant during each interval of the form $[s_k, s_{k+1})$, Constraint (3.4.3) preserves the rate capacity of the fixed facility by dictating that the amount of demand serviced during $[s_k, s_{k+1})$ by the fixed facility cannot exceed the maximum amount of demand it could serve during that time. Constraint (3.4.4) says the amount of demand serviced by the fixed facilities during period k from event point e is less than or equal to the amount of demand generated by event point e during period k. Constraint (3.4.5) says that a facility may only service demand from event point e only if the

facility is positioned at a location $l \in L_e$. Constraints (3.4.6) specifies nonnegativity of the variables d_{fek} and (3.4.7) specifies the variables x_{fl} are binary.

Solving this mixed integer program can be computationally challenging as K can be quite large. We implemented this mixed integer program in ILOG OPL 5.2 for the data sets we generated for our heuristics. For many of the mathematically challenging data sets, this mixed integer program was computationally intractable. For the data sets where we were able to find the optimal solution to the static problem using this mixed integer program, a comparison is presented Table 3.7 of solutions to MFRP instances generated with the sequential routing heuristic with local search, and the optimal solution to the static problem. On average, using the sequential heuristic with local search to route mobile facilities allows 6.4% more demand to be serviced than the optimal use of the same number of fixed facilities for the mathematically challenging data sets. This improvement increases to 16.6% when comparing results of the realistic scenarios. In some instances, we observed improvement as high as 61.83%. In a few instances we did observe the optimal solution to the static problem outperform our heuristic solution, although typically by less than 2%.

To further evaluate the tradeoff between deploying mobile facilities or fixed facilities, we developed a third type of scenario. We begin the name of each of the third type of scenario with the letter "T". These scenarios are designed to test the performance of the heuristics relative to the percentage of time each event point is generating demand. Each of these scenarios is defined together with a parameter $\lambda \in (0, 1]$. Upon creation, an event point e in one of these scenarios is assigned

Data Set Type	Static Solution	Sequential Routing Heuristic with Local Search	Improvement Max.	Min.	Median	Instances Improved
R0M3L25E75	117.19	133.41	32.46%	4.56%	15.05%	25/25
R1M5L25E75	169.76	199.11	32.57%	7.90%	17.06%	25/25
R2M7L25E75	232.78	264.30	29.07%	1.94%	12.20%	25/25
R3M10L25E75	283.08	320.64	20.90%	4.44%	13.80%	25/25
R4M3L25E35	71.08	87.57	49.84%	7.83%	22.70%	25/25
R5M5L25E35	111.81	134.62	39.74%	9.02%	20.27%	25/25
R6M7L25E35	129.06	152.60	40.93%	6.65%	17.50%	25/25
R7M10L25E35	152.43	171.85	30.66%	2.67%	12.08%	25/25
R8M3L15E35	78.84	92.22	47.62%	0.99%	15.53%	25/25
R9M5L15E35	119.03	135.71	33.69%	0.93%	13.46%	25/25
R10M7L15E35	144.78	158.44	21.76%	1.57%	8.70%	25/25
R11M10L15E35	162.67	171.99	13.23%	1.17%	5.32%	25/25
R12M3L15E20	53.96	66.21	61.83%	1.99%	26.54%	25/25
R13M5L15E20	77.41	89.03	37.13%	2.94%	13.92%	25/25
MC0M3L25E75	2473.36	2495.69	11.32%	-4.03%	0.66%	30/40
MC1M5L25E75	3996.97	4065.89	5.94%	-2.87%	1.50%	35/40
MC4M5L10E20	2202.41	2377.22	21.14%	-9.85%	7.76%	38/40
MC6M5L15E20	2109.78	2259.30	28.01%	-13.92%	6.07%	35/40

Tab. 3.7: A comparison of the total demand serviced in the optimal static solution and in the solution to the MFRP generated with the sequential routing heuristic with local search. Each row contains the averaged results of either 40 mathematically challenging data sets, or 25 "realistic" data sets. The maximum, minimum, and median improvement for the data sets in each row is displayed, as well as the proportion of data sets on which the heuristic solution outperforms the optimal static solution.

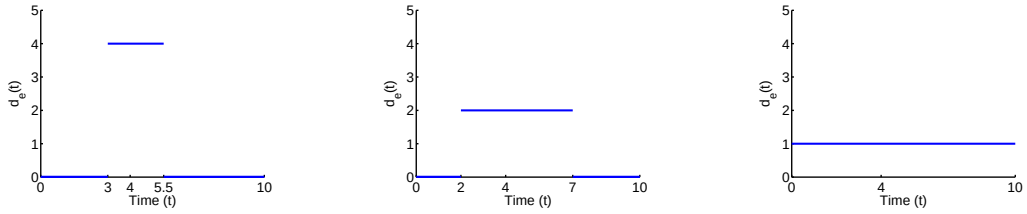
a total amount of demand it will generate during the planning horizon, D_e, and a randomly chosen time in the planning horizon, t_e. For a fixed λ, event point e will generate demand at rate $\frac{D_e}{\lambda T}$ during the interval of time $[(1-\lambda)t_e,\ t_e+\lambda(T-t_e))$ and at rate zero outside of that interval. Thus the total amount of demand generated by a particular event point e is equal for every value of λ. Figure 3.3 displays an example of the moment demand function, $d_e(t)$, for a single event point e and several values of the parameter λ. Notice that as λ approaches zero, the demand becomes more "spiky".

Each of these scenarios are generated in a 25 by 25 region of the plane with

a planning horizon of length 100. Each location is generated at a random position where at least one event point can be covered. We chose a moderately wide coverage radius of 3.5 to encourage the existence of locations that can cover multiple events. The rate capacity of each mobile facility is $\sum_{e \in E} \frac{D_e}{\lambda T}$. By definition, this rate capacity will never be exceeded for any value of λ. Furthermore, because any particular event point generates the same amount of total demand for every value of λ, this rate capacity guarantees any optimal static solution for a particular value of λ is optimal for every $\lambda \in (0, 1]$. Figure 3.4 shows four graphs comparing the demand serviced in solutions generated by the sequential heuristic with local search and in the optimal solution to the static problem for several instances of this types of scenarios as λ takes on values between zero and one. These four were chosen as being qualitatively representative of the types of behavior displayed by these data sets. These plots demonstrate the amount of demand the mobile facilities are able to service relative to the amount of demand serviced in the static problem increases as the length of time each event point generates demand decreases, and the overall demand profile becomes more "spiky". It should also be noted that as the coverage radius decreases, each facility is able to cover less nearby demand while at a location and the quality of our heuristic solution for the MFRP relative to the static solution should be expected to increase.

3.4.3 Moving Mobile Facilities to Arrive or Depart only at Critical Times

An approximation of the IDMIP formulation of the MFRP may be obtained by discretizing time. This provides an approximate optimal solution and a lower



(a) (b) (c)

Fig. 3.3: An example of the demand profile of a single event point from the third types of scenarios for three different values of λ. In all three graphs, $D_e = 10$, $T = 10$, and $t_e = 4$. Panel 3.3(a) displays the rate demand is generated at the event point for $\lambda = 0.25$. Panel 3.3(b) displays the rate demand is generated at the same event point for $\lambda = 0.50$. Panel 3.3(c) displays the rate demand is generated at the same event point for $\lambda = 1$.

bound for the MFRP. The finer the discretization, the better the approximation. On the other hand, the size of this time-discretized formulation is extremely large, and grows rapidly with the level of discretization.

For the time discretization, let $\{t_0, t_1, \ldots, t_K\}$ be the set of times when any mobile facility can be assumed to either arrive at or depart from any location. Furthermore, to discretize the IDMIP, the rate each event point generates demand must be constant during each interval of time $[t_k, t_{k+1})$. Notice the set $\bigcup_{l \in L} S_l$ satisfies this condition. We use this time discretization in our computational experiments.

To formulate this time discretization of the IDMIP as a mixed integer program, let P be the set of time periods between these times, $\{(t_k, t_{k+1}) : k = 0, \ldots, K-1\}$. For each $p \in P$ define $l_p = t_{p+1} - t_p$, the length of period p. By design, during each period p, a mobile facility will either be traveling from one location to another or be providing service from a location for the entire period. Let D_e^p be the total demand

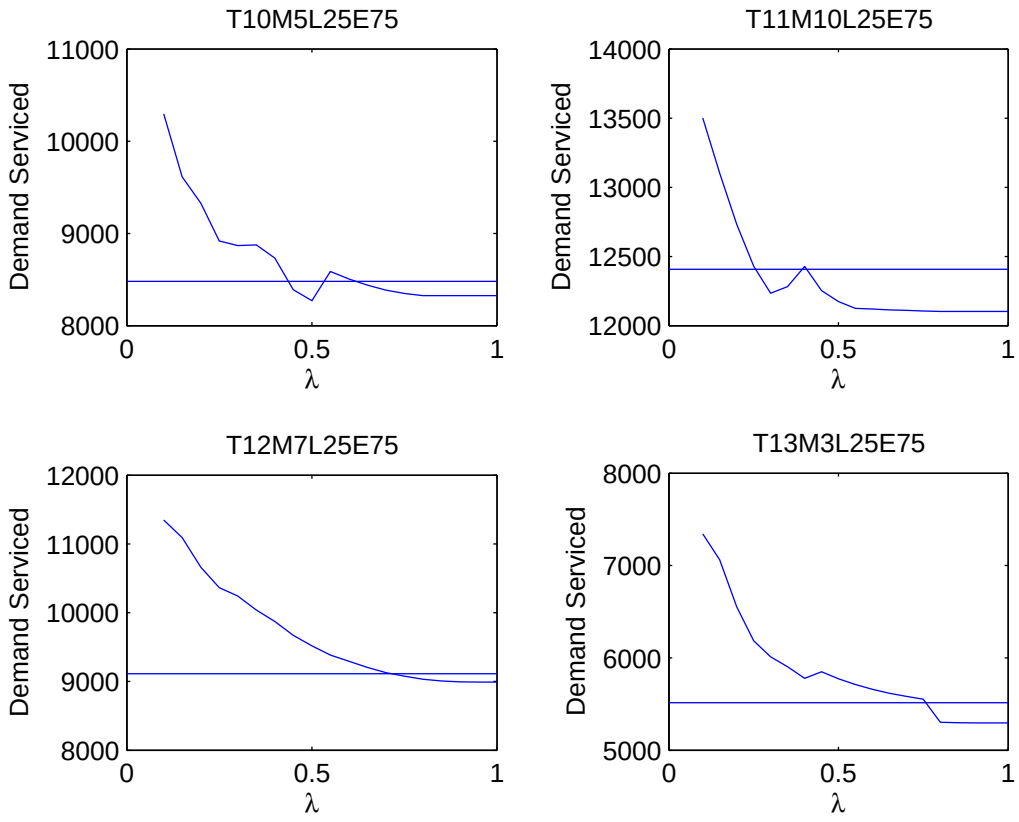


Fig. 3.4: Results from four scenarios of the third type. The horizontal line displays the amount of demand serviced in the optimal solution to the static problem, which is equal for every value of λ. The second curve displays the demand serviced in the solution to the MFRP generated by the sequential routing heuristic with local search for several scenarios as λ takes on values between 0 and 1.

generated at event point e during period p. Define the binary decision variable x_{ml}^{p} to be 1 if mobile facility m is providing service from location l in period p, and 0 otherwise. For each $e \in E$, $l \in L_e$, and $p \in P$, define the nonnegative real-valued decision variable d_{el}^{p} to be the amount of demand generated from event point e during period p serviced by all mobile facilities at location l. Then this lower bound

can be found by solving the following mixed integer program:

$$\text{Maximize} \quad \sum_{p\in P}\sum_{e\in E}\sum_{l\in L_e} d_{el}^{p} \tag{3.4.8}$$

$$\text{subject to:} \quad \sum_{l\in L_e} d_{el}^{p} \le D_e^p \qquad \text{for each } p\in P, e\in E \tag{3.4.9}$$

$$\sum_{e\in E_l} d_{el}^{p} \le (Cl_p)\sum_{m\in M} x_{ml}^{p} \qquad \text{for each } p\in P, l\in L \tag{3.4.10}$$

$$x_{ml}^{p} + x_{ml'}^{p'} \le 1 \qquad \text{for each } m\in M, p, p'\in P, l, l'\in L \text{ s.t. } (t_{p'}, t_{p'+1}) \cap (t_p - TT_{l'l}, t_{p+1} + TT_{ll'}) \neq \emptyset \tag{3.4.11}$$

$$d_{el}^{p} \ge 0 \qquad \text{for each } p\in P, e\in E, l\in L_e \tag{3.4.12}$$

$$x_{ml}^{p} \in \{0,1\} \qquad \text{for each } p\in P, m\in M, l\in L \tag{3.4.13}$$

The objective function, Equation (3.4.8) seeks to maximize the amount of demand covered. Constraint (3.4.9) ensures the amount of demand serviced during period p from event point e does not exceed the amount of demand generated by event point e during period p. Constraint (3.4.10) ensures the amount of demand serviced by mobile facilities at location l during period p does not exceed the amount of demand those mobile facilities may service during period p, which is Cl_p for each mobile facility. Since the moment demand functions are constant during each period, the rate capacity of a mobile facility won't be exceeded if this constraint is met. Constraint (3.4.11) ensures that the route of each mobile facility is feasible, respecting the travel times between locations. Constraints (3.4.12) and (3.4.13) specify the allowed values of the variables.

In general, this mixed integer program is computationally intractable for any

problem of reasonable size. Even for small problems, $|P|$ can be very large. We solved this mixed integer program using $\cup_{l \in L} S_l$ as the set of times defining the periods in P in ILOG OPL 5.2. Table 3.8 displays a comparison of the demand serviced in the routes generated with this mixed integer program and with the sequential routing heuristic with local search for small instances of the problem with two mobile facilities and four or five event points and locations. We specified in each of these scenarios that for each location l, E_l must contain the two closest event points, in addition to any event points within the coverage range from location l. In implementing this MIP, we found that problems with more than two mobile facilities, more than five event points or locations, and more than five pieces in each piecewise constant demand profile were computationally intractable. In fact, in some instances of the MFRP with five event points and five locations were difficult to solve to optimality. Because the event points in a realistic scenario only produce demand during a distinct interval of time, a realistic data set with only four or five event points could easily have no more than two events generating demand at any given time. Consequently, we present results for mathematically challenging problems generated having two mobile facilities, four or five locations, and four or five event points, which each have between 2 and 5 pieces in their piecewise constant demand profile.

Even when considering problems of such a small size, the mixed integer program can take a long time to solve. Consequently, the mixed integer program was terminated if it had not finished running after four hours and the best solution found was taken. The solution gap produced by the mixed integer program is also

displayed in Table 3.8. The solutions to the MFRP given by the sequential heuristic with local search are competitive with the solutions found by solving this mixed integer program. The heuristic solutions to these instances were computed almost instantly, while the mixed integer program sometimes couldn't be solved within four hours. In only three of the MFRP instances below with two mobile facilities, five locations, and five event points was the optimal solution to the mixed integer program found in under one hour, running in 17 minutes, 19 minutes, and 45 minutes.

The computational intractability of this MIP for problems of reasonable size, and the large amount of time it takes to solve small data sets with this mixed integer program suggest that heuristics are the best choice to determine routes for mobile facilities in the MFRP. We used our sequential routing heuristic with local search to generate routes for these problems. For each of these data sets, our heuristic executed almost instantly, returning a runtime of under 0.01 seconds. The solutions generated by our heuristics were competitive with the solutions produced by this mixed integer program, on average servicing 98.95% of the amount of demand serviced in the routes computed with this mixed integer program. We have not found any solutions to these data sets where it is locally optimal for a mobile facility at some point in its route to depart from a location and arrive at the next location at times that are not critical times. This is most probably because the data sets that could be solved by the mixed integer program were of such small size this situation did not occur. Consequently, the solutions found by the mixed integer program are most likely the optimal solutions for these small problems. Consequently, we believe it is unlikely that our heuristics could find better routes for these problems

Data Set	MIP Solution	Runtime	Solution Gap	Sequential Heuristic with Local Search	Percentage of MIP
MC10M2L4E4	1310.40	31.29	0.00%	1306.73	99.72%
MC11M2L4E4	1148.87	2.32	0.00%	1148.87	100.00%
MC12M2L4E4	711.53	6.90	0.00%	701.97	98.66%
MC13M2L4E4	1365.70	1.60	0.00%	1334.22	97.69%
MC14M2L4E4	1826.99	2.31	0.00%	1826.99	100.00%
MC15M2L4E4	1642.82	30.59	0.00%	1642.82	100.00%
MC16M2L4E4	641.21	3.07	0.00%	623.48	97.23%
MC17M2L4E4	749.18	2.57	0.00%	749.18	100.00%
MC18M2L4E4	1251.38	3.00	0.00%	1189.16	95.03%
MC19M2L4E4	515.86	3.26	0.00%	515.86	100.00%
MC20M2L5E5	1409.38	14400.00	1.54%	1391.01	98.70%
MC21M2L5E5	1318.38	2522.28	0.00%	1315.68	99.79%
MC22M2L5E5	1300.05	14400.00	0.19%	1300.05	100.00%
MC23M2L5E5	1436.13	14400.00	3.32%	1387.75	97.23%
MC24M2L5E5	1396.10	1053.80	0.00%	1389.24	99.51%
MC25M2L5E5	1211.44	2999.03	0.00%	1211.44	100.00%
MC26M2L5E5	711.00	19.79	0.00%	711.00	100.00%
MC27M2L5E5	998.24	14400.00	2.32%	998.24	100.00%
MC28M2L5E5	690.14	7483.68	0.00%	658.63	95.43%
MC29M2L5E5	1329.55	8656.25	0.00%	1329.55	100.00%

Tab. 3.8: A comparison of the demand serviced by solutions obtained with the MIP where mobile facilities must either depart from or arrive at any location at during a predetermined discrete set of times, and the demand serviced in the solutions generated by the sequential routing heuristic with local search for the MFRP. Each row displays the results from a single data set. When the MIP was solved to optimality, solution gap of 0.00% is shown.

than found by the mixed integer program. Furthermore, the data sets this mixed integer program could solve are in some sense outside the scope of the problem. Since these data sets are very small in size, the rate demand is generated by each event point varies less dramatically than in the larger data sets, which are more representative of the type of demand profiles a mobile facility would encounter in practice, and for which our heuristics are designed. Even so, the very fact that the heuristics were on average able to service 98.95% of the demand serviced by the MIP is a strong endorsement of our heuristic methods.

3.5 Conclusion

Mobile facilities are used in many different applications, ranging from cellular communication to humanitarian logistics. In this chapter, we introduced the MFRP. We showed that the optimal route for a single mobile facility may be found in polynomial time, though in general the MFRP is NP-hard. We presented several heuristics for routing mobile facilities to maximize the amount of demand they service. Our computational results on a variety of data sets have confirmed the effectiveness of our heuristics. In comparison to the optimal placement of fixed facilities, we generated routes for a fleet of mobile facilities that service more demand when demand levels are more "spiky".

The heuristics introduced in this chapter provide the operator of a fleet of mobile facilities the ability to effectively plan a set of routes to maximize the service provided by the mobile facilities. This ability can be used to efficiently plan a set of routes that will be followed periodically, such as when routing a mobile post office, or that will be followed for the immediate future, such as in the providing of relief after a disaster.

Our heuristics focus on a homogeneous fleet of mobile facilities, with predetermined configurations. However, the heuristics (and models) can be easily extended to a heterogeneous fleet of mobile facilities. For example, one may wish to select the remaining mobile facility with the highest rate capacity as the next facility to route in the sequential heuristic. In other situations, the configuration of a mobile facility may be able to be modified by adding servers to accommodate a particular demand

profile of a scenario. This is to be part of our future research on the MFRP.

4. MINIMIZING TOTAL MOVEMENT IN MOBILE FACILITY LOCATION

4.1 Introduction

The mobile facility location problem (MFLP) was proposed by Demaine et al. [20]. The MFLP is set in a graph where clients and facilities are initially located at vertices. Destination vertices must be determined for each client and facility so that each client shares its destination with at least one facility. The objective is to minimize the total weighted sum of the distances traveled by clients and facilities. Demaine et al. [20] also discuss two other objectives: minimax and minimizing the total number of clients and facilities moved.

One application of the MFLP arises in the distribution of relief supplies in response to a disaster. Suppose a relief agency has supplies stored at warehouses and must determine a distribution point for each warehouse, and from those distribution points send supplies to aid stations. The relief agency incurs a cost of shipping supplies from a warehouse to a distribution point. This is simply the per unit cost of shipping from the warehouse to the distribution point times the amount of supplies shipped. Additionally, the relief agency incurs a cost for distributing supplies from a distribution point to an aid station. Similarly, this is the per unit cost of shipping

from the distribution point to the aid station times the amount of supplies shipped. The objective is to distribute the supplies at minimum cost.

In this chapter, we show that given a set vertices that are to be the facility destinations, the MFLP can be decomposed into two polynomially solvable subproblems of determining a specific destination in this set for each client and each facility. We call these two subproblems the client assignment subproblem and the facility assignment subproblem. Using this observation, we present a novel integer programming (IP) formulation for the MFLP with significantly fewer nonzero constraint coefficients and integer variables than an earlier formulation in the literature [25]. This new formulation uses less memory, which allows us to solve large-scale instances to optimality. We introduce two new classes of effective local search heuristics for the MFLP, which we call n-OptSwap and n-SmartSwap. These local search heuristics always terminate at solutions where the client and facility assignment subproblems are solved optimally. We then introduce a new framework for the MFLP with a more general cost structure that enables the p-median and uncapacitated facility location problems to be seen as special cases of the MFLP. The IP formulations and the local search heuristics discussed in this chapter may be easily implemented in this new framework. These local search heuristics reduce to well-known local search heuristics for the p-median and uncapacitated facility location problems with bounded locality gaps. We test these local search heuristics and the new IP formulations on a variety of instances. Our results demonstrate that these local search heuristics produce high quality solutions for a large variety of test problems. In addition, our tests demonstrate that the new IP formulation allows for the solution

of large-scale instances that could not be solved previously.

The remainder of this chapter proceeds as follows. Section 4.2 formally introduces the MFLP, including previous results, the decomposition of the MFLP, and two distinct IP formulations. Two novel classes of local search heuristics, n-SmartSwap and n-OptSwap, are described in Section 4.3. A new framework for the MFLP is presented in Section 4.4. Computational experiments with these heuristics and IP formulations on a wide variety of data sets are presented in Section 4.5. A brief discussion of using Lagrangian relaxation to solve the MFLP is presented in Section 4.6. Finally, concluding remarks are in Section 4.7.

4.2 Problem Description

The MFLP is set in an edge-weighed graph $G = (V, E)$ where V denotes the set of vertices and E denotes the set of edges. There are clients at a subset $C \subseteq V$ of vertices in the graph. Each client i has a weight u_i, representing the per distance cost of satisfying the client's demand. There are also p facilities initially positioned at a subset $F \subseteq V$ of vertices. Each facility j has nonnegative weight w_j, which represents the per distance cost of relocating mobile facility j. A feasible solution to the MFLP is a selection of a destination vertex $v(j)$ for each facility j and a destination vertex $v(i)$ for each client i so that the destination of each client is also the destination of at least one facility (i.e., for each i, $v(i) = v(j)$ for some j). The (minisum) objective is to minimize the sum of the weighted distances traveled by facilities and clients. Here, the cost of moving a client or facility is proportional to

the distance traveled. When this is not the case, we will describe a generalization of the statement of the MFLP and the graph that allows us to model this.

Without loss of generality, it may assumed no two facilities are initially located at the same vertex. In such a case, for a vertex where multiple facilities begin, a copy of that vertex may be created for each of those facilities, and the distance between the copies of the vertex set equal to zero. Thus, a facility may be uniquely denoted by its vertex of origin $j \in F$. In addition, two clients beginning at the same vertex can be assumed to have the same destination vertex. If not, both clients could be reassigned to the closer of their two destinations without increasing the cost of the solution. Therefore, multiple clients beginning at the same vertex may be aggregated into a single client with weight equal to the sum of their individual weights. Thus, each client may be uniquely denoted by its vertex of origin $i \in C$. Let $d_{vv'}$ be the distance of the shortest path from vertex v to vertex v' in the graph. Then the objective of the MFLP may be written as,

$$\text{minimize} \sum_{j \in F} w_j d_{jv(j)} + \sum_{i \in C} u_i d_{iv(i)}. \tag{4.2.1}$$

4.2.1 Related Work

The MFLP was introduced by Demaine et al. [20] as one of a class of movement problems. In these problems, pebbles are located at an initial configuration at the vertices of a transportation network. The pebbles must be moved to a new target configuration to satisfy a property P while minimizing either the total movement of all pebbles (minisum), the maximum movement of some pebble (minimax), or the total number of pebbles moved. Several properties P were considered, in-

cluding connectivity, directed connectivity, path connectivity between two vertices, independence, and achieving a perfect matching. Tight approximation and inapproximability results were presented for some of these problems. Demaine et al. [20] introduced the MFLP as an extension of these movement problems, where two types of pebbles were placed on the graph, one for facilities and one for clients. When initially introduced, it was assumed the weight of each client and facility was equal to 1. It was observed that keeping facilities fixed and moving facilities to the closest client provides a 2-approximation to the minimax variant of MFLP when the weight of each client and facility equals one. The approximability of the minisum variant of the MFLP (i.e., the Minimum Total Movement MFLP) was left as an open problem.

Friggstad and Salavatipour [25] presented an 8-approximation algorithm for the minisum variant of the MFLP when travel times satisfy the triangle inequality and the facility weight w_j equals one for every facility $j \in F$. This approximation algorithm may easily be extended to the case where the facilities are homogeneous (i.e., when the weights of all facilities are equal). In such a case, the weights of the clients and facilities can be scaled so that the weight of every facility is one. This approximation algorithm produces a feasible solution by rounding the optimal solution of the linear relaxation of an IP formulation of the MFLP.

The MFLP has so far only been considered in the computer science literature, where the focus has been on approximation algorithms. Friggstad and Salvatipour [25] discuss a local search heuristic for the MFLP and demonstrate that it may have an arbitrarily large approximation ratio. To demonstrate this, Friggstad and Sal-

vatipour show how to construct an instance with a local minimum that is arbitrarily far from optimal. Our local search heuristics take advantage of the decomposability of the MFLP, which prevents our heuristics from getting stuck at the bad local minima in the Friggstad and Salavatipour instances.

Some location-relocation problems resemble the MFLP in the sense that they deal with the relocation of facilities. Gendreau et al. [28] described a system for the real-time relocation of ambulances to maintain coverage after an ambulance responds to a call. In this problem, the ambulances begin at locations and must be redeployed to satisfy coverage constraints, while maximizing the demand that is double covered minus a penalty for relocating ambulances. Demand points in this ambulance relocation problem are either double covered and add a fixed value to the objective, or are not double covered and do not contribute to the objective. Kolesar and Walker [34] considered the redeployment of fire companies in New York City while some companies are responding to a call. Again, several differences exist between the problem Kolesar and Walker considered and the MFLP. Kolesar and Walker must relocate two different types of fire companies, as some companies have ladder trucks and others do not. Kolesar and Walker seek to relocate fire companies to satisfy minimum coverage constraints. Another multi-objective model for the repositioning of response units in a probabilistic network after a change in the state of the network was presented by Sathe and Miller-Hooks in [42]. The two objectives where to maximize the demand double covered and minimize the expected relocation costs. Nair and Miller-Hooks applied this model in determining relocation policies for ambulances in [39]. The MFLP is significantly different in two regards. First

these problems deal with coverage (demand is either covered or not). Second, the objective of the MFLP explicitly includes a cost incurred from servicing a client that is dependent on the distance from its initial location to its assigned destination.

The location and relocation of servers in a stochastic network to minimize the expected travel time of clients to servers and the expected relocation costs was studied by Berman and LeBlanc [5], in which they presented a heuristic that uses the Hungarian algorithm [38] to minimize relocation costs. This model was generalized by Berman and Rahnama [6], enabling the transition of network states to be a Markovian process. The MFLP is distinct from these models. These models, as well as the stochastic models in [39] and [42], seek to determine placements of facilities for each realization of a stochastic network. Thus, a solution to these problems defines a policy specifying how to initially locate facilities given the current network state, and how to relocate facilities as the network transitions from one state to another. Since each of these problems considers the objective of minimizing the *expected* relocation costs, the facilities are not necessarily relocated at least cost after the network transitions from one specific network state to the another specific state. This separates these problems from the MFLP in two regards. Unlike these problems which seek to determine a policy that given the network state specifies where to locate facilities, the MFLP considers the initial locations of the facilities as an input to the problem. Second, the MFLP seeks to determine the least cost relocation of the facilities and clients in a static network.

4.2.2 IP Formulations

We now present an IP formulation for the MFLP described by Friggstad and Salavatipour [25]. Define a binary variable x_{iv} for each $i \in C$ and $v \in V$, and a binary variable y_{jv} for each $j \in F$ and $v \in V$. Let $x_{iv} = 1$ if the destination of client i is vertex v, and $x_{iv} = 0$ otherwise. Similarly, let $y_{jv} = 1$ if the destination of facility j is vertex v, and $y_{jv} = 0$ otherwise. The MFLP can be formulated as follows:

$$\text{(IP1) Minimize} \quad \sum_{j \in F} \sum_{v \in V} w_j d_{jv} y_{jv} + \sum_{i \in C} \sum_{v \in V} u_i d_{iv} x_{iv} \tag{4.2.2}$$

$$\text{subject to:} \quad \sum_{v \in V} x_{iv} = 1 \qquad \forall i \in C \tag{4.2.3}$$

$$\sum_{v \in V} y_{jv} = 1 \qquad \forall j \in F \tag{4.2.4}$$

$$\sum_{j \in F} y_{jv} \geq x_{iv} \qquad \forall i \in C, v \in V \tag{4.2.5}$$

$$y_{jv}, x_{iv} \in \{0, 1\} \qquad \forall i \in C, j \in F, v \in V. \tag{4.2.6}$$

The objective function (4.2.2) minimizes the total weighted movement of mobile facilities plus the total weighted movement of the clients. Constraint (4.2.3) specifies that each client i is sent to precisely one destination. Similarly, Constraint (4.2.4) sends facility j to a single destination vertex. Constraint (4.2.5) allows the destination of client i to be vertex v only if vertex v is the destination of some facility j. Constraint (4.2.6) specifies that all the variables are binary. We note that the variables x_{iv} can be relaxed to be nonnegative continuous variables provided that the variables y_{jv} remain binary.

It is easy to see that given the destination of each facility, each client should be assigned to the closest facility destination (ties may be broken arbitrarily). Since no two facilities begin at the same vertex, it may be assumed that no two facilities will have the same destination vertex in an optimal solution. If this were to occur, all but one of the facilities sharing a destination could instead be kept at their initial locations without increasing the value of the objective function (4.2.2). Thus, an optimal solution exists where no two facilities share the same initial or destination vertex. Noting this, we formulate the MFLP as follows. For each vertex $v \in V$, define the binary variable z_v. Let $z_v = 1$ if vertex v is the destination of some facility, and $z_v = 0$ otherwise. The MFLP may now be formulated as follows:

$$\text{(IP2) Minimize} \quad \sum_{j \in F} \sum_{v \in V} w_j d_{jv} y_{jv} + \sum_{i \in C} \sum_{v \in V} u_i d_{iv} x_{iv} \tag{4.2.7}$$

$$\text{subject to:} \quad \sum_{v \in V} x_{iv} = 1 \qquad \forall i \in C \tag{4.2.8}$$

$$\sum_{v \in V} y_{jv} = 1 \qquad \forall j \in F \tag{4.2.9}$$

$$\sum_{j \in F} y_{jv} - z_v = 0 \qquad \forall v \in V \tag{4.2.10}$$

$$x_{iv} - z_v \leq 0 \qquad \forall i \in C, v \in V \tag{4.2.11}$$

$$z_v, y_{jv}, x_{iv}, \in \{0, 1\}. \qquad \forall i \in C, j \in F, v \in V \tag{4.2.12}$$

In this second formulation, IP2, the objective function (4.2.7), and Constraints (4.2.8) and (4.2.9) are unchanged. If $z_v = 1$, Constraint (4.2.10) specifies that vertex v is the destination of precisely one facility. Alternatively, if $z_v = 0$, this constraint specifies that no facility may have vertex v as its destination. Constraint (4.2.11)

specifies that client i may travel to location z_v only if $z_v = 1$. Notice that if client i travels to vertex v in a feasible solution to IP2, then $z_v = 1$ by Constraint (4.2.11). Consequently, by Constraint (4.2.10) vertex v must also be a destination of some facility. Constraint (4.2.12) specifies the variables are binary. However, as we will show, the variables x_{iv} and y_{jv} may be changed to be nonnegative continuous variables so long as the variables z_v remain binary.

While IP2 has more variables and constraints than IP1, IP2 has fewer nonzero coefficients in the constraint matrix than IP1. Table 4.1 gives the exact number of variables, binary variables, constraints, and nonzero constraint coefficients for each IP. In addition, the minimum number of necessary binary variables in IP2 is less than IP1. As the computational results demonstrate in Section 4.5, this enables larger instances to be solved using IP2.

Any solution to LP relaxation of IP2 may be translated into a feasible solution to the LP relaxation of IP1 with an equal objective value by keeping the values of the variables x_{iv} and y_{jv} fixed. We only need to show Constraint (4.2.5) is not violated. In any such solution, $x_{iv} \leq z_v = \sum_{j \in F} y_{jv}$ by Constraints (4.2.10) and (4.2.11). Conversely, the values of y_{jv} and x_{iv} in a feasible solution to the LP relaxation of IP1 cannot always be used in a feasible solution to the LP relaxation of IP2. For example, suppose there are three vertices, a, b, and c, and the distance between any pair of vertices is 1. Let facilities begin at vertices a and b, and a client begin at vertex c. One feasible solution to the LP relaxation of IP1 would be to send the two facilities and the client to vertex b, which would set $y_{ab} = 1$, $y_{bb} = 1$, $x_{cb} = 1$, and all other variables to 0. The value of these variables cannot be used in a feasible

	Number of Variables	Number of Binary Variables	Number of Constraints	Number of Nonzero Entries in the Constraint Matrix
IP1	$\|V\|(\|C\|+\|F\|)$	$\|V\|\|F\|$	$\|V\|\|C\|+\|C\|+\|F\|$	$\|V\|\|C\|\|F\|+2\|V\|\|C\|+\|V\|\|F\|$
IP2	$\|V\|(\|C\|+\|F\|+1)$	$\|V\|$	$\|V\|\|C\|+\|C\|+\|F\|+\|V\|$	$3\|V\|\|C\|+2\|V\|\|F\|+\|V\|$

Tab. 4.1: A comparison of IP1 and IP2.

solution to the linear relaxation of IP2 since Constraint (4.2.10) would necessitate that $z_b = 2$ and contradict the requirement that $z_b \leq 1$ from relaxing Constraint (4.2.12).

However, the optimal objective values of the two relaxations are equal. Given any optimal solution to the LP relaxation of IP1, we will first describe how to modify the optimal solution to obtain a new optimal solution with $\sum_{j \in F} y_{jv} \leq 1$ for each $v \in V$, and then use this modified optimal solution to create an optimal solution for the LP relaxation of IP2 with an equal objective value. If $\sum_{j \in F} y_{jv} > 1$ for some v, decreasing y_{jv} and increasing y_{jj} by an amount δ that is less than the minimum of y_{jv} and $\sum_{j \in F} y_{jv} - 1$ maintains feasibility. If $w_j d_{jv} > 0$, this would decrease the value of the objective function by $w_j d_{jv} \delta$ and contradict the assumption that we started with an optimal solution to LP relaxation of IP1. Therefore, if $\sum_{j \in F} y_{jv} > 1$, then $w_j d_{jv} = 0$ for each j such that $y_{jv} > 0$. To create a new optimal solution to the LP relaxation of IP1 with $\sum_{j \in F} y_{jv} \leq 1$, for each $v \in V$ that currently has $\sum_{j \in F} y_{jv} > 1$, for each $j \in F$ decrease y_{jv} by $\frac{y_{jv}}{\sum_{j' \in F} y_{j'v}} (\sum_{j' \in F} y_{j'v} - 1)$ and increase y_{jj} by the same amount. This modification only changes the values of y_{jv} with $w_j d_{jv} = 0$. Therefore applying this modification for each $v \in V$ with $\sum_{j \in F} y_{jv} > 1$ produces a new optimal solution to the LP relaxation of IP1 with $\sum_{j \in F} y_{jv} \leq 1$ for

each $v \in V$.

To generate an optimal solution to the LP relaxation of IP2 from this modified optimal solution to LP relaxation of IP1, keep the values of x_{iv} and the modified values of y_{jv}, and define $z_v = \sum_{j \in F} y_{jv}$. By the modification above, we have that $0 \leq z_v \leq 1$ for each $v \in V$. Constraints (4.2.8) and (4.2.9) remain satisfied since the values of x_{iv} and y_{jv} are those from a feasible solution to IP1. Constraint (4.2.10) is immediately satisfied by this definition of z_v. Constraint (4.2.11) must be satisfied since by (4.2.5) and (4.2.10), $x_{iv} \leq \sum_{j \in F} y_{jv} = z_v$.

We conclude this discussion by noting that if the constraint

$$\sum_{j \in F} y_{jv} \leq 1 \qquad \forall v \in V \tag{4.2.13}$$

is added to IP1, then the LP relaxation of the two IP formulations would be equally strong. Any feasible solution to the LP relaxation of IP1 with this additional constraint can be translated to a feasible solution to the LP relaxation of IP2 with an equal objective value by setting $z_v = \sum_{j \in F} y_{jv}$.

4.2.3 Decomposing the MFLP

Suppose we are given a subset $Z \subset V$ containing p vertices, and it is specified that each mobile facility must have as its destination a distinct vertex in Z. In IP2, this is equivalent to fixing $z_v = 1$ for each $v \in Z$ and $z_v = 0$ for each $v \in V \backslash Z$. With the z_v variables now fixed, IP2 decomposes into the two disjoint subproblems of assigning each facility to a distinct vertex in Z, and assigning each client to a vertex in Z.

The *facility assignment subproblem* is to find a least cost weighted bipartite

matching between the p facilities in F and the p destinations in Z. In this case, $y_{jv} = 0$ for each $j \in F$ and $v \in V \backslash Z$. Fixing these variables at zero in IP2 reduces the size of the problem. For each $v \in Z$, we have $z_v = 1$, allowing Constraint (4.2.10) to be rewritten as $\sum_{v \in Z} y_{jv} = 1$. The facility assignment subproblem can now be formulated as the least cost weighted bipartite matching problem,

$$FA(Z) = \text{Minimize} \quad \sum_{j \in F} \sum_{v \in Z} w_j d_{jv} y_{jv} \tag{4.2.14}$$

$$\text{subject to:} \quad \sum_{v \in Z} y_{jv} = 1 \qquad \text{for each } j \in F \tag{4.2.15}$$

$$\sum_{j \in F} y_{jv} = 1 \qquad \text{for each } v \in Z \tag{4.2.16}$$

$$y_{jv} \geq 0 \qquad \text{for each } j \in F,\ v \in Z. \tag{4.2.17}$$

Since the constraint matrix is totally unimodular, the integrality of the y_{jv} variables has been relaxed. Thus for a fixed Z, the solution to the facility assignment subproblem may be computed in polynomial time using one of may algorithms, such as the Hungarian algorithm. [38]

The *client assignment subproblem* is to choose a destination in Z for each client that minimizes the weighted distance traveled by clients. Constraint (4.2.11) specifies that a client may travel to a vertex v only if $z_v = 1$. By setting each variable $x_{iv} = 0$ for $v \in V \backslash Z$, Constraint (4.2.11) may be set aside for each $v \in V \backslash Z$. Thus, Constraint (4.2.11) may be rewritten as $x_{iv} \leq 1$ for each $i \in C$, $v \in Z$. This constraint is redundant since x_{iv} is a binary variable and may also be thrown out.

Thus, the client assignment subproblem may be rewritten as,

$$\begin{aligned} CA(Z) = \text{Minimize} \quad & \sum_{i \in C} \sum_{v \in Z} u_i d_{iv} x_{iv} & & \\ \text{subject to:} \quad & \sum_{v \in Z} x_{iv} = 1 & \text{for each } i \in C & \qquad (4.2.18) \\ & x_{iv} \geq 0 & \text{for each } i \in C,\ v \in Z. & \end{aligned}$$

Again, since the constraint matrix is totally unimodular, the integrality of the x_{iv} variables has been relaxed. Since no constraints link two different clients, this can be further decomposed into a separate subproblem for each client, which can be solved by sending the client to the closest location in Z. This produces a formulaic explanation why the client assignment subproblem may be solved by sending each client $i \in C$ to the closest vertex in Z.

Consequently, each subset of $Z \subset V$ of size p can be identified with a unique solution where both subproblems are solved optimally. Noting this, we focus on developing local search heuristics that seek to find a subset Z minimizing the sum $FA(Z) + CA(Z)$.

4.3 Local Search for the MFLP

Friggstad and Salavatipour [25] described a collection of local search operations that define a solution neighborhood. It was shown a local search heuristic allowing only these operations (which we now describe) can produce an arbitrarily large locality gap. For a fixed number n, the neighborhood of solutions is defined by the following two types of operations:

1. Select a subset of $k \leq n$ facilities, $j_1, \ldots, j_k$, and a subset of k unoccupied

destination vertices for them, $v_1, \ldots, v_k$, respectively; change the destination of facility j_l to be v_l for each $l = 1, \ldots, k$.

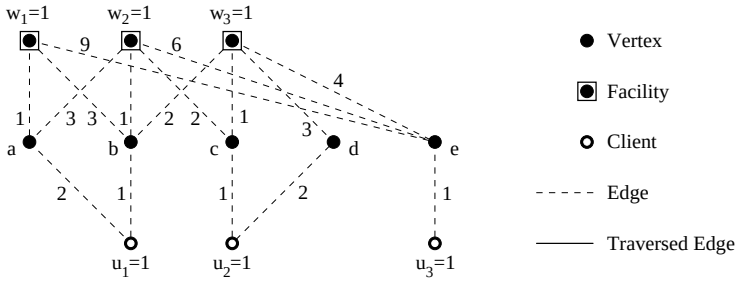
2. Select a subset of $k \leq n$ facilities, $j_1, \ldots, j_k$, and choose a permutation $\pi : \{1, \ldots, k\} \to \{1, \ldots, k\}$; change the destination of facility j_l to $v(j_{\pi(l)})$ for each $l = 1, \ldots, k$.

These two operations produce a large neighborhood of solutions. The first operation described above generates $\sum_{k=1}^{n} \binom{|F|}{k} \binom{|V|-|F|}{k} k!$ solutions in the neighborhood and the second operation generates an additional $\sum_{k=1}^{n} \binom{|F|}{k} k!$ solutions in the neighborhood.

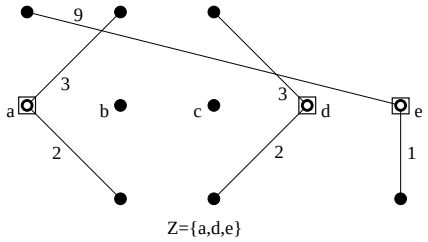
One example of a local search heuristic using only these operations is to start with the solution where each facility remains fixed at its initial location. Then repeatedly select the least cost solution in local search neighborhood if it improves the current solution (where the cost of a solution is calculate by the cost of moving each facility to its destination plus the cost of sending each client to the closest facility). We name this n-Swap local search. Figure 4.1 gives an example of how solutions in this local search neighborhood are generated.

Friggstad and Salavatipour demonstrate how to construct an MFLP instance where a local search heuristic allowing only the above operations may exhibit an arbitrarily large locality gap. To construct this example, choose a large positive integer F.[1] The graph will contain $F + n + 1$ vertices in a cycle that are numbered

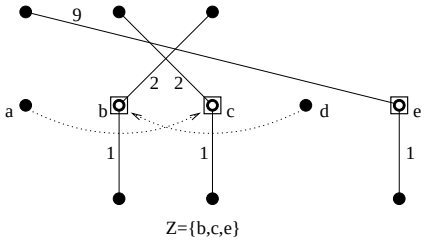
[1] The local search in Friggstad and Salavatipour [25] considers any improving solution in the neighborhood. n-Swap (choosing the best solution in the neighborhood) will not terminate in the local minimum with cost F in these examples. However, we can construct examples where n-Swap terminate at a local minimum that is arbitrarily far from optimal (even when the best neighbor is selected).



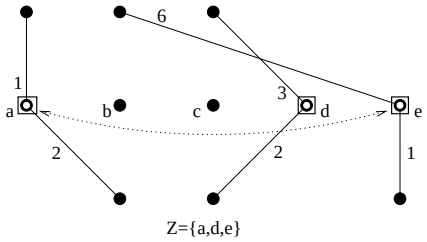
(a) Initial Configuration



(b) Initial Solution



(c) Example of the first type of operation.



(d) Example of the second type of operation.

Fig. 4.1: An example of 2-Swap local search. Panel 4.1(a) gives the configuration of the graph. Clients and facilities begin, respectively, at the vertices labeled with their weights u_i and w_j. Panel 4.1(b) gives an initial feasible solution with cost 20. Here, Facility 1 and Client 1 travel to Vertex a, Facility 2 and Client 2 travel to Vertex d, and Facility 3 and Client 3 travel to Vertex e. Panel 4.1(c) shows an example of a feasible solution in the neighborhood explored by 2-Swap local search generated by the first type of operation. Here, Facility 2 which had Vertex a as its destination is instead moved to Vertex c, and Facility 3 which had Vertex d as its destination is instead moved to Vertex b. The cost of this solution is 16. Panel 4.1(d) shows another solution in the neighborhood explored by 2-Swap resulting from the second type of operation where the destinations of Facilities 1 and 2 are permuted. The cost of this solution is 15. 2-Swap local search explores the neighborhood defined by all such operations and selects the solution in the neighborhood with the lowest cost.

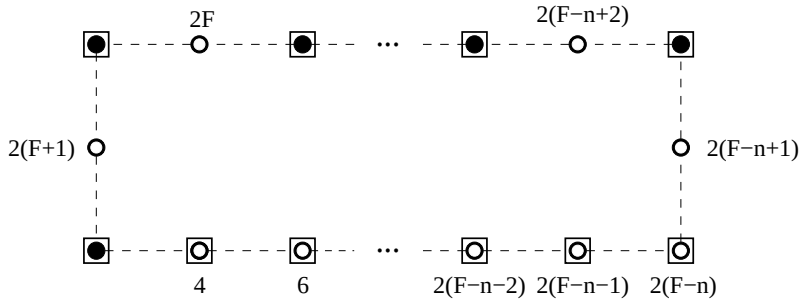


Fig. 4.2: An example of an MFLP instance with an arbitrarily large locality gap. Weights are shown for each client. Each facility has weight 1 and each edge has length 1.

in counter-clockwise order starting with 1 and ending with $F + n + 1$. Each edge is given length 1. A facility and client with demand $2k$ is placed at each vertex $k = 2, \ldots, F - n$. The remaining vertices alternate between having a client and a facility, beginning with a client on vertex $F - n + 1$. The demand of each of these clients is two more than the demand of the previous clients. Figure 4.2 depicts this instance.

The solution produced by keeping all clients fixed and moving each facility one vertex counter-clockwise produces a local minimum with cost F. The solution generated by keeping all clients fixed moving each facility to the closest client in the clockwise direction produces a solution of cost $n + 1$. This yields a locality gap of $F/(n+1)$. Thus, an instance with an arbitrarily large locality gap may be produced by choosing a sufficiently large value F.

In this example, the clients do not move in either solution and each facility travels to a distinct destination with a client. In fact, the set of facility destinations (i.e., Z) in both solutions are equal. The facility assignment subproblem is solved optimally in only one of these two solutions, namely the lower-cost solution where each facility moves to the nearest client in the clockwise direction.

4.3.1 n-OptSwap Local Search

Each set of facility destinations Z can be identified with a unique solution that can be computed in polynomial time by solving the facility and client assignment subproblems. This is a least cost solution where facilities occupy the destinations in Z. This suggests how to define an improved local search neighborhood. To do so, associate with each subset of facility locations Z the solution where the facility and client assignment subproblems are solved optimally. Then rather than searching for the changes to the individual destinations of each facility independently, define a local search neighborhood by a collection of changes to the set of facility destinations Z.

We have developed a local search heuristic that explores such a neighborhood, which we call n-OptSwap local search. In n-OptSwap, each subset of p vertices Z is associated with the solution found by solving optimally the facility assignment subproblem and the client assignment subproblem. Given a current set of facility destinations Z, the solutions in the neighborhood searched by n-OptSwap are generated by replacing of each subset of n facility destinations in Z with each subset of n destinations in $V \backslash Z$, and then optimally solving the facility assignment subproblem and the client assignment subproblem. The solution found providing the best improvement is selected as the new solution. This process is iterated until no further improvements are found. Notice the second type of exchange operation referenced above would never improve a solution generated by n-OptSwap since n-OptSwap optimally solves the facility assignment subproblem. The steps for n-OptSwap are

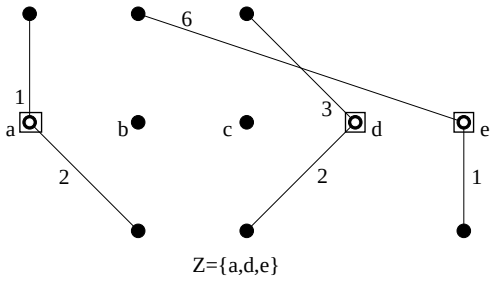
as follows:

Step 1: Define the variables $\overline{Z}$ and Z'. Initialize the variables $\texttt{curObj} := FA(Z) + \texttt{curCAcost}$ and $\texttt{curImp} := 0$.

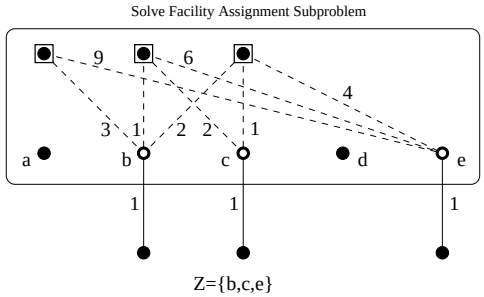
Step 2: For each subset $\{v_1, \ldots, v_n\} \in V \backslash Z$ of distinct vertices and each subset $\{z_1, \ldots z_n\} \subset Z$ of distinct vertices do the following: Set $Z' := Z \cup \{v_1, \ldots, v_n\} \backslash \{z_1, \ldots, z_n\}$. If $CA(Z') > \texttt{curObj}$, no improvement will be found so move onto the next pair of subsets. Otherwise, compute $FA(Z')$. If $\texttt{curObj} - FA(Z') - CA(Z') > \texttt{curImp}$, then set $\overline{Z} := Z'$ and set $\texttt{curImp} := \texttt{curObj} - FA(Z') - CA(Z')$.

Step 3: If $\texttt{curImp} = 0$, terminate. Otherwise, set $Z := \overline{Z}$, set $\texttt{curObj} := \texttt{curObj} - \texttt{curImp}$, and update the client and facility assignments. Return to Step 2.

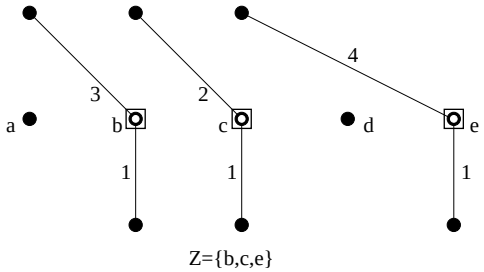
Figure 4.3 gives an example of how 2-OptSwap computes solutions in the local search neighborhood. n-OptSwap has a considerable smaller neighborhood of solutions to explore during each iteration than n-Swap. Specifically, $\binom{|F|}{n}\binom{|V|-|F|}{n}$ solutions are in the neighborhood of a current solution during n-OptSwap. However, more time must be taken to compute each solution since the facility assignment subproblem is solved for each feasible solution in this neighborhood. Thus, we now describe a method for quickly solving the facility assignment subproblem in Step 2 of n-OptSwap.



(a) Initial Solution



(b) Facility Assignment Subproblem to Solve



(c) Solution in Search Neighborhood

Fig. 4.3: An example of how solutions in the neighborhood explored by 2-OptSwap are generated. Panel 4.3(a) gives an initial solution where $Z = \{a, d, e\}$ with cost 15. Unlike 2-Swap local search, 2-OptSwap would only see the solution in Figure 4.1(b) if it is initialized with that solution, since the facility assignment subproblem is not solved optimally in that solution. Panel 4.3(b) shows the facility assignment subproblem that is solved by 2-OptSwap when a and d in Z are replaced with b and c. Panel 4.3(c) displays the solution found after solving the facility assignment subproblem. The cost of this solution is 12. 2-OptSwap explores all solutions in the neighborhood and selects the one with the least cost.

Quickly Resolving the Facility Assignment Subproblem

To decrease runtime during execution of Step 2, we do not resolve the facility assignment subproblem $FA(Z')$ from scratch. Rather, we use a modified version the Hungarian algorithm to reduce runtime. Recall that given an edge-weighted bipartite graph $G = (X \cup Y, E)$, the Hungarian algorithm takes as input a feasible labeling of the vertices, $\{l_x : x \in X\} \cup \{l_y : y \in Y\}$, and outputs a maximum weight perfect matching between X and Y. Munkres [38] observed that the Hungarian

algorithm is strongly polynomial. The worst case runtime is $O(|X|^4)$. Edmonds and Karp [22] and Tomizawa [44] described modifications that produce a $O(|X|^3)$ running time, which we adopt. Gabow and Tarjan [26] devised an algorithm for solving the maximum weight bipartite matching problem that has polynomial time complexity of $O(\sqrt{|X|}|E|\log(D|X|))$ where D is the distance matrix with entries $w_j d_{jz}$ for $j \in F$ and $z \in Z'$.

The Hungarian algorithm may be easily modified to quickly find a least cost perfect matching in the facility assignment subproblem. The Hungarian algorithm is initialized with an initial feasible labeling $L = \{l_x : x \in X\} \cup \{l_y : y \in Y\}$ and an initial matching M. In the context of the facility assignment subproblem given by (4.2.14)-(4.2.17), a labeling is feasible if $l_x + l_y \leq d_{xy}$ for all $(x, y) \in E$. Let $G_L = (V, E_L)$ be the graph formed by the tight edges of G with labeling L (i.e., E_L contains those edges (x, y) where $l_x + l_y = d_{xy}$.) For a subset of vertices $V' \subset X \cup Y$, define the neighborhood of V' as $J_{G_L}(V') = \{u : (v, u) \in E_l\}$. The steps of the $O(|X|^3)$ algorithm for finding the least cost perfect matching are as follows:

Step 1: If M is a perfect matching, then stop. Otherwise find an unmatched vertex $x \in X$ and set $S := \{x\}$ and $T := \emptyset$. For each $y \in Y$, define $\texttt{slack}_y := \max_{x \in S}\{l_x + l_y - d_{xy}\}$.

Step 2: If $J_{G_L}(S) \neq T$, goto Step 3. Otherwise, set $\delta := \max_{y \in Y \setminus T} \texttt{slack}_y$. Then

update the labels as follows:

$$l_v := \begin{cases} l_v - \delta, & \text{if } v \in S; \\ l_v + \delta, & \text{if } v \in T; \\ l_v & \text{otherwise.} \end{cases}$$

Update the graph G_L to include the additional tight edges under the new labeling. For each $y \in Y \backslash T$, set $\texttt{slack}_y := \texttt{slack}_y - \delta$. Goto Step 3.

Step 3: Choose a $y \in J_{G_L}(S)$. If y is matched by M with a vertex x', set $S := S \cup \{x'\}$ and $T := T \cup \{y\}$. For each $y \in Y$, update $\texttt{slack}_y := \max\{\texttt{slack}_y, l_{x'} + l_y - d_{x'y}\}$. Goto Step 2. Otherwise, an path P exists from x to y that alternates between edges in M and edges in $E_l \backslash M$. Increase the size of the matching M by setting $M := (M \backslash P) \cup (P \backslash M)$. Return to Step 1.

When solving the facility assignment subproblem, we create a node labeled x_j for each facility $j \in F$ and a node labeled y_v for each vertex $v \in Z$. An edge is added between node x_j and node y_v with length d_{jv}. Assume we have constructed this graph for a given Z, we have solved the least cost perfect matching problem on this graph, and we have the corresponding labels l_{x_j} and l_{y_j}. Now, suppose we are computing $FA(Z')$ during Step 2. Changing $z_k \in Z$ to $v_k \in Z'$ is equivalent to changing the length on each edge (x_j, y_{z_k}) from d_{jz_k} to d_{jv_k} and relabeling vertex z_k as v_k. As a result, if we define the label $l_{y_{v_k}}$ by $l_{y_{v_k}} = l_{y_{z_k}} + \min_{j \in F}\{d_{jv_k} - l_{x_j} - l_{y_{z_k}}\}$ for $k = 1, \ldots, n$, then we create feasible labels for the least cost perfect matching problem between F and Z'. After updating each label $l_{y_{v_k}}$ in this manner, we look to see if we can increase each label l_{x_j} by setting $l_{x_j} = l_{x_j} + \min_{v \in Z'}\{d_{jv} - l_{x_j} - l_{y_v}\}$.

During execution of Step 2 of n-OptSwap, an improved solution to the MFLP is found if $\texttt{curObj} - FA(Z') - CA(Z') > \texttt{curImp}$. Therefore, an improved solution is found in Step 2 if $FA(Z) + CA(Z) - \texttt{curImp} > FA(Z') + CA(Z')$. Since these new vertex labels for the updated least cost perfect matching problem are a dual feasible solution to the facility assignment subproblem, it follows that $FA(Z') \geq \sum_{j \in F} l_{x_j} + \sum_{v \in Z'} l_{y_v}$. Therefore, if at some stage of calculating $FA(Z')$ we find that $\sum_{j \in F} l_{x_j} + \sum_{v \in Z'} l_{y_v} \geq FA(Z) + CA(Z) - \texttt{curImp} - CA(Z')$, then $FA(Z') \geq FA(Z) + CA(Z) - \texttt{curImp} - CA(Z')$. Moving $CA(Z')$ to the left hand side, we have that $FA(Z') + CA(Z') \geq FA(Z) + CA(Z) - \texttt{curImp}$. In other words, replacing Z with Z' will not improve the least cost solution observed.

n-OptSwap initializes the Hungarian algorithm with these vertex labels and also seeds the maximum cardinality matching with whatever edges from the matching between F and Z' remain tight with respect to these new labels. After each time the Hungarian algorithm updates the labels, n-OptSwap checks to ensure $\sum_{j \in F} l_{x_j} + \sum_{v \in Z'} l_{y_v} < FA(Z) + CA(Z) - \texttt{curImp} - CA(Z')$. If this inequality does not hold, Z' is not an improvement over the least cost solution observed. If the inequality does hold true, then the Hungarian algorithm is allowed to continue searching for augmenting paths to increase the size of the matching.

As described above, each iteration of the Hungarian algorithm has runtime $O(p^2)$. If a least cost perfect matching was computed from scratch to obtain $FA(Z')$ using the Hungarian Algorithm, the runtime would be $O(p^3)$. However, updating the solution to the Hungarian algorithm as described above, reduces the runtime to $O(np^2)$ since at least $p - n$ edges from the matching between Y and Z will remain

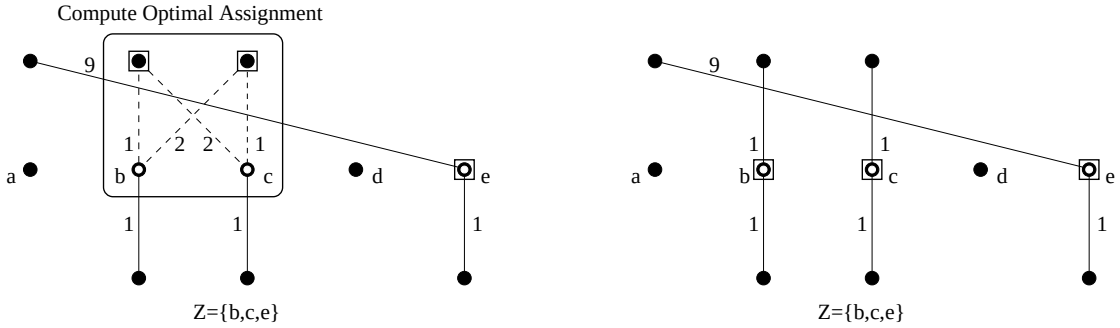
tight under the new labels. Since n typically should be small relative to p, this method for computing $FA(Z')$ can substantially reduce runtime.

4.3.2 n-SmartSwap Local Search

The computational results presented in Section 4.5 will show that while n-OptSwap produces high quality solutions, the runtime of n-OptSwap, even with the improved method for resolving the facility assignment subproblem in Step 2, can be undesirably long. Consequently, we devised a hybrid local search called n-SmartSwap, which combines the speed of n-Swap local search with the capability of n-OptSwap to resolve the facility assignment subproblem optimally. Again, let $Z \subset V$ be the set of facility destinations for a current feasible solution. The n-SmartSwap heuristic searches a neighborhood of solutions defined by all swaps of the destinations of n facilities in Z with n distinct vertices in $V \backslash Z$. All such swaps are considered. If an improvement is found, the swap that produces the best improvement is implemented. If no improvement can be found, the facility assignment subproblem is resolved optimally to see if a new assignment of the facilities will produce a lower cost solution. The steps in the algorithm are as follows:

Step 1: Define the variables $\overline{v}_1, \ldots \overline{v}_n$ and $\overline{j}_1, \ldots, \overline{j}_n$. Next, set `curCAcost` := $CA(Z)$ and set `curObj` := $FA(Z)$ + `curCAcost`. Set `curImp` := 0.

Step 2: For each subset $\{v_1, \ldots, v_n\} \subset V \backslash Z$ and subset $\{j_1, \ldots, j_n\} \subset F$ of distinct facilities, do the following. Set $Z' := Z \cup \{v_1, \ldots, v_n\} \backslash \{v(j_1), \ldots, v(j_n)\}$. If $CA(Z') >=$ `curObj`, continue to the next pair of subsets of facilities and locations. Otherwise, let `matchCost` be the cost of the least cost perfect matching



(a) Matching Problem to Solve (b) New Solution

Fig. 4.4: An example of how 2-SmartSwap generates solutions in its local search neighborhood during Step 2. Starting with the initial solution from Figure 4.1(b) where $Z = \{a, d, e\}$, the pictures here display how the solution in the neighborhood is computed by replacing a and d in Z with b and c. Panel 4.4(a) displays the matching problem to be solved during Step 2 of 2-SmartSwap between facilities 2 and 3 and vertices b and c. Panel 4.4(b) displays the solution found by computing this assignment. The particular member of the search neighborhood found has cost 14. 2-SmartSwap explores all solutions in the neighborhood and selects the one with the least cost.

between $\{v_1, \ldots, v_n\}$ and $\{j_1, \ldots, j_n\}$. If $\texttt{curCAcost} + \sum_{k=1}^{n} d_{j_k, v(j_k)} - CA(Z') - \texttt{matchCost} > \texttt{curImp}$, set $\texttt{curImp} := \texttt{curCAcost} + \sum_{k=1}^{n} d_{j_k, v(j_k)} - CA(Z') - \texttt{matchCost}$. Then set $\overline{j}_k := j_k$ and $\overline{v}_k$ to the vertex in $\{v_1, \ldots v_n\}$ that facility j_k was matched with.

Step 3: If $\texttt{curImp} = 0$, goto Step 4. Otherwise implement the improvement found. Set $Z := Z \cup \{\overline{v}_1, \ldots, \overline{v}_n\} \backslash \{v(\overline{j}_1), \ldots, v(\overline{j}_n)\}$. Next set $\texttt{curCAcost} := CA(Z)$ and set $\texttt{curObj} := \texttt{curObj} - \texttt{curImp}$. Set $v(\overline{j}_k) := \overline{v}_k$. Reset $\texttt{curImp} := 0$. Goto Step 2.

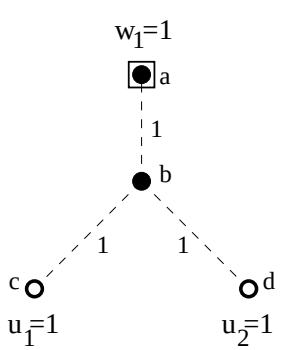
Step 4: Resolve $FA(Z)$. If $\texttt{curObj} - \texttt{curCAcost} - FA(Z) \leq 0$, then terminate. Otherwise, set $\texttt{curObj} := \texttt{curCAcost} + FA(Z)$ and update the facility and client assignments. Goto Step 2.

Figure 4.4 gives an example of how 2-SmartSwap computes solutions in its local search neighborhood during Step 2. There are $\binom{|F|}{n}\binom{|V|-|F|}{n}$ solutions in the neighborhood explored by n-SmartSwap, which is the same size as the neighborhood explored n-OptSwap. However, the time n-SmartSwap takes to compute each solution is significantly less than n-OptSwap when n is small relative to $|F|$. Using the Hungarian algorithm, the computation of each solution in the search neighborhood of n-SmartSwap has runtime $O(n^3)$ while the computation of each solution by n-OptSwap has runtime $O(n|F|^2)$.

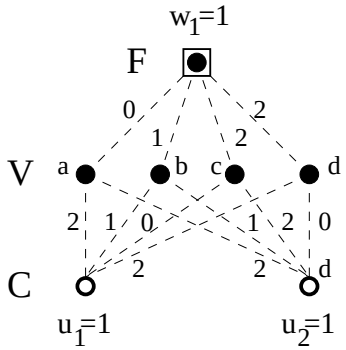
At the end of execution, the solution generated by n-SmartSwap will always have the facility assignment subproblem and the client assignment subproblem solved optimally. Consequently, n-SmartSwap may be able to get out of a local minimum where n-Swap may stuck.

4.4 A New Framework for the MFLP that generalizes the p-Median and Uncapacitated Facility Location Problems

The MFLP was originally formulated as taking place in a graph $G(V, E)$ where facilities are initially located at a subset of vertices $F \subset V$ and clients are initially located at a subset of vertices $C \subset V$. However, the decomposition of the MFLP into the polynomially solvable client and facility assignment subproblems for a given set of facility destinations hints at a new framework for the MFLP that generalizes both the p-median and uncapacitated facility location problems [19]. In addition, this new framework permits the p-center problem [19] to be seen as a special case of the MFLP with a minimax objective. As the MFLP in this new framework is



(a) MFLP in the original framework, set on a graph $G(V, E)$ with $F \subset V$ and $C \subset V$.



(b) MFLP in the new framework, set on a graph $G(F \cup V \cup C, E)$ where F, V, and C are disjoint sets of vertices.

Fig. 4.5: An example of transforming the MFLP into the more general framework. Panel 4.5(a) shows an instance of the MFLP on a graph $G(V, E)$, while Panel 4.5(b) shows the same instance in the new framework. In this new graph, F is the lone vertex in the top row, V contains vertices a, b, c, and d in the second row, and C contains the two vertices in the bottom row.

a generalization of these important and well-studied problems in facility location, algorithms and results for the MFLP can be applied to these other problems.

Our new framework for the MFLP is created by separating the initial locations of the facilities F and clients C from the vertices V, so that the MFLP is set in a graph $G(F \cup V \cup C, E)$ where F, V, and C are disjoint sets of vertices. More specifically, given an instance of the MFLP, create a new vertex for each $j \in F$ and create an edge between each $j \in F$ and each $v \in V$ with length d_{jv}. In addition, create a new vertex for each $i \in C$ and an edge between each $i \in C$ and each $v \in V$ with length d_{iv}. This creates a graph with three disjoint levels of vertices: F, V, and C. Figure 4.5 gives an example of the transformation of an MFLP instance from the original framework to our new framework. In this new framework, a feasible solution to the MFLP is a selection of a destination vertex $v(j) \in V$ for each facility $j \in F$ and a destination vertex $v(i) \in V$ for client $i \in C$ so that the destination of

each client is the destination of at least one facility.

This new framework permits a more general cost structure, where the relocation costs of clients and facilities over an edge are not proportional to each other. As a result, the p-median and uncapacitated facility location problems can now be modeled as special cases of the MFLP.

The p-median is set on a graph $G(\overline{V}, \overline{E})$ with clients located at vertices in $\overline{V}$. The objective is to place p or less facilities and assign each client to a facility to minimize the cost of serving all clients. To realize this as a special case of the MFLP, create p vertices in F that are each the initial location of one facility. Set $V = \overline{V}$, and set $d_{jv} = 0$ for each $j \in F$ and $v \in V$. Create a vertex in C for each client i in the p-median problem, and set d_{iv} equal to the cost in the p-median problem of serving client i from a facility at vertex v. Set the weight of each facility and client equal to one. An optimal solution determines a placement of p facilities in V that minimizes the cost of servicing all clients. The p-Center Problem may be realized as a special case of the MFLP in the same manner, except with a minimax objective.

The uncapacitated facility location problem is set in a graph $G(\overline{V}, \overline{E})$ with clients located at vertices in $\overline{V}$. There is a cost h_v for opening a facility at each vertex $v \in \overline{V}$. The problem is to determine which facilities to open and determine an assignment of clients to open facilities that minimizes the cost of opening facilities and serving clients. To realize this as a special case of the MFLP, create $|\overline{V}|$ vertices in F, each with a facility with weight one. Create a vertex $i \in C$ with weight one for each client i in the uncapacitated facility location problem. V will consist of two disjoint sets of vertices, V_1 and V_2. For each $v \in \overline{V}$, create a vertex $v \in V_1$.

Assign the length of the edge from each $j \in F$ to each $v \in V_1$ by $d_{iv} = h_v$, and set the length of the edge from each $i \in C$ to each $v \in V_1$ as the cost of assigning client i to a facility at vertex v in the uncapacitated facility location problem. Next, create $|\overline{V}| - 1$ new vertices in V_2. For each $v \in V_2$, set $d_{jv} = 0$ for all $j \in F$ and $d_{iv} = M$ for each $i \in C$, where M is a number large enough to prevent a client from ever having a destination in V_2 in an optimal solution. (For example, set $M = \sum_{i \in C} \sum_{v \in V} u_i d_{iv} + \sum_{j \in F} \sum_{v \in V} w_j d_{jv}$.)

If a facility moves from $j \in F$ to $v \in V_1$ in a feasible solution to the MFLP, the cost incurred is the same as the cost of opening a facility at vertex v in the uncapacitated facility location problem. For each vertex without an open facility in a solution to the uncapacitated facility location problem, there will be a facility $j \in F$ in the MFLP that cannot have a destination $v \in V_1$. These facilities may be given a destination in V_2 without incurring an additional cost in the solution to the MFLP.

In our new framework, n-Swap, n-OptSwap, and n-SmartSwap reduce to the well-known local search heuristics for the p-median and uncapacitated facility location problems presented in [2], where it was shown that when $n = 1$, these local search heuristics provide a 5-approximation algorithm for the p-median problem and a 3-approximation algorithm for the uncapacitated facility location problem. While it was previously proven that the neighborhood explored by n-Swap has a local minimum that is arbitrarily far away from optimal, we conjecture that by terminating at a solution with the facility assignment subproblem solved optimally, n-OptSwap and n-SmartSwap produce constant-factor approximations to the MFLP.

4.5 Computational Results

This section presents results from our extensive computational experiments with the IP formulations and the heuristics presented in this chapter. First, we discuss the computational tractability of IP1 and IP2. These computational experiments demonstrate IP1 and IP2 may often be solved reasonably quickly; however, instances were observed where one or both IP formulations could not be solved, or the time taken to solve either IP was considerable. Second, we present results from our implementations of the local search heuristics discussed in this chapter. These results demonstrate that while, these heuristics are all effective at finding high quality solutions to the MFLP, optimally resolving the facility location subproblem significantly lowers the optimality gap of n-SmartSwap and n-OptSwap. In most cases, the runtime of 1-SmartSwap is shorter than the time taken by CPLEX to solve either IP.

All computational results were implemented on a Dell Optiplex 740 with a AMD Athlon 64 X2 5000+ dual core processor and 3GB of RAM running Microsoft Windows XP. Our implementations were coded in C++ and compiled using Microsoft Visual Studio 2005. All IP formulations were solved using ILOG CPLEX 11.0.

Our MFLP instances were adapted from the 40 uncapacitated p-median instances of Beasley [31], and were obtained from the OR-Library [4]. These MFLP instances may be view as set in the original framework for the MFLP. (i.e., F and C are subsets of V.) The p-median data files are named "pmed1.txt", "pmed2.txt",

..., "pmed40.txt". Each p-median instance is defined on a graph. The data file for an instance provides the number of vertices in the network, the lengths of a number of edges in the network, and the number of facilities to be placed p. The length of any edge not listed is defined to be infinity. Floyd's algorithm was used to determine the shortest path between all pairs of vertices of the network. The MFLP instance has p facilities. The initial location of each facility is a randomly chosen vertex. For each data set, a number ρ chosen from a uniform distribution between 0.2 and 0.8. With probability ρ, a client is given a weight chosen from a uniform distribution between 1 and 10, and assigned weight 1 with probability $1 - \rho$. Table 4.2 gives the number of vertices, facilities, and clients in each of our MFLP data sets.

Table 4.3 gives the results from solving IP1, and IP2 when relaxing the integrality of the variables x_{iv} and y_{jv}, and the linear relaxation of IP2 for each MFLP data set. IP1 was able to solve 26 of the 40 instances, while IP2 was able to solve 37 of the 40 instances. Furthermore, for the 26 instances solved in both formulations, CPLEX took on average 213.8% longer to solve IP1 than to solve IP2. In addition, IP2 was solved in less time than IP1 for 23 of those 26 instances. The IP-LP gap was generally small, with a maximum gap of 1.202%. In many cases, the solution to the LP relaxation produced integer valued variables, yielding an optimal solution to the MFLP. These results were generated by solving the LP relaxation of IP2, however since the optimal objective values of the linear relaxation of either IP1 or IP2 are equal, the IP-LP gap will be equal as well.

Our computational results indicate the heuristics proposed in this chapter provide effective methods for determining low cost solutions. Table 4.4 gives the

Scenario	Locations	Facilities	Clients
1	100	5	100
2	100	10	100
3	100	10	100
4	100	20	100
5	100	33	100
6	200	5	200
7	200	10	200
8	200	20	200
9	200	40	200
10	200	67	200
11	300	5	300
12	300	10	300
13	300	30	300
14	300	60	300
15	300	100	300
16	400	5	400
17	400	10	400
18	400	40	400
19	400	80	400
20	400	133	400
21	500	5	500
22	500	10	500
23	500	50	500
24	500	100	500
25	500	167	500
26	600	5	600
27	600	10	600
28	600	60	600
29	600	120	600
30	600	200	600
31	700	5	700
32	700	10	700
33	700	70	700
34	700	140	700
35	800	5	800
36	800	10	800
37	800	80	800
38	900	5	900
39	900	10	900
40	900	90	900

Tab. 4.2: The number of vertices, facilities and clients in each of our MFLP data sets.

optimality gap of the solution produced by 1-Swap, 1-SmartSwap, and 1-OptSwap as well as the runtime of 1-Swap, 1-SmartSwap, 1-OptSwap, and IP2 each MFLP instance. The results show that each local search heuristic produces low-cost solutions. Not surprisingly, 1-SmartSwap and 1-OptSwap generate equal or better solutions

Scenario	IP1 Obj	IP1 RT	IP2 Obj	IP2 RT	IP-LP Gap	LP2 RT
1	7231.77	3.094	7231.77	1.438	0.634%	0.516
2	4914.03	0.578	4914.03	0.250	0.000%	0.250
3	5792.74	0.531	5792.74	0.250	0.000%	0.188
4	4748.11	0.859	4748.11	0.281	0.000%	0.172
5	2170.57	1.125	2170.57	0.234	0.000%	0.125
6	8958.09	6.141	8958.09	5.547	0.019%	6.625
7	7241.09	3.047	7241.09	2.359	0.000%	2.563
8	6077.76	4.500	6077.76	2.016	0.000%	1.906
9	4348.86	5.719	4348.86	1.375	0.000%	0.844
10	2377.40	8.938	2377.40	1.313	0.000%	0.672
11	8444.63	22.438	8444.63	38.672	0.128%	17.547
12	9219.27	35.797	9219.27	24.000	0.056%	15.234
13	5487.23	27.219	5487.02	11.516	0.013%	5.641
14	3963.37	22.141	3963.37	4.047	0.000%	2.438
15	2642.84	34.594	2642.84	3.531	0.000%	1.922
16	9655.15	355.891	9655.15	186.188	0.882%	34.406
17	8300.98	108.391	8300.98	70.078	0.418%	26.859
18	5844.42	49.313	5844.42	31.422	0.002%	9.000
19	4229.19	56.406	4229.19	9.297	0.000%	4.922
20	3178.70	90.297	3178.70	7.703	0.000%	3.984
21	10908.30	23.391	10908.30	27.453	0.000%	61.578
22	10856.90	372.828	10856.90	433.922	0.606%	137.078
23	6756.93	67.625	6756.93	19.703	0.000%	14.328
24	O.O.M.	O.O.M.	4782.64	15.281	0.000%	8.938
25	D.N.L.	D.N.L.	3033.29	12.750	0.000%	5.734
26	13314.30	553.375	13314.30	490.734	0.716%	222.719
27	11200.00	234.641	11200.00	350.078	0.120%	228.969
28	O.O.M.	O.O.M.	6133.26	48.406	0.023%	22.781
29	D.N.L.	D.N.L.	4756.74	38.188	0.015%	15.031
30	D.N.L.	D.N.L.	3151.90	19.281	0.000%	8.266
31	12524.20	915.906	12524.20	439.281	0.646%	291.922
32	O.O.M.	O.O.M.	10743.60	743.063	0.118%	433.203
33	D.N.L.	D.N.L.	6740.80	45.469	0.000%	41.781
34	D.N.L.	D.N.L.	4507.58	75.766	0.016%	19.000
35	O.O.M.	O.O.M.	12408.10	3547.840	0.991%	423.531
36	O.O.M.	O.O.M.	12943.40	56590.700	1.202%	629.281
37	D.N.L.	D.N.L.	6296.17	196.422	0.023%	69.578
38	O.O.M.	O.O.M.	O.O.M.	O.O.M.	N/A	654.516
39	O.O.M.	O.O.M.	O.O.M.	O.O.M.	N/A	597.75
40	D.N.L.	D.N.L.	O.O.M.	O.O.M.	N/A	226.922

Tab. 4.3: Above shows the size of each data set, and gives a comparison of the performance of CPLEX when solving IP1 and IP2, and the LP-IP ratio found by solving the LP relaxation of IP2 (LP2). An entry "O.O.M." indicates CPLEX ran out of memory when attempting to solve the corresponding IP. An entry "D.N.L" indicates CPLEX could not load the problem into memory. The runtimes (RT) are displayed in seconds.

than 1-Swap in every instance, respectively reducing the optimality gap by factors of roughly two and four. Furthermore, 1-SmartSwap found the optimal solution in 12 of the 37 instances solved with IP2 and 1-OptSwap found the optimal solution in 13 of those instances, while 1-Swap found the optimal solution in only 3 of those instances. The runtime of 1-Swap and 1-SmartSwap were quick and comparable, while 1-OptSwap in most cases took longer to solve than either IP. On average, the execution time of 1-SmartSwap was only 2.04 seconds longer than 1-SmartSwap and at most 29.49 seconds longer in the largest instance.

These results indicate that 1-SmartSwap provides the best option of the three heuristics we implemented when $n = 1$ for quickly computing high quality solutions. In addition, the higher quality solutions produced by 1-OptSwap and 1-SmartSwap in these results highlights the importance of optimally solving the facility location subproblem.

One interesting characteristic of the MFLP is that the nature of the optimal solution to an MFLP instance changes as the ratio of the client weights to facility weights changes. Consider an instance of the MFLP set in the original framework. (i.e., set on a graph $G(V, E)$ where F and C are subsets of V.) When the client weights are small relative to the facility weights, there is less of an incentive to relocate the facilities. In the extreme, the optimal solution would be given by keeping each facility at its initial location and solving the client assignment subproblem. Conversely, when the client weights are very large relative to the facility weights, the incentive is to pick facility destinations that minimize the movement of the clients. In the extreme, the optimal solution to the MFLP would be the same as

the optimal solution to the p-median problem on the same graph. Thus, we ask the question, how does 1-SmartSwap perform as the demand parameters change?

To test this, we generated four variants from each of our MFLP instances, which we named "MFLP n-1" through "MFLP n-4". Each of the four variants of an MFLP instance have the same underlying network and p facility locations. The four variants differ by the weight of each client. Client weights are created for each variants by scaling the client weights in our MFLP instances that we described earlier. In variant 1, u_i is computed by scaling the client weights to lie in $[0, 0.1]$. Similarly, the values of u_i in variants 2, 3, and 4 are computed by scaling the client weights to lie in $[0.1, 1]$, $[1, 10]$ and $[10, 100]$, respectively. The vertices in the graph, the length of the edges in the graph, the facility weights, and the initial locations of facilities and clients in each variant are the same in the original instance. Notice that variant 3 of an instance is identical the original MFLP instance that we described at the start of this section. Thus, the results above may also be considered as results for variant 3.

Tables 4.5 and 4.6 provide the optimality gap and runtime of 1-SmartSwap as well as the CPLEX runtime when solving IP2 for each variant of each MFLP instance. The results show 1-SmartSwap performs well on a variety of client weights. 1-SmartSwap produces a lower optimality gap on the lower number variants, when the client weights u_i lie in a lower range. However, as noted previously, lower client weights provide a greater incentive to keep each facility fixed at its initial location, and move clients longer distances. Since these local search heuristics keep each facility fixed in the initial solution, they are more likely to find the optimal solution

Data Set	1-SmartSwap		1-OptSwap		1-Swap		IP2
	Gap	Runtime	Gap	Runtime	Gap	Runtime	Runtime
1	0.00%	0.016	0.00%	0.047	0.32%	0.000	1.438
2	0.00%	0.047	0.00%	0.172	0.31%	0.047	0.250
3	0.00%	0.032	0.00%	0.125	0.00%	0.031	0.250
4	1.30%	0.078	0.32%	1.015	1.30%	0.078	0.281
5	0.58%	0.094	0.58%	2.078	0.58%	0.094	0.234
6	0.00%	0.063	0.00%	0.063	0.84%	0.047	5.547
7	0.00%	0.203	0.00%	0.422	0.94%	0.188	2.359
8	0.03%	0.531	0.03%	2.766	2.01%	0.453	2.016
9	0.63%	1.484	0.24%	24.063	1.97%	1.312	1.375
10	1.60%	1.702	0.22%	106.172	1.60%	1.672	1.313
11	0.86%	0.094	0.86%	0.109	0.86%	0.094	38.672
12	0.00%	0.656	0.00%	1.016	0.26%	0.625	24.000
13	1.23%	2.749	0.02%	23.453	2.05%	2.734	11.516
14	1.06%	9.655	0.63%	243.359	1.57%	8.969	4.047
15	0.00%	10.530	0.00%	962.234	0.00%	10.359	3.531
16	0.04%	0.343	0.00%	0.375	0.09%	0.312	186.188
17	0.00%	0.781	0.00%	1.312	0.36%	0.719	70.078
18	0.10%	14.091	0.05%	108.984	0.29%	13.750	31.422
19	1.00%	33.682	0.36%	1436.700	2.82%	27.594	9.297
20	0.97%	32.354	0.84%	8802.000	0.97%	32.562	7.703
21	0.00%	0.453	0.00%	0.407	0.04%	0.375	27.453
22	0.00%	2.718	0.00%	3.156	0.99%	2.500	433.922
23	0.67%	43.274	0.09%	467.703	1.98%	38.078	19.703
24	0.76%	67.426	0.28%	4057.610	1.77%	65.656	15.281
25	0.80%	85.907	0.38%	36432.500	0.86%	82.985	12.750
26	0.48%	0.813	1.02%	0.750	0.60%	0.703	490.734
27	0.00%	3.234	0.00%	3.156	0.00%	3.063	350.078
28	0.39%	80.112	0.02%	1211.500	1.40%	79.328	48.406
29	1.02%	150.381	0.17%	11585.900	1.23%	148.672	38.188
30	0.31%	176.627	0.23%	119512.000	0.37%	172.032	19.281
31	0.02%	0.875	0.02%	0.954	0.02%	0.875	439.281
32	0.02%	4.624	0.02%	5.328	0.26%	4.344	743.063
33	0.75%	147.554	0.10%	2228.391	1.44%	146.219	45.469
34	0.60%	259.613	0.31%	27091.734	0.78%	258.578	75.766
35	0.00%	1.672	0.00%	1.547	0.01%	1.672	3547.840
36	0.53%	5.859	0.34%	8.125	0.72%	5.453	56590.700
37	0.67%	248.521	0.02%	3681.969	1.17%	225.562	196.422
38	1.11%*	2.781	1.11%*	2.625	1.11%*	2.422	O.O.M
39	0.55%*	10.123	0.55%*	11.468	0.92%*	9.328	O.O.M
40	0.98%*	431.131	0.04%*	8822.156	1.79%*	401.641	O.O.M
Average	0.45%	45.822	0.22%	5671.147	0.85%	43.778	1716.104

Tab. 4.4: The optimality gap for each heuristic as well as the runtime (RT) in seconds of each heuristic with $n = 1$, and of IP2 for each of our 40 MFLP instances. The entries "O.O.M" indicate that IP2 could not be solved because CPLEX ran out of memory. An asterisk indicates the gap was calculated using the optimal solution to the LP relaxation of IP2 as a lower bound.

in such cases. Thus, the increase in the average optimality gap as client weights increase may not necessarily indicate a decrease in the ability of 1-SmartSwap to find high quality solutions. In particular, the average optimality gap for variant 4, where the client weights lie in $[10, 100]$ is less than the average optimality gap for variant 3, where client weights lie in the lower range, $[1, 10]$.

2-SmartSwap explores a larger neighborhood that 1-SmartSwap, giving it the potential to produce higher quality solutions. While 1-SmartSwap explores a neighborhood with $|F|(|V| - |F|)$ solutions, 2-SmartSwap explores a much larger neighborhood containing $\frac{|F|(|F|-1)}{2} \cdot \frac{(|V|-|F|)(|V|-|F|-1)}{2}$ solutions. In other words, for each solution in the neighborhood explored by 1-SmartSwap, there are $\frac{(|F|-1)(|V|-|F|-1)}{4}$ solutions in the neighborhood explored by 2-SmartSwap. With so many more solutions, the runtime of 2-SmartSwap typically exceeds the runtime of 1-OptSwap and the time CPLEX needs to solve either IP1 or IP2. Table 4.7 gives results from implementing 2-SmartSwap on (variant 3 of) each data set. (We return to using our original data sets, which are the same as variant 3, since 1-SmartSwap produced the highest average optimality gap on these instances.) Because each iteration of 2-SmartSwap is not quick, our implementation of 2-SmartSwap is initialized with the solution found with 1-SmartSwap. The runtime of 2-SmartSwap was limited to two hours. After executing for two hours, the best improvement found during the current iteration of 2-SmartSwap was implemented. The facility assignment subproblem was then resolved and 2-SmartSwap was terminated.

These results show that 2-SmartSwap was able to find a local minimum in two hours for only 14 of the 40 MFLP instances. In seven of the 40 data sets,

	Variant 1			Variant 2			Variant 3			Variant 4		
Instance	Gap	1-SmartSwap Runtime	IP2 Runtime	Gap	1-SmartSwap Runtime	IP2 Runtime	Gap	1-SmartSwap Runtime	IP2 Runtime	Gap	1-SmartSwap Runtime	IP2 Runtime
1	0.00%	0.00	0.38	0.00%	0.00	2.13	0.00%	0.02	1.44	0.00%	0.02	3.08
2	0.00%	0.00	0.08	0.00%	0.00	0.33	0.00%	0.05	0.25	0.00%	0.06	0.47
3	0.00%	0.00	0.19	0.00%	0.00	0.45	0.00%	0.03	0.25	0.00%	0.05	0.47
4	0.00%	0.00	0.19	0.00%	0.00	0.36	1.30%	0.08	0.28	0.00%	0.13	0.44
5	0.00%	0.00	0.20	0.00%	0.02	0.38	0.58%	0.09	0.23	0.58%	0.19	0.45
6	0.00%	0.00	0.69	0.00%	0.03	22.81	0.00%	0.06	5.55	0.00%	0.08	5.27
7	0.00%	0.00	0.94	0.00%	0.13	11.52	0.00%	0.20	2.36	0.00%	0.20	7.92
8	0.00%	0.00	0.67	0.00%	0.09	2.31	0.03%	0.53	2.02	0.00%	0.64	2.77
9	0.00%	0.03	1.23	0.00%	0.11	2.06	0.63%	1.48	1.38	0.00%	2.12	2.52
10	0.00%	0.05	0.95	0.00%	0.16	2.02	1.60%	1.70	1.31	0.70%	4.23	2.61
11	0.00%	0.00	1.08	0.00%	0.09	11.30	0.86%	0.09	38.67	1.24%	0.13	39.27
12	0.00%	0.13	23.56	0.00%	0.33	33.80	0.00%	0.66	24.00	0.00%	0.75	32.94
13	0.00%	0.03	1.11	0.00%	0.56	7.42	1.23%	2.75	11.52	0.09%	4.06	9.36
14	0.00%	0.05	1.05	0.00%	0.47	7.06	1.06%	9.66	4.05	0.66%	13.81	8.69
15	0.00%	0.09	1.44	0.00%	0.31	7.02	0.00%	10.53	3.53	0.67%	20.36	9.31
16	0.00%	0.05	20.23	0.00%	0.34	97.75	0.04%	0.34	186.19	0.08%	0.31	326.94
17	0.00%	0.02	4.88	0.00%	0.64	28.64	0.00%	0.78	70.08	0.00%	0.86	123.64
18	0.00%	0.06	1.69	0.00%	3.45	16.50	0.10%	14.09	31.42	0.04%	18.09	79.59
19	0.00%	0.28	5.19	0.00%	2.36	16.56	1.00%	33.68	9.30	0.31%	42.18	19.72
20	0.00%	0.49	7.77	0.00%	0.77	16.58	0.97%	32.35	7.70	0.76%	76.32	20.95

Tab. 4.5: The optimality gap and runtime in seconds of 1-SmartSwap and the runtime in seconds of CPLEX when solving IP2 for variants 1 through 4 of the first twenty instances. The instance is listed on the left hand side and the variant is listed on top.

	Variant 1			Variant 2			Variant 3			Variant 4		
	Gap	1-SmartSwap Runtime	IP2 Runtime	Gap	1-SmartSwap Runtime	IP2 Runtime	Gap	1-SmartSwap Runtime	IP2 Runtime	Gap	1-SmartSwap Runtime	IP2 Runtime
21	0.00%	0.11	45.61	0.00%	0.45	255.06	0.00%	0.45	27.45	0.00%	0.45	44.22
22	0.00%	0.36	52.89	1.22%	2.22	247.38	0.00%	2.72	433.92	0.00%	2.70	504.16
23	0.00%	0.58	18.56	0.00%	8.48	27.97	0.67%	43.27	19.70	0.25%	48.87	35.09
24	0.00%	0.94	18.31	0.00%	7.08	27.78	0.76%	67.43	15.28	0.66%	109.70	34.17
25	0.00%	0.55	6.41	0.00%	1.56	31.06	0.80%	85.91	12.75	0.63%	181.35	172.28
26	1.20%	0.38	132.84	0.63%	0.91	1163.52	0.48%	0.81	490.73	1.01%	0.81	780.23
27	0.00%	0.72	32.83	0.00%	2.48	70.05	0.00%	3.23	350.08	0.00%	2.98	314.70
28	0.00%	0.81	18.84	0.30%	12.72	48.13	0.39%	80.11	48.41	0.15%	91.78	54.61
29	0.00%	1.56	26.25	0.00%	8.75	42.52	1.02%	150.38	38.19	0.27%	245.93	52.34
30	0.00%	0.83	8.56	0.00%	5.69	44.34	0.31%	176.63	19.28	0.95%	389.19	60.19
31	0.00%	0.36	107.56	0.00%	0.89	473.22	0.02%	0.88	439.28	0.04%	0.89	712.66
32	0.00%	0.22	18.17	0.34%	4.69	1549.34	0.02%	4.62	743.06	0.00%	5.44	644.08
33	0.00%	4.56	34.44	0.00%	35.12	62.45	0.75%	147.55	45.47	0.04%	183.08	595.64
34	0.00%	1.75	23.67	0.00%	10.94	56.80	0.60%	259.61	75.77	0.14%	485.65	77.59
35	0.00%	0.34	108.42	0.03%	1.27	1599.11	0.00%	1.67	3547.84	0.00%	1.67	1491.14
36	0.00%	1.72	220.95	0.00%	6.39	61156.50	0.53%	5.86	56590.70	0.47%	7.53	15665.90
37	0.00%	0.86	12.55	0.00%	28.48	95.86	0.67%	248.52	196.42	0.13%	257.44	920.92
38	0.00%	1.19	483.75	O.O.M.	2.22	O.O.M.	1.11%*	2.78	O.O.M.	O.O.M.	2.78	O.O.M.
39	0.00%	0.17	15.70	O.O.M.	7.91	O.O.M.	0.55%*	10.12	O.O.M.	O.O.M.	10.11	O.O.M.
40	0.00%	3.03	53.16	0.00%	73.08	116.23	0.98%*	431.13	O.O.M.	O.O.M.	510.13	O.O.M.
Average	0.03%	0.56	37.83	0.07%	5.78	1772.48	0.45%	45.82	1716.10	0.27%	68.08	617.74

Tab. 4.6: The optimality gap and runtime in seconds of 1-SmartSwap and the runtime in seconds of CPLEX when solving IP2 for variants 1 through 4 of the second twenty instances. The average results from all forty instance is also presented in the last row. The instance is listed on the left hand side and the variant is listed on top. An asterisk indicates the gap was calculated using the optimal solution to the LP relaxation of IP2 as a lower bound.

2-SmartSwap was able to improve the solution found by 1-SmartSwap. While 2-SmartSwap could potentially find further improvements to the solutions produced by 1-SmartSwap if the runtime was not limited, there do exist data sets, such as data sets 8 and 9, for which 2-SmartSwap could not find any improvements to the solution found by 1-SmartSwap without being limited by time. Conversely, 2-SmartSwap also found significant improvements in other cases, such as when it produced the optimal solution to data set 5. However, overall 2-SmartSwap was able to find rather limited improvements to the solutions found by 1-SmartSwap while typically requiring more time to execute than 1-OptSwap, even when limiting execution time to 2 hours. Furthermore, the solutions found by 1-OptSwap were superior to those found by 2-SmartSwap. When limiting execution time, it may be more effective to greedily search the neighborhood explored by 2-SmartSwap. However, in 26 of the 40 instances, 2-SmartSwap was unable to complete even a single iteration in two hours. In such cases, a greedy heuristic would also not be capable of exploring the entire neighborhood once in two hours. With a large number of solutions in each neighborhood, the chances of finding an improvement might also decrease as the size of the problem gets larger since the percentage of the neighborhood explored in the limited execution time would decrease.

2-OptSwap also displayed unreasonably long runtimes. Table 4.8 displays the runtimes of 2-OptSwap on the first nine data sets without any limitation on the runtime. The runtime of 2-OptSwap was excessively long, even for relatively small data sets. 2-OptSwap took more than 13 hours to terminate when solving data set 9. 2-OptSwap did not terminate in 72 hours for data set 10. As a result of

Data Set	Gap	Runtime (s)	Iterations Completed
1	0.00%	2.109	1
2	0.00%	8.578	1
3	0.00%	8.484	1
4	0.82%	84.266	3
5	0.00%	161.297	3
6	0.00%	58.391	1
7	0.00%	287.532	1
8	0.03%	944.766	1
9	0.63%	4004.969	1
10	1.42%	7200.015	0
11	0.00%	2025.719	4
12	0.00%	1743.610	1
13	1.23%	7200.016	0
14	1.06%	7200.016	0
15	0.00%	7200.032	0
16	0.00%	4333.797	3
17	0.00%	6235.031	1
18	0.10%	7200.016	0
19	1.00%	7200.031	0
20	0.89%	7200.063	0
21	0.00%	3818.844	1
22	0.00%	7200.015	0
23	0.67%	7200.015	0
24	0.76%	7200.031	0
25	0.80%	7200.125	0
26	0.00%	7200.016	0
27	0.00%	7200.016	0
28	0.39%	7200.016	0
29	1.02%	7200.062	0
30	0.31%	7200.203	0
31	0.02%	7200.016	0
32	0.02%	7200.016	0
33	0.75%	7200.031	0
34	0.60%	7200.094	0
35	0.00%	7200.015	0
36	0.53%	7200.016	0
37	0.67%	7200.047	0
38	1.11%*	7200.015	0
39	0.98%*	7200.015	0
40	0.55%*	7200.047	0
Average	**0.41%**	**5272.96**	**0.575**

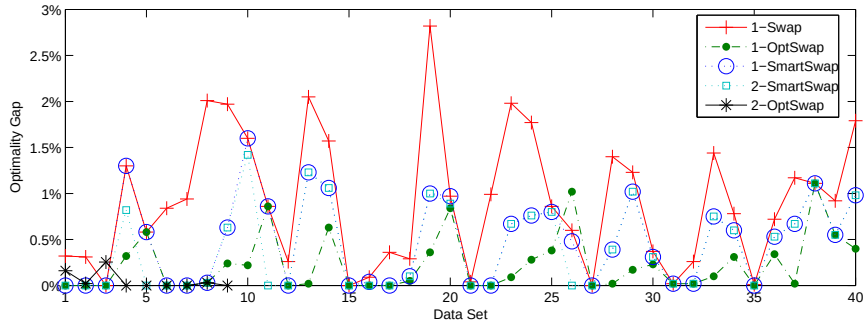
Tab. 4.7: The optimality gap and runtime of 2-SmartSwap on (variant 3 of) each data set. 2-SmartSwap was initialized with the solution found by 1-SmartSwap. The runtime of 2-SmartSwap was limited to two hours. If 2-SmartSwap could not terminate in two hours, the best solution found so far in the current iteration was implemented and the facility assignment subproblem was resolved one final time. An asterisk indicates the gap was calculated with the solution to the LP relaxation of IP2 as a lower bound.

2-OptSwap		
Data Set	Gap	Runtime
1	0.16%	1.703
2	0.02%	22.438
3	0.25%	18.469
4	0.00%	488.312
5	0.00%	1827.937
6	0.00%	12.906
7	0.00%	110.5
8	0.03%	2387.718
9	0.00%	49818.75
Average	**0.05%**	**6076.526**

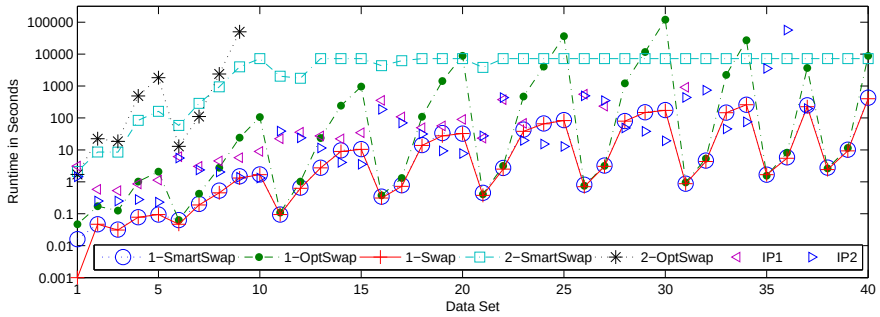
Tab. 4.8: The Performance of 2-OptSwap on (variant 3 of) the first nine data sets.

these extremely long runtimes, results for larger instances were not computed. The neighborhood explored by 2-OptSwap has the large same size as the neighborhood explored by 2-SmartSwap; however, 2-OptSwap must spend significantly more time to compute each solution in the neighborhood.

Figure 4.6(a) displays a graph comparing the optimality gaps of the results reported for (variant 3 of) each data set. Similarly, Figure 4.6(b) shows a graph comparing the runtimes of each local search heuristic implemented as well as the time CPLEX took to solve each data set. We believe these demonstrate that 1-SmartSwap provides the best choice of the methods presented in this chapter for quickly computing efficient solutions to the MFLP. While the runtime of 1-SmartSwap is very close to the runtime of 1-Swap local search, the solutions it finds very often come close to the best solution we computed with a heuristic. In addition, the runtime of 1-SmartSwap typically outperforms the time CPLEX needed to solve IP1 or IP2. This appears to be particularly true when the number of mobile facilities is small compared to the number of locations.



(a) Optimality Gap of Local Search Heuristics



(b) Runtimes of Local Search Heuristics, IP1, and, IP2

Fig. 4.6: Plots of the optimality gaps and runtimes from Table 4.3, Table 4.4, Table 4.7, and Table 4.8. Panel 4.6(a) gives a comparison of the optimality gap of the computational results presented for the local search heuristics. Panel 4.6(b) gives a comparison of the runtimes on a logscale of the computational results presented on the local search heuristics and the runtimes of CPLEX when solving IP1 and IP2.

4.6 Lagrangian Heuristics for the MFLP

The local search heuristics presented in this chapter provide effective methods for finding high quality solutions to the MFLP. Another approach for computing solutions to the MFLP would be to solve the Lagrangian dual of either IP1 or IP2. Such Lagrangian heuristics have been effective in solving many problems that can be formulated as integer programs [24], including problems in locating facilities such as the uncapacitated facility location problem [12], the p-median problem [35], and the maximal covering location problem [27].

A Lagrangian dual for the MFLP can be formulated by relaxing Constraint (4.2.5) in IP1 to give,

$$\text{Minimize} \qquad \text{SP}(\boldsymbol{\lambda}) = \sum_{i \in C} \sum_{v \in V} \left(u_i d_{iv} + \lambda_{iv}\right) x_{iv} + \sum_{j \in F} \sum_{v \in V} \left(w_j d_{jv} - \sum_{i \in C} \lambda_{iv} \right) y_{jv} \tag{4.6.1}$$

subject to (4.2.3), (4.2.4), and (4.2.6).

For fixed values of λ_{iv}, the problem can be decomposed into polynomially solvable subproblems of assigning each facility j to the vertex v minimizing $w_j d_{jv} - \sum_{i \in C} \lambda_{iv}$ and assigning each client i to the vertex v minimizing $u_i d_{iv} + \lambda_{iv}$. This yields a feasible solution for the Lagrangian, and thus a lower bound for the MFLP. An upper bound may then be computed by sending the facilities to the same destination as in the feasible solution to the Lagrangian, and then solving the client assignment subproblem (i.e., assign each client to its nearest facility). However, the solution to the MFLP found in this manner may have one vertex v as the destination of multiple facilities. If this is the case, we improve the solution by moving only the mobile facility with destination v with the weighted distance to vertex v, and keeping the other facilities with destination vertex v at their original location prior to solving the client assignment subproblem.

We solved the Lagrangian dual using subgradient optimization. However, we found that the Lagrangian dual converges very slowly and the solutions it obtained for the MFLP were of poor quality.

4.7 Conclusion

We have shown for each set of facility locations Z, the MFLP decomposes into the facility assignment subproblem and the client assignment subproblem, which can both be solved in polynomial time. Thus, the MFLP can be viewed as the problem of determining the set Z yielding the least cost solution. Using this viewpoint, we presented an improved IP formulation for the MFLP with a smaller memory footprint, allowing a commercial solver like CPLEX to solve much larger instances than possible with an earlier model for the MFLP.

In addition, we have introduced a new framework for the MFLP that permits a more general cost structure. In this new framework, the MFLP generalizes both the p-median and uncapacitated facility location problems. The local search heuristics discussed in this chapter reduce to well known local search heuristics for these other problems that have known constant factor approximation algorithms.

To appropriately design heuristics for the MFLP, it is necessary to resolve the facility assignment subproblem optimally. Friggstad and Salvatipour [25] produced an example where a local search heuristic may terminate at a solution with an arbitrarily large locality gap. We proposed and tested two local search heuristics that always produce solutions with the facility subproblem solved optimally. Our computational results confirm the effectiveness of our local search heuristics. In particular, 1-SmartSwap quickly finds high quality solutions for large instances of the MFLP. Whether either n-OptSwap or n-SmartSwap provide a constant factor approximation ratio remains as a open question.

5. CONCLUDING REMARKS AND FUTURE RESEARCH

In the past several chapters, we have discussed two problems on modeling the efficient deployment of mobile facilities. These models have wide-ranging applications in telecommunication networks, humanitarian relief logistics, public sector services, health care, and private industry. We have presented several innovative heuristics, exact algorithms, and novel formulations for these problems. The effectiveness of these heuristics and computational tractability of these formulations was demonstrated with computational experiments on large numbers of heterogeneous data sets. Many interesting research questions arise from the study of these models. Below we describe the unique contributions of this work and potential directions of future research.

5.1 The SMFRP

In Chapter 2, we described a widely applicable model for routing a single mobile facility to maximize the total demand serviced during a continuous-time planning horizon. We named this problem the single mobile facility routing problem (SMFRP). The SMFRP can be used to model a large class of mobile facilities that are used in many application settings. We introduced two exact algorithms for solving the SMFRP when the moment demand functions are piecewise constant, and a number of supporting theoretical results describing characteristics of optimal routes

and methods for improving the runtime of these algorithms. The first of these exact algorithms, the schedule resolution dynamic program (SRDP), may have a worst case exponential runtime, although it typically executes quickly. In addition, the SRDP naturally extends to cases when the moment demand functions take on more general forms. The second of these two exact algorithms is the single mobile facility longest path algorithm (SMFLPA), and has a worst case runtime that is polynomial in the number of pieces of the piecewise constant moment demand functions.

We have shown how the SMFRP can be extended to several variants of the problem. The SRDP (and the SMFLPA) can be modified to optimally solve many cases when the moment demand functions are not piecewise linear. In addition, the SMFRP can easily be extended to model the requirement that the mobile facility starts and ends its route at a depot or at specific locations by simply modifying the moment demand functions. The SMFLPA permits the addition of relocation costs to the SMFRP, which allows the most profitable route to be found in applications where demand serviced can be equated with revenue.

There are several interesting and useful variants of the SMFRP that could be fruitful topics of future research. These are driven by service constraints an operator of a mobile facility may impose in practice. For example, it may be deemed appropriate for a mobile facility to visit a location only if it can remain there for a minimum length of time. This constraint may reconcile the mathematical solution with the realities of operating a mobile facility in practice. In such a case, an optimal route must be computed where the mobile facility remains at each stop for a minimum duration of time. Alternatively, an operator may wish to encourage the

mobile facility to visit a number of locations, in which case they could impose a constraint specifying the minimum number of locations that must be visited, or a maximum amount of time a mobile facility may be at a location.

5.2 The MFRP

The MFRP introduced in Chapter 3 extends the SMFRP to the case when multiple mobile facilities are operating in an area. We proposed several novel heuristics for solving the MFRP and demonstrated their effectiveness through computational results on a wide range of test instances. The SMFRP and MFRP are unique among mobile facility models in that these models are both set in a continuous-time planning horizon and seek to maximize the demand serviced. The SMFRP and MFRP are two of the few models of mobile facility operations set in a continuous-time planning horizon. In fact, we are aware of only one other [8]. Traditionally, most facility location problems are set in either a single period planning horizon, or a multi-period planning horizon. This continuous-time planning horizon allows the SMFRP and MFRP to more accurately model the relocation times of the mobile facilities. This is important in settings where facility relocation times are significant in relation to the length of the planning horizon and mobile facilities can not provide service while relocating, such as when routing portable cellular base stations or mobile medical facilities.

An exact method for solving the MFRP remains a topic of future research. We can model the MFRP as an infinite dimensional mixed integer program. Unfortunately, no general methods are known for solving infinite dimensional integer

programs. One possible method for solving the MFRP would be to determine a finite set of times when a mobile facility could either depart a location, or arrive at a location. If this set of times was known, an optimal solution for the MFRP could be computed by solving the integer program given by (3.4.8)-(3.4.13).

Since the MFRP is NP-complete, any exact method found likely has an exponential worst case runtime. However, an exact method would be useful for evaluating the performance of heuristics for the MFRP. In lieu of an exact solution method for the MFRP, a procedure for producing a quality upper bounding would be useful for evaluating these heuristics and in developing improved heuristics in the future.

Both the SMFRP and MFRP assume that demand is either serviced at the moment it is generated, or the demand is lost. However, in some applications the demand for service may accumulate over time. For example, when using mobile facilities to provide humanitarian relief after a natural or manmade disaster, an individual who arrives at a location in need of food, water, or other supplies may continue to need these items if a mobile facility is not around to provide them. Perhaps models similar to the SMFRP and MFRP can be developed for such situations?

The MFRP also assumes the planner has perfect control of assigning demand from event points to mobile facilities. However, this may not be possible in every application. For example, the population may choose which mobile facility to travel to, giving little control to the operator of the facility. In some cases, the heuristics described in Chapter 3 may be modified to handle such situations. For example, the demand assignment phase of our heuristics for the MFRP can be easily modified

when service is always provided by the closest mobile facility. In other cases, it may be that the heuristics described in Chapter 3 provide effective routes even if the method for demand assignment in the heuristics does not exactly mirror reality. However, there may exist other situations requiring new solution methods.

The SMFRP and MFRP make an assumption that the demand profile of each location and event point, respectively, is known perfectly for the entire planning horizon. In some applications, this may not be the case. The rate demand for service is generated at an event could deviate from a predicted demand profile over time. In addition, there may exist cases where the demand profile cannot be accurately forecasted. The extension of the SMFRP and MFRP to stochastic environments may prove to be interesting topics of future research.

5.3 The MFLP

In Chapter 4 we demonstrated that given a set of locations to be occupied by facilities, the MFLP can be decomposed into two polynomially solvable subproblems of assigning customers and facilities to specific locations in this set. We named these subproblems the facility assignment subproblem and the customer assignment subproblem. With this in hand, we proposed a new IP formulation for the MFLP that permits a commercial solver, such as CPLEX, to solve large scale instances. In addition, we used this decomposition to devise novel local search heuristics for solving the MFLP. These local search heuristic always terminate at a solution where the facility assignment subproblem and customer assignment subproblem are solved optimally. As a result these local search heuristics will never terminate in the local

minimum demonstrated by Friggstad and Salavatipour [25], which is arbitrarily far away from optimal. We demonstrated the effectiveness of these local search heuristics on a wide variety of data sets. In addition, we proposed a new framework for the MFLP that allows the MFLP to model more general cost structures. In this new framework, the MFLP generalizes both the p-median and uncapacitated location problems.

It remains an open question whether either n-OptSwap or n-SmartSwap is a constant ratio approximation algorithm. If so, these heuristics would provide a simpler constant factor approximation algorithm than the LP rounding procedure introduced by Friggstad and Salavatipour [25].

It is easy to produce several simple variants of the MFLP that could be interesting problems. For example, one could introduce capacity constraints on the amount of demand or number of clients that may be assigned to a facility. Alternatively, one could impose a constraint on the maximum distance that a facility or client may travel in a solution. Another interesting variant would be a multi-objective formulation of the problem, where the two objectives to be minimized are the maximum movement of some client and the total movement of all facilities. This may closely mirror situations where the planner wants to minimize the costs of relocating facilities while ensuring that individual clients move as little as possible.

BIBLIOGRAPHY

[1] Betty B. Alexy and Christine A. Elnitsky. Community outreach: Rural mobile health unit. *Journal of Nursing Administration*, 26(12):38–42, December 1996.

[2] Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristic for k-median and facility location problems. In *STOC '01: Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 21–29, New York, NY, USA, 2001. ACM.

[3] R.H. Ballou. Dynamic warehouse location analysis. *Journal of Marketing Research*, 5(3):271–276, August 1968.

[4] J.E. Beasley. OR-library: Distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41(11):1069–1072, 1990.

[5] O. Berman and B. LeBlanc. Location-relocation of mobile facilities on a stochastic network. *Transportation Science*, 18(4):315–330, November 1984.

[6] Oded Berman and Mina R. Rahnama. Optimal location-relocation decisions on stochastic networks. *Transportation Science*, 19(3):203–221, August 1985.

[7] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 1995.

[8] S. Bespamyatnikh, B. Bhattacharya, D. Kirkpatrick, and M. Segal. Mobile facility location. In *Proc. Int. ACM Workshop of Discrete Algorithms and Methods for Mobile Computing and Communications*, volume 4, pages 199–217, 2000.

[9] Luce Brotcorne, Gilbert Laporte, and Frederic Semet. Ambulance location and relocation models. *European Journal of Operational Research*, 147(3):451–463, June 2003.

[10] C. Canel, B. M. Khumawala, J. Law, and A. Loh. An algorithm for the capacitated, multi-commodity multi-period facility location problem. *Computers and Operations Research*, 28(5):411–427, April 2001.

[11] R. L. Church, D. M. Stoms, and F. W. Davis. Reserve selection as a maximal covering location problem. *Biological Conservation*, 76(2):105–112, 1996.

[12] Gerard Cornuejols, Marshall L. Fisher, and George L. Nemhauser. Location of bank accounts to optimize float: An analytic study of exact and approximate algorithms. *Management Science*, 23(8):789–810, 1977.

[13] J. R. Current and J. E. Storbeck. Capacitated covering models. *Environment and Planning B: Planning and Design*, 15(2):153–163, 1988.

[14] J.R. Current, H. Pirkul, and E. Rolland. Efficient algorithms for solving the shortest covering path problem. *Transportation Science*, 28(4):317–327, November 1994.

[15] J.R. Current, C.S. ReVelle, and J. Cohon. The shortest covering path problem: An application of locational constraints to network design.

[16] J.R. Current, C.S. ReVelle, and J.L. Cohon. The maximum covering/shortest path problem: A multiobjective network design and routing formulation. *European Journal of Operational Research*, 21:189–199, 1985.

[17] J.R. Current and David A. Schilling. The covering salesman problem. *Transportation Science*, 23(3):208–213, August 1989.

[18] J.R. Current and David A. Schilling. The median tour and maximal covering tour problems: Formulations and heuristics. *European Journal of Operational Research*, 73:114–126, 1994.

[19] M. Daskin. *Network and Discrete Location*. John Wiley & Sons, Inc, New York, NY, 1995.

[20] Erik D. Demaine, MohammadTaghi Hajiaghayi, Hamid Mahini, Amin S. Sayedi-Roshkhar, Shayan Oveisgharan, and Morteza Zadimoghaddam. Minimizing movement. In *SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, 2007.

[21] S. Durocher and D. Kirkpatrick. The Steiner centre of a set of points: Stability, eccentricity, and applications to mobile facility location. *International Journal of Computational Geomerty and Applications*, 16(4):345–371, 2006.

[22] Jack Edmonds and Richard M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *J. ACM*, 19(2):248–264, 1972.

[23] Donald Erlenkotter. A comparative study of approaches to dynamic location problems. *European Journal of Operational Research*, 6(2):133–143, February 1981.

[24] Marshall L. Fisher. The lagrangian relaxation method for solving integer programming problems. *Management Science*, 27(1):1–18, 1981.

[25] Zachary Friggstad and Mohammad R. Salavatipour. Minimizing movement in mobile facility location problems. In *FOCS '08: Proceedings of the 2008 49th Annual IEEE Symposium on Foundations of Computer Science*, 2008.

[26] Harold N. Gabow and Robert E. Tarjan. Faster scaling algorithms for network problems. *SIAM Journal on Computing*, (5):1013–1036, 1989.

[27] Roberto Diguez Galvo and Charles ReVelle. A lagrangean heuristic for the maximal covering location problem. *European Journal of Operational Research*, 88:114–123, 1996.

[28] M. Gendreau, G. Laporte, and F. Semet. A dynamic model and parallel tabu search heuristic for real-time ambulance relocation. *Parallel Computing*, 27(12):1641–1653, 2001.

[29] Hongkong Post. Mobile post office locations. `http://www.hongkongpost.com/eng/locations/mobile/index.htm`, December 1 2008. Viewed Jan 21, 2009.

[30] M. L. Jay. Katrina leaves wireless industry better prepared for next disaster. `http://www.ctia.org/content/index.cfm/AID/10410`, 2006.

[31] J.E.Beasley. A note on solving large p-median problems. *European Journal of Operational Research*, 21:270–273, 1985.

[32] Marisa G. Kantor and Moshe B. Rosenwein. The orienteering problem with time windows. *The Journal of the Operational Research Society*, 43(6):629–635, June 1992.

[33] Orhan Karasakal and Esra K. Karasakal. A maximal covering location model in the presence of partial coverage. *Computers and Operations Research*, 31(9):1515–1526, 2004.

[34] Peter Kolesar and Warren E. Walker. An algorithm for the dynamic relocation of fire companies. *Operations Research*, 22(2):249–274, 1974.

[35] Lazaros P. Mavrides. An indirect method for the generalized k-median problem applied to lock-box location. *Management Science*, 25(10):990–996, 1979.

[36] Pitu B. Mirchandani and Richard L. Francis. *Discrete Location Theory*. Wiley-Interscience, July 1990.

[37] Mike Moss, Bruce Alan, and Joe Farren. COLTs and COWs to help with cell phone demand, 2008. December 10, 2008.

[38] James Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38, March 1957.

[39] Rahul Nair and Elise Miller-Hooks. Evaluation of relocation strategies for emergency medical service vehicles. *Transportation Research Record*, 2010.

[40] H. Pirkul and D. A. Schilling. The maximum covering location problem with capacities on total workload. *Management Science*, 37(2):233–248, February 1991.

[41] T. J. Van Roy and D. Erlenkotter. A dual-based procedure for dynamic facility location. *Management Science*, 28(10):1091–1105, October 1982.

[42] Aamod Sathe and Elise Miller-Hooks. Optimizing location and relocation of response units in guarding critical facilities. *Transportation Research Record*, 1923:127–136, 2005.

[43] Smiths Detection. Smiths detection security technologies selected by U.S. postal inspection service for mobile screening. `http://www.smithsdetection.com/eng/1025_4071.php`, January 18, 2009. Viewed January 21, 2009.

[44] N. Tomizawa. On some techniques useful for solution of transportation network problems. *Networks*, 1, 1971.

[45] Theodore Tsiligride. Heuristic methods applied to orienteering. *Journal of the Operational Research Society*, 35(9):797–809, September 1984.

[46] United Kingdom, Department of Transportation. Mobile post office website. `http://www.dft.gov.uk/pgr/regional/ltp/accessibility/knowledgepool/aos/mobilepostoffice`, 2009. Viewed Jan 21, 2009.

[47] United States Postal Service. Postal facts 2008. `http://www.usps.com/communications/newsroom/postalfacts.htm`, 2008. Viewed January 21, 2009.

[48] M. Wendland. Cell capacity will be super during the big game in Detroit. *Detroit Free Press*, 2006. February 1, 2006.

[49] G. O. Wesolowsky. Dynamic facility location. *Management Science*, 19(11):1241–1248, 1973. Theory Series.

[50] G. O. Wesolowsky and W. G. Truscott. The multiperiod location-allocation problem with relocation of facilties. *Management Science*, 22(1):57–65, 1975.

[51] Laurence A. Wolsey. *Integer Programming*. Wiley-Interscience, September 1998.