VEHICLE ROUTING PROBLEMS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Patrick R. Steele

January 2017

VEHICLE ROUTING PROBLEMS

Patrick R. Steele, Ph.D.

Cornell University 2017

In this dissertation we consider variants of the vehicle routing problem applied to two problem areas. First, we consider the problem of scheduling deliveries from a central depot to clients in a metric space using a single delivery vehicle. Although this problem involves only a single vehicle rather than a fleet, it is amenable to analysis from both a worst-case and average-case perspective, and has applications to real-world systems. Second, we consider two problems related to the scheduling of air ambulances, one in an offline setting and another in an online setting. Air ambulances are used to provide emergency medical services to residents of both British Columbia and Ontario, Canada. We consider techniques to improve the efficiency of service in these systems.

BIOGRAPHICAL SKETCH

Patrick Steele was born in Stuart, Florida in 1989, and was raised in Sandwich, Massachusetts. He received a B.S. in Applied Mathematics and Physics from the College of William and Mary in Williamsburg, Virginia in 2011, and immediately went on to study at Cornell University. After completing his Ph.D. he will begin working at Wayfair in Boston, Massachusetts.

This dissertation is dedicated to my wife, Anna, for her unending patience and support during my studies.

ACKNOWLEDGEMENTS

I would like to sincerely thank my advisors, David Shmoys and Shane Henderson, for their support and guidance during my time at Cornell. I would also like to acknowledge the support of the National Science Foundation through the grants CCF-1522054, CCF-1526067, CMMI-1537394, and CMMI-1200315.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Approximation and Competitive Algorithms

1.1.1 Approximation Algorithms

An *approximation algorithm* is an algorithm that is capable of finding a good quality solution to a problem in time bounded by a polynomial of the input size. See [23] for an overview of approximation algorithms. Approximation algorithms are useful when dealing with NP-hard problems, where exactly computing the optimal solution can be prohibitively expensive. An approximation algorithm is characterized by its *approximation ratio*, defined as follows.

Definition 1. Let a (possibly randomized) algorithm ALG be given for a minimization problem along with an optimal algorithm OPT. If

$$\mathrm{E}\left[\mathrm{ALG}(x)\right] \leq \alpha \cdot \mathrm{OPT}(x)$$

for all inputs x, then ALG is an α-approximation algorithm for the given problem.

1.1.2 Competitive Algorithms

Online algorithms are algorithms that receive information about an input sequence over time, and must make decisions as the information arrives. See [11] for an overview of competitive algorithms. For example, an online sorting algorithm would receive each number to sort over time, and must maintain a sorted list of all elements observed so far. To describe the performance of an

online algorithm we use its *competitive ratio*, which is the worst-case ratio between the online algorithm's performance and an offline optimal algorithm's performance.

Definition 2. Let a (possibly randomized) online algorithm ALG be given for a minimization problem along with an optimal offline algorithm OPT. If

$$\mathrm{E}\left[\mathrm{ALG}(x)\right] \leq c \cdot \mathrm{OPT}(x)$$

for all inputs x chosen by a given adversary, then ALG is c-competitive against the chosen adversary.

The type of adversary we consider can influence the competitive ratio of an algorithm.

1.1.3 Types of Adversaries

An *adversary* is responsible for constructing input sequences to an online algorithm. We classify adversaries by the amount of information they have at their disposal when constructing an input sequence.

Oblivious adversary. An oblivious adversary is given full knowledge of the online algorithm, but must construct the input sequence before seeing the algorithm make any decisions. Thus an oblivious adversary cannot construct an input sequence for which later inputs depend on the realization of random actions by the online algorithm.

Adaptive online adversary. An adaptive online adversary is given full knowledge of the online algorithm. Additionally, the adaptive online adversary is allowed to choose each input value after seeing the algorithm react to all previous inputs.

Adaptive offline adversary. An adaptive offline adversary is given full knowledge of the online algorithm as well as the outcome of any random decisions by the online algorithm.

Thus the oblivious, adaptive online, and adaptive offline adversaries are progressively stronger, with each adversary knowing more than the previous adversary.

1.2 Vehicle Routing Problems

The *vehicle routing problem* (VRP) encompasses a large class of problems involving the distribution of goods through a network using a collection of delivery vehicles. Formally, we have a number of *depots* from which orders for goods originate to be sent to a number of *clients*. These goods must be delivered by a fleet of vehicles moving through the network, which we will refer to as *couriers*. The couriers can have different starting locations, speeds, and carrying capacities. The objective of the VRP is to minimize the cost of serving all deliveries. A VRP may have a number of side constraints, including time windows on deliveries, capacity restrictions on the couriers, delivery route length maximums, or order release dates. See [19] for a survey of VRPs.

The VRP in the previous paragraph is also known as the *offline* VRP, as all deliveries to be made are known in advance. A natural variant of the VRP is the *online* VRP where the deliveries to be made are revealed over time. A survey of results can be found in [17].

A closely related problem is the *traveling salesman problem* (TSP). The TSP is a classic optimization problem in which the goal is to compute a minimum cost tour over n cities in a metric space [23]. If we take edge costs to the the time required to traverse this edge, the optimal TSP tour computes the minimum time

required for a courier to depart from a depot, make a number of deliveries, and return to the depot. The *online traveling salesman problem* (OLTSP), introduced in [6], is a natural variant of the TSP where the cities to be visited are revealed over time. They give an algorithm that is 2-competitive. This result is generalized to the m-courier case in [17].

The OLTSP problem can be viewed as a 1-courier instance of a VRP where all the products being delivered are fungible and the courier departs the depot with an infinite supply of goods; this ensures that the courier does not need to return to the depot before all cities have been visited.

We will consider a different objective value than the ones discussed above. So far we have considered minimizing the time required to complete all deliveries and return to the depot; this is also known as the *makespan* of the schedule. We will instead focus on minimizing the total time between the delivery of a good and its release date; this is also known as the total *latency* of the schedule. If the VRP with a makespan objective is viewed as a generalization of the TSP, then the VRP with a total latency objective can be viewed as a generalization of the *traveling repairman problem* (TRP). Like the TSP, the input to the TRP is a set of cities in a metric space, and the output is a tour over those cities. However, where the TSP finds a tour to minimize the time it takes to return to the depot, the TRP finds a tour to minimize the sum of the times it takes to visit each individual city. See citekrumke2003news for a discussion of the TRP.

1.3 Average-Case Analysis

In Section 1.1 our goal was to analyze algorithms in order to understand how they perform in the worst case. However, the world is often not so adversarial. We are also interested in understanding the *typical* performance of algorithms,

for example, by measuring the average performance under inputs drawn from a distribution, rather than selected by an adversary. We introduce some techniques in the following sections that we will later utilize.

1.3.1 Markov Decision Processes

Throughout we use the machinery of [20]. A *Markov decision process* (MDP) is a mathematical model of a discrete-time system with random events. We say $\Phi = (X, \mathcal{A}, C, P)$ is an MDP with state space X, action space $\mathcal{A}$, costs $C : X \times \mathcal{A} \to \mathbb{R}$, and action-dependent state transition probabilities $P : X \times X \times \mathcal{A} \to X$. At each discrete time the system is in some state $x \in X$, and an action $a \in \mathcal{A}$ must be chosen. A cost $C(x, a)$ is then incurred, and we transition to state $x' \in X$ with probability $P_{x,\,x'}(a)$. A *discounted* MDP has a discount factor $\alpha \in (0, 1)$ that is used to weight the costs accrued based on when they occur. If we visit states $x_0, x_1, \ldots, x_n$ and take actions $a_0, a_1, \ldots, a_n$ in each of those states the discounted cost over those $n + 1$ states is

$$\sum_{t=0}^{n} \alpha^t C(x_t, a_t).$$

The goal is to determine a sequence of actions to take to minimize the expected discounted cost over an infinite horizon.

A *policy* $\pi : X \to \mathcal{A}$ is a mapping from states to actions to take in that state. Although policies that depend on the full history of the system are permitted, Chapters 4 and 7 of [20] show that we can restrict our attention to *stationary* policies that depend on only the current state. Let $X_0, X_1, \ldots \in X$ be random variables representing the state of the system at each time $0, 1, \ldots$. Under a policy π we have that

$$\Pr\left[X_t = x' \mid X_{t-1} = x\right] = P_{x,\,x'}(\pi(x)).$$

For a given discount factor $\alpha \in (0,1)$, we define the (discounted) *value function*

$$V_{\alpha,\pi}(x) = \sum_{t=0}^{\infty} \alpha^t \, \mathrm{E}\left[C(X_t, \pi(X_t)) \mid X_0 = x\right] \tag{1.1}$$

for all $x \in X$. Thus $V_{\alpha,\pi}(x)$ represents the expected discounted cost incurred under policy π when beginning in state x. We also define the value function

$$V_\alpha(x) = \inf_\pi V_{\alpha,\pi}(x) \tag{1.2}$$

for all $x \in X$. The goal is to find a policy π that realizes $V_\alpha(x)$ for a given initial state.

A continuous-time analog of the MDP is the *continuous-time Markov decision process*. We say $\Psi = (X, \mathcal{A}, G, g, \nu, P)$ is a CTMDP with state space X, action space $\mathcal{A}$, fixed costs $G : X \times \mathcal{A} \to \mathbb{R}$, rate costs $g : X \times \mathcal{A} \to \mathbb{R}$, transition rates $\nu : X \times \mathcal{A} \to \mathbb{R}_+$, and transition probabilities $P : X \times X \times \mathcal{A} \to X$. Events occur in continuous time. If at some time we are in state $x \in X$ and action $a \in \mathcal{A}$ is chosen we incur costs $G(x,a)$ as well as rate costs $g(x,a)$ until the next transition time. The next transition time is exponentially distributed with rate $\nu(x,a)$. We transition to state x' with probability $P_{x,\,x'}(a)$. Our goal is to find a policy that minimizes the long-run average cost incurred by the system.

1.3.2 Sample Average Approximation

Sample average approximation (SAA) is a technique used to solve simulation optimization problems. In such problems the exact objective value of the problem is either unknown or complicated, but can be estimated by a simulation. In particular, we assume that the true objective function can be approximated by

$$f(x) = \mathrm{E}\left[f(x,\xi)\right],$$

where the distribution of the random variable ξ does not depend on x. For example, ξ might represent the arrival times of customers in a service queue, while x represents the choice of labor allocated to the queue. We can then approximate f via

$$f_n(x) = \frac{1}{n} \sum_{i \in [n]} f(x, \xi_i)$$

where $\xi_1, \ldots, \xi_n$ are all drawn from the same distribution. Given this fixed sample $\xi_1, \ldots, \xi_n$, we can then use f_n as our approximate objective function. See [18] for an overview of SAA.

1.4 General Notation

Table 1.1 contains notation used throughout this dissertation.

Symbol	Definition
$[n]$	The set $\{1, 2, \ldots, n\}$ for any positive integer n
$(x)^+$	The value $\max\{0, x\}$ for any real number x
$\mathrm{E}_X[f]$	The expectation of the expression f with respect to a random variable X; X may be omitted if it is clear from context.
$\Pr[x]$	The probability of the random event x happening
$\mathrm{OPT}(x)$	The optimal cost of an input x, where the problem is context-dependent.
$\mathrm{ALG}(x)$	The cost of an input x under some algorithm, where the problem and algorithm are context-dependent.

Table 1.1: A summary of the notation used throughout this dissertation.

1.5 Source Code

The source code for projects discussed in this dissertation are available online at the following locations:

- `github.com/prsteele/mdp` contains the Haskell code used to solve the MDPs described in Chapter 2.
- `github.coecis.cornell.edu/ornge/ornge` contains the code used to solve the base selection problem in Chapter 3.
- `github.coecis.cornell.edu/BCEHS/BCEHS-Simulation` contains the code used to simulate operations at BCEHS as in Chapter 4.

CHAPTER 2

AGGREGATING COURIER DELIVERIES

2.1 Motivation

We consider the problem of scheduling deliveries of goods from a central depot with an uncapacitated courier under online arrivals. This is a problem facing many companies offering on-demand delivery services. Uber Rush and Amazon Prime Now offer on-demand delivery services for online purchases, with deliveries made by local couriers within hours of purchase [1, 5]. The couriers delivering these products typically operate in urban areas where orders can originate from clustered retailers, offering the possibility for multiple orders to be grouped and delivered together. Thus there is a tension between immediately dispatching a courier when a delivery arrives to minimize the latency of that single order and waiting some amount of time to minimize the total latency over several nearby orders.

To explore this tension we consider the problem where all deliveries arrive at a central depot and must be delivered by a single uncapacitated courier. The deliveries will lie in some metric space. The courier can pick up deliveries at the depot, move through the space to make each delivery, and then return to the depot to serve future requests. The objective is to minimize the total time between arriving in the system and being delivered across all deliveries.

2.1.1 Notation

We define $\mathcal{S}$ as a finite discrete metric space with distance function $\|\cdot\|$. The depot is located at $s^* \in \mathcal{S}$. A *delivery request* (t, s) is a request for a delivery to location $s \in \mathcal{S}$ arriving at time $t \in \mathbb{R}_+$. A *request sequence* of length n is a list

of n delivery requests ordered by increasing arrival time. We define $\sigma^{(n)}$ as a random variable over request sequences of length n with distribution function $\mu^{(n)}$; distribution functions are defined within the context of each section. For a request sequence $((t_1, s_1), (t_2, s_2), \ldots, (t_n, s_n))$ and a departure schedule that delivers delivery request (t_i, s_i) at time t'_i, the *latency* of the request is $w_i = t'_i - t_i$. The objective is to minimize $\sum_{i=1}^{n} w_i$.

Finally, we define OPT(x) as the optimal cost of serving a request sequence x with an offline algorithm, while ALG(x) is the cost of serving the request sequence with a given context-specific online algorithm.

2.2 Problem Statement

The problem is to design an online algorithm that chooses departure times to minimize the total latency of all delivery requests in a request sequence chosen by an adversary. We require that when the algorithm sends the courier out for a delivery that all waiting delivery requests are served, and we do not allow the server to return to the depot before all deliveries are made.

We will consider this problem from two perspectives. In Section 2.3 we consider the case where both arrival times and locations are controlled by an adversary. We present a $(3\beta\Delta/2\delta - 1)$-competitive algorithm, along with a lower bound of $(1 + 0.271\Delta/\delta)$ for the competitive ratio of any online algorithm; here, β is the approximation ratio of the TSP, Δ is the optimal TSP tour length over all clients, and δ is the minimum distance between the depot and any client, noting that all clients are a positive distance from the depot. In Section 2.4 we consider the case where arrival times and locations occur according to a Poisson process. We derive structural results on the optimal policies for minimizing the long-run average latency; in particular, we show that optimal policies exhibit an intuitive

threshold structure. Finally in Section 2.5 we explore the performance of our randomized competitive algorithm in the Poisson arrivals setting.

In both sections we will rely on the notion of an *a priori* tour of the clients. An *a priori* tour is a TSP tour over all the locations that is used to dictate the order in which clients are visited. In particular, when only a subset of clients needs to be visited following the *a priori* tour causes us to visit the clients in the same order as in the full tour while eliminating unnecessary legs. While there is extensive literature on computing *a priori* TSP tours [16, 9, 21], we will not require any particular measure of optimality with respect to the *a priori* TSP problem. In Section 2.3 this assumption is used to construct an algorithm, but the lower bounds derived to not depend on it. In Section 2.4 this assumption is used more directly, and allows us to tractably model the problem.

2.3 Adversarial Setting

We consider the case where the request sequences are chosen by an adversary. We consider both the *oblivious adversary* and the *adaptive offline adversary*.

Our analysis will depend on a polynomial time traveling salesman approximation algorithm TSP_β with approximation guarantee β, where by convention we always include the depot s^* as the starting location of the tour. Define

$$\Delta = \text{TSP}_1(\mathcal{S}) \tag{2.1}$$

along with

$$\delta = \min \{\|s - s^*\| \mid s \in \mathcal{S} \setminus \{s^*\}\}. \tag{2.2}$$

Lemma 1. *Let a request sequence $S = \{(t_1, s_1), (t_2, s_2), \ldots, (t_n, s_n)\}$ be given. Then*

$$OPT(S) \geq \sum_{i=1}^{n} \|s_i - s^*\| \geq n\delta.$$

Algorithm 1 RAND-SINGLE

Draw $\alpha \leftarrow \text{Uniform}(0, \beta\Delta)$
for $i \leftarrow 0, 1, \ldots$ **do**
  Let S be the set of delivery requests at the depot at time $i\beta\Delta + \alpha$
  Depart at time $i\beta\Delta + \alpha$ with all requests in S, short-cutting $\text{TSP}_\beta(S)$
end for

Proof. Consider delivery request (t_i, s_i) and the set S of delivery requests that it is sent with on the courier.

From (2.2), we have that $\|s_i - s^*\| \geq \delta$ for all $i \in [n]$. When s_i is delivered along any tour by the triangle inequality the distance traveled before reaching s_i is at least $\|s_i - s^*\|$. Thus the optimal offline algorithm incurs a cost of at δ per delivery request, and so pays at least $n\delta$ in total. □

Theorem 1. *Algorithm 1 is* $(3\beta\Delta/2\delta - 1)$*-competitive against an oblivious adversary.*

Proof. We first show that Algorithm 1 produces a feasible schedule that serves all delivery requests. Consider a departure at time $k\beta\Delta + \alpha$ for any $k \geq 0$ and any realization of α, and let S be the set of delivery requests at the depot at that time. The algorithm departs at $k\beta\Delta + \alpha$ and embarks on a tour according to $\text{TSP}_\beta(S)$. By construction we have that $\beta\Delta \geq \text{TSP}_\beta(S)$, and so the courier will return to the depot before the next scheduled departure at $(k+1)\beta\Delta + \alpha$, after having served all requests in S.

We now show that the competitive ratio is as claimed. We proceed by bounding the cost of any single delivery request. Let a delivery request (t_i, s_i) be given, and define $k = \lfloor t/\Delta \rfloor$. Our algorithm will depart at times $k\Delta+\alpha$ and $(k+1)\Delta+\alpha$. Thus (t, s) will be sent for delivery at time $k\Delta + \alpha$ when $t \leq k\Delta + \alpha$ and at time $(k+1)\Delta + \alpha$ when $t > (k+1)\Delta + \alpha$. If $t \leq k\beta\Delta + \alpha$ the request will wait $k\beta\Delta + \alpha - t$ before departing, and otherwise will wait $(k+1)\beta\Delta + \alpha - t$ before departing.

When we depart with this request it will be delivered along a tour of all requests being delivered at that time. For any set of requests S being delivered we have that $\text{TSP}_\beta(S) \leq \beta\Delta$. Since the tour must begin and end at s^*, we will visit s_i after traveling at most $\beta\Delta - \|s_i - s^*\|$. The latency of this request consists of the waiting time before the courier departs and the delivery time after it departs. This gives us

$$\begin{aligned} w_i &= \beta\Delta - \|s_i - s^*\| + (k\beta\Delta + \alpha - t)\,\mathbb{1}_{\{t \leq k\beta\Delta + \alpha\}} + ((k+1)\beta\Delta + \alpha - t)\,\mathbb{1}_{\{t > k\beta\Delta + \alpha\}} \\ &= \beta\Delta - \|s_i - s^*\| + k\beta\Delta + \alpha - t + \beta\Delta\,\mathbb{1}_{\{\alpha < t - k\beta\Delta\}}\,. \end{aligned}$$

Since α is uniformly distributed over $[0, \Delta]$, we can compute

$$\begin{aligned} \mathrm{E}\left[w_i\right] &= \beta\Delta - \|s_i - s^*\| + k\beta\Delta + \frac{\beta\Delta}{2} - t + \beta\Delta\,\Pr\left[\alpha < t - k\beta\Delta\right] \\ &= \beta\Delta - \|s_i - s^*\| + k\beta\Delta + \frac{\beta\Delta}{2} - t + t - k\beta\Delta \\ &= \frac{3}{2}\beta\Delta - \|s_i - s^*\|\,. \end{aligned}$$

Thus the expected cost of any request sequence $S = \{(t_1, s_1), (t_2, s_2), \ldots, (t_n, s_n)\}$ is

$$\mathrm{E}\left[\text{ALG}(S)\right] = \mathrm{E}\left[\sum_{i=1}^{n} w_i\right] = \sum_{i=1}^{n} \mathrm{E}\left[w_i\right] = \frac{3}{2}n\beta\Delta - \sum_{i=1}^{n} \|s_i - s^*\|\,.$$

by the linearity of expectations. Finally, by Lemma 1 we have that

$$\begin{aligned} \text{ALG}(S) &\leq \frac{3}{2}n\beta\Delta - \sum_{i=1}^{n} \|s_i - s^*\| \\ &= \left(\frac{3\beta\Delta}{2\delta} - 1\right)\text{OPT}(S), \end{aligned}$$

as required. □

It is worth noting where this proof depends on the assumption of an oblivious adversary. In particular, we use this assumption when we take an expectation over α. An adaptive online adversary can learn α by sending just a single

delivery request at time 0 and observing the algorithm's response. An adaptive offline adversary simply knows the realization of α in advance. This leads to the following result.

Theorem 2. *Algorithm 1 is* $(2\beta\Delta/\delta - 1)$*-competitive against an adaptive offline adversary.*

Proof. From the definition of Algorithm 1 there is a departure within $\beta\Delta$ of the arrival of any delivery request. Once on the courier, a delivery request waits at most an additional $\beta\Delta - \|s_i - s^*\|$ time before being delivered, as in the proof of Theorem 1. Thus the latency of the delivery request (t_i, s_i) is at most

$$w_i = \beta\Delta + \beta\Delta - \|s_i - s^*\|,$$

and so for a request sequence $S = \{(t_1, s_1), (t_2, s_2), \ldots, (t_n, s_n)\}$ we have

$$\text{ALG}(S) \leq \sum_{i=1}^{n} (2\beta\Delta - \|s_i - s^*\|) \leq \left(2\frac{\beta\Delta}{\delta} - 1\right) \text{OPT}(S),$$

as required. □

2.3.1 A Lower Bound on the Competitive Ratio

We now provide a lower bound on the competitive ratio of any online algorithm by utilizing Yao's Lemma, shown in Theorem 7 of Section A. We proceed as follows. We first describe an input distribution. We then provide an upper bound on the expected cost of the optimal offline algorithm for this input distribution. Next we show that the optimal deterministic algorithm for any given input will only choose to depart at certain times, and then we will provide a lower bound on the cost of such an algorithm. Finally we will apply these results to Yao's Lemma.

We begin by constructing an input distribution $\mu^{(N)}$ over N-length request sequences. To construct $\mu^{(N)}$, let $\bar{S} = \mathcal{S} \setminus \{s^*\}$ and let

$$\underline{s} = \arg\min \{\|s - s^*\| \mid s \in \mathcal{S} \setminus \{s^*\}\};$$

that is, $\bar{S}$ represents a worst-case TSP instance in $\mathcal{S}$ while $\underline{s}$ is a location as close to the depot as possible. Define as well

$$m_N = \max\left\{i \in \mathbb{Z}_+ \mid (i+1)^2 \leq N\right\},$$

and so $(m_N + 1)^2 \leq N < (m_N + 2)^2$. Let X be a random variable with mass function $f(i) = 1/m_N$, $i \in [m_N]$, along with *i.i.d.* exponential random variables Y_i with rate parameter λ for each $i \in [X + 1]$. Define

$$\tau_i = \begin{cases} 0, & i = 1, \\ \tau_{i-1} + Y_i + \Delta + 2\delta, & 2 \leq i \leq X, \\ \tau_X + Y_{X+1}, & i = X + 1. \end{cases}$$

for $i \in [X + 1]$.

Our input distribution $\mu^{(N)}$ consists of $X + 1$ *bunches* of arrivals, where a bunch is a collection of delivery requests arriving at the same time. We only consider N such that $m_N > \left|\bar{S}\right|$. The bunches arrive at *bunch times* $\tau_1, \tau_2, \ldots, \tau_{X+1}$. For $i \in [X]$, the bunch arriving at τ_i consists of one delivery request going to each location in $\bar{S}$, along with $m_N - \left|\bar{S}\right|$ delivery requests going to $\underline{s}$, for a total of m_N delivery requests. The bunch arriving at time τ_{X+1} consists of $N - m_N X$ delivery requests all going to $\underline{s}$.

Lemma 2. *Let N be given. Assuming $X \geq i$, the bunch arriving at time τ_i can be delivered so that the total latency of requests in the bunch is at most*

$$m_N \delta + \left|\bar{S}\right| \Delta.$$

This delivery strategy requires at most $\Delta + 2\delta$ time to complete.

Proof. To prove the upper bound on the cost we provide a tour that achieves the desired cost. Suppose that $\text{TSP}_1(\bar{S})$ gives a tour $s^*, s_1, \ldots, s_k, s^*$, which by construction has length at most Δ. Consider the path $s^*, \underline{s}, s_1, \ldots, s_k$, which may visit $\underline{s}$ twice. From the triangle inequality we have that

$$\|\underline{s} - s_1\| \leq \|\underline{s} - s^*\| + \|s^* - s_1\|,$$

and so this path is no longer than the path $s^*, \underline{s}, s^*, s_1, \ldots, s_k$. Since $\|s^* - \underline{s}\| \leq \delta$, this path has length at most $2\delta + \Delta$. Note that since s_k is at least δ from s^*, all deliveries to locations in $\bar{S}$ travel no more than $\delta + \Delta$. The cost of serving the requests to $\underline{s}$ is $(m_N - |S|)\delta$, while the cost of serving the requests to locations in $\bar{S}$ is at most $\left|\bar{S}\right| (\Delta + \delta)$. Thus the total latency of the requests in the bunch is at most

$$(m_N - |S|)\delta + \left|\bar{S}\right| (\Delta + \delta) = m_N\delta + \left|\bar{S}\right| \Delta,$$

as required. □

Lemma 3. *Let N be given. Assuming $X \geq i$, the total latency of requests in the bunch arriving at time τ_i is at least $m_N\delta$, and the delivery takes at least Δ time.*

Proof. By construction $\|s - s^*\| \geq \delta$ for all $s \in \bar{S}$, and so each of the m_N delivery requests in the bunch incurs a cost of at least δ. Finally, by assumption we have that $\text{TSP}_1(\bar{S}) = \Delta$, and so the courier can return from delivery no sooner than Δ after departing. □

We now provide an upper bound on the expected cost of the optimal offline algorithm for this input distribution.

Lemma 4. *For any N such that $m_N > \left|\bar{S}\right|$,*

$$\text{E}_{\mu^{(N)}} \left[OPT(\sigma^{(N)})\right] \leq \delta N + \left(1 + \frac{1}{\lambda}\right) o(N).$$

Proof. We provide an offline algorithm that achieves the desired cost; the optimal offline algorithm must do at least as well. For a given realization of X, we choose to depart at times $\tau_1, \ldots, \tau_{X-1}$, and then at time τ_{X+1}. Each time we depart we follow the tour described in Lemma 2. Note that since each departure returns us to the depot in at most $\Delta + 2\delta$ time that this departure schedule is feasible. This incurs a cost of at most $m_N\delta + \left|\bar{S}\right| \Delta$ for the bunches at times $\tau_1, \ldots, \tau_{X-1}$. Delivery requests in the bunch at time τ_X will wait an additional Y_{X+1} time before being delivered alongside the requests in the final bunch. When we make the final delivery we follow the same path as in previous bunches, yielding a total cost of $m_N Y_{X+1} + m_N\delta + \left|\bar{S}\right| (\Delta + \delta)$ for requests in the bunch at time τ_X and a total cost of $(N - m_N X)\delta$ for requests in the final bunch. Thus we have that

$$\begin{aligned}\text{OPT}(\sigma^{(N)}) &\leq \left(m_N\delta + \left|\bar{S}\right| \Delta\right) X + m_N Y_{X+1} + (N - m_N X)\delta \\ &= N\delta + m_N Y_{X+1} + X \left|\bar{S}\right| \Delta.\end{aligned}$$

Taking expectations, we find

$$\begin{aligned}\mathrm{E}_{\mu^{(N)}}\left[\text{OPT}(\sigma^{(N)})\right] &\leq N\delta + \frac{m_N}{\lambda} + \frac{m_N + 1}{2} \left|\bar{S}\right| \Delta \\ &\leq N\delta + \left(1 + \frac{1}{\lambda}\right) o(N),\end{aligned}$$

since $\lim_{N\to\infty} m_N/N \to 0$ by construction. □

We must now provide lower bounds on the cost of the best deterministic algorithm for any input sequence $\sigma^{(N)}$. We first show that we can restrict our attention to algorithms that depart only at times that are a subset of bunch times $\{\tau_1, \ldots, \tau_{X+1}\}$.

Lemma 5. *For any request sequence $\sigma^{(N)}$ drawn from $\mu^{(N)}$, let any algorithm be given that chooses to depart at some time not in the set $\{\tau_1, \ldots, \tau_{X+1}\}$. Then this algorithm*

performs no better than an algorithm that chooses only to depart at times in the set $\{\tau_1, \ldots, \tau_{X+1}\}$.

Proof. Let ALG be a deterministic algorithm that chooses to depart at some time not in $\{\tau_1, \ldots, \tau_{X+1}\}$. Since ALG is deterministic, for it to depart at some time not in the set $\{\tau_1, \ldots, \tau_{X+1}\}$ it must choose to depart a fixed time $\tau > 0$ after some bunch time τ_j, unless perhaps $\tau_j + \tau \geq \tau_{j+1}$. Let τ_j be the first time the algorithm chooses to wait $\tau > 0$ before departing. Note that if $j = m_N + 1$, then the algorithm is waiting to depart after the final arrival and so is trivially worse than an otherwise equivalent algorithm that chooses $\tau = 0$. For $j < m_N + 1$ there exist algorithms that are otherwise equivalent to this one, except that they either choose $\tau = 0$ or $\tau = \infty$. Namely, let ALG_1 be the algorithm that chooses $\tau = 0$ and let ALG_2 be the algorithm that chooses $\tau = \infty$.

Suppose that $\tau_j + \tau < \tau_{j+1}$, and so ALG departs before the next bunch time. In this case we have that $\text{ALG}(\sigma^{(N)}) \geq \text{ALG}_1(\sigma^{(N)}) + m_N\tau$, since the m_N delivery requests that arrived at τ_j wait an additional τ before departure relative to what they wait under ALG_1. Alternatively, if $\tau_j + \tau \leq \tau_{j+1}$ then ALG behaves exactly like ALG_2, and so $\text{ALG}(\sigma^{(N)}) \geq \text{ALG}_2(\sigma^{(N)})$. Let $p = \Pr[\tau_j + \tau < \tau_{j+1}]$. Then

$$\text{ALG}(\sigma^{(N)}) \geq p\text{ALG}_1(\sigma^{(N)}) + pm_N\tau + (1-p)\text{ALG}_2(\sigma^{(N)}).$$

Thus either $\text{ALG}_1(\sigma^{(N)}) \leq \text{ALG}(\sigma^{(N)})$ or $\text{ALG}_2 \leq \text{ALG}(\sigma^{(N)})$, as required. □

We now provide a lower bound on the cost of any deterministic algorithm for a particular choice of λ. Define the function $f(x) = xe^x$, and let $\text{LambertW}(x)$ be the inverse of f [22].

Lemma 6. *For*

$$\lambda = \frac{2 + \text{LambertW}\left(-e^{-2}\right)}{\Delta}$$

and $\sqrt{N} > \left|\bar{S}\right|$,

$$\mathrm{E}\left[ALG(\sigma^N)\right] \geq m_N(m_N+1)\left(\delta + \frac{1}{2}\cdot\frac{\Delta}{\text{LambertW}(-e^{-2})+2}\right).$$

Proof. Suppose we are at time τ_k, and so the algorithm has just observed the kth bunch. We compute (lower bounds on) the expected cost of choosing to depart immediately at τ_k and the cost of choosing to remain until at least τ_{k+1}. Note that we assign the cost of delivering the final bunch to the bunch at time τ_X.

We first consider the cost of departing immediately. By Lemma 3 the algorithm must pay at least $m_N\delta$ to deliver the requests in bunch at time τ_k, and the algorithm takes at least Δ time to return to the depot. Additionally, if $X = k$ there is the chance that the bunch at τ_{X+1} must wait until we return (after no less than Δ time) before being delivered, where again each delivery takes at least δ time. Thus the cost of departing is at least

$$C_\Delta = m_N\delta + \mathbb{1}_{\left\{X=k \,\middle|\, X>k-1\right\}}\left(N - m_N k\right)\left(\delta + \left(\Delta - Y_{X+1}\right)^+\right).$$

Taking the expectation over X, we find

$$\begin{aligned}\mathrm{E}\left[C_{\text{Depart}}\right] &= m_N\delta + \frac{N-m_Nk}{m_N-k+1}\left(\delta + \int_0^\Delta (\Delta - y)\,\lambda e^{-\lambda y}\,dy\right)\\ &= m_N\delta + \frac{N-m_Nk}{m_N-k+1}\left(\delta + \frac{e^{-\lambda\Delta}-1}{\lambda} + \Delta\right).\end{aligned}$$

Since $N \geq (m_N+1)^2 \geq m_N(m_N+1)$, we have that

$$\begin{aligned}\mathrm{E}\left[C_{\text{Depart}}\right] &\geq m_N\delta + m_N\left(\delta + \frac{e^{-\lambda\Delta}-1}{\lambda} + \Delta\right)\\ &= 2m_N\delta + m_N\left(\frac{e^{-\lambda\Delta}-1}{\lambda} + \Delta\right).\end{aligned}$$

We now consider the cost of choosing to remain at the depot until at least time τ_{k+1}. Since $\tau_{k+1} - \tau_k = Y_{k+1} + \Delta + 2 \geq Y_{k+1}$, the total cost of delivering these requests is at least $m_N Y_{k+1} + m_N\delta$. If $X = k$ we must also pay for the delivery

requests in the last bunch. This gives us that the cost of remaining is at least

$$C_{\text{Remain}} = m_N Y_{k+1} + m_N\delta + \mathbb{1}_{\left\{X=k \,\middle|\, X>k-1\right\}}(N - m_N k)\delta.$$

Taking expectations, we find

$$\begin{aligned}\mathrm{E}\left[C_{\text{Remain}}\right] &= \frac{m_N}{\lambda} + m_N\delta + \frac{N - m_N k}{m_N - k + 1}\delta \\ &\geq \frac{m_N}{\lambda} + 2m_N\delta.\end{aligned}$$

To bound the cost of the algorithm, it will be sufficient to choose λ such that $\min\left\{\mathrm{E}\left[C_{\text{Depart}}\right], \mathrm{E}\left[C_{\text{Remain}}\right]\right\} \geq \phi$ for some positive constant ϕ; if this holds, then we incur at least ϕ at each bunch time. Consider

$$\lambda = \frac{2 + \text{LambertW}\left(-e^{-2}\right)}{\Delta}.$$

Note that this implies that

$$\begin{aligned}e^{-\lambda\Delta} &= e^{-2-\text{LambertW}(-e^{-2})} \\ &= e^{-2}e^{-\text{LambertW}(-e^{-2})} \\ &= e^{-2}\frac{\text{LambertW}(-e^{-2})}{-e^{-2}} \\ &= -\text{LambertW}(-e^{-2}).\end{aligned}$$

Then

$$\begin{aligned}\mathrm{E}\left[C_{\text{Depart}}\right] &\geq 2m_N\delta + m_N\left(\frac{e^{-\lambda\Delta} - 1}{\lambda} + \Delta\right) \\ &\geq 2m_N\delta + m_N\Delta\left(1 - \frac{\text{LambertW}(-e^{-2}) + 1}{\text{LambertW}(-e^{-2}) + 2}\right) \\ &\geq 2m_N\delta + \frac{m_N\Delta}{\text{LambertW}(-e^{-2}) + 2},\end{aligned}$$

while

$$\begin{aligned}\mathrm{E}\left[C_{\text{Remain}}\right] &\geq 2m_N\delta + \frac{m_N}{\lambda} \\ &\geq 2m_N\delta + \frac{m_N\Delta}{\text{LambertW}(-e^{-2}) + 2}.\end{aligned}$$

From Lemma 5 we know that we can restrict our attention to algorithms which only choose to depart at bunch times. Any such algorithm incurs a cost of at least

$$2m_N\delta + \frac{m_N\Delta}{\text{LambertW}(-e^{-2})+2}$$

at each of the first X bunch times. This gives us

$$\begin{aligned}
\mathrm{E}_{\mu^{(N)}}\left[\text{ALG}(\sigma^{(N)})\right] &\geq \mathrm{E}\left[\sum_{i=1}^{X}\left(2m_N\delta + \frac{m_N\Delta}{\text{LambertW}(-e^{-2})+2}\right)\right] \\
&= \mathrm{E}\left[X\right]\left(2m_N\delta + \frac{m_N\Delta}{\text{LambertW}(-e^{-2})+2}\right) \\
&= \frac{m_N+1}{2}\left(2m_N\delta + \frac{m_N\Delta}{\text{LambertW}(-e^{-2})+2}\right) \\
&= m_N(m_N+1)\left(\delta + \frac{1}{2}\cdot\frac{\Delta}{\text{LambertW}(-e^{-2})+2}\right).
\end{aligned}$$

□

We are now prepared to provide a lower bound on the competitive ratio of any online algorithm via Yao's principle.

Theorem 3. *There does not exist an online algorithm with competitive ratio less than* $1 + 0.271\Delta/\delta$.

Proof. We apply Yao's principle to our input distribution $\mu^{(N)}$ with

$$\lambda = \frac{2 + \text{LambertW}\left(-e^{-2}\right)}{\Delta}.$$

From Lemma 4 we have that

$$\begin{aligned}
\mathrm{E}_{\mu^{(N)}}\left[\text{OPT}(\sigma^{(N)})\right] &\leq \delta N + \left(1+\frac{1}{\lambda}\right)o(N) \\
&\leq \delta N + o(N)
\end{aligned}$$

since λ does not depend on N. Likewise, from Lemma 6 we have that

$$\begin{aligned}
\inf_i \mathrm{E}_{\mu^{(N)}}\left[\text{ALG}_{N,i}\right] &\geq m_N(m_N+1)\left(\delta + \frac{1}{2}\cdot\frac{\Delta}{\text{LambertW}(-e^{-2})+2}\right) \\
&\geq m_N^2\left(\delta + \frac{1}{2}\cdot\frac{\Delta}{\text{LambertW}(-e^{-2})+2}\right) + o(N),
\end{aligned}$$

where $\{\text{ALG}_{N,i} \mid i \in \mathbb{Z}_+\}$ is the set of all deterministic algorithms for request sequences of length N. Thus

$$\begin{aligned}
\lim_{N\to\infty} \frac{\inf_i \mathrm{E}_{\mu^{(N)}}\left[\text{ALG}_{N,i}\right]}{\mathrm{E}_{\mu^{(N)}}\left[\text{OPT}(\sigma^{(N)})\right]} &\geq \lim_{N\to\infty} \frac{m_N^2\left(\delta + \frac{1}{2}\cdot\frac{\Delta}{\text{LambertW}(-e^{-2})+2}\right) + o(N)}{\delta N + \left(1+\frac{1}{\lambda}\right)o(N)} \\
&= 1 + \frac{\frac{1}{2}\cdot\frac{\Delta}{\text{LambertW}(-e^{-2})+2}}{\delta} \\
&= 1 + \frac{\Delta}{2\delta\left(\text{LambertW}(-e^{-2})+2\right)} \\
&> 1 + 0.271\frac{\Delta}{\delta},
\end{aligned}$$

as required. □

2.4 Average-case Setting

We now consider the case where delivery requests occur according to a Poisson process. We consider request sequences of unbounded length, and seek to minimize the long-run average cost of serving such request sequences. The times of delivery requests will be distributed according to a Poisson process with rate λ, and the location of the requests will be distributed *i.i.d.* according to some probability mass function $f_{\mathcal{S}} : \mathcal{S} \to \mathbb{R}_+$; this ensures that each request sequence $(t_1, s_1), (t_2, s_2), \ldots$ is distributed according to a marked Poisson process. We assume that $f_{\mathcal{S}}(s) > 0$ for all $s \in \mathcal{S} \setminus \{s^*\}$.

In Section 2.3 we considered online algorithms that made use of approximation algorithms to produce an *a priori* TSP tour. Here we only rely on having any *a priori* TSP tour Π over all locations in $\mathcal{S}$. We make no assumptions about the quality of this tour; rather, for any such tour we derive the structure of the optimal policy for serving delivery requests using that tour. For the remainder of this section we will assume that some Π is given and fixed, and without loss of generality we assume that Π visits $s_1, s_2, \ldots, s_n, s^*$ in that order.

We will further relax our assumption of travel times. In particular we assume that any path through $\mathcal{S}$ of length d will take d time to travel in expectation, with the actual time required to traverse the path being exponentially distributed with mean d. This will make the problem amenable to analysis as a CTMDP.

2.4.1 The Problem as a CTMDP and MDP

We express this problem as CTMDP. We define the state space $X = \mathbb{Z}_+^{|\mathcal{S}|}$. We index elements $x \in X$ by the location each coordinate represents, and so x_{s_i} represents the number of delivery requests to $s_i \in \mathcal{S}$ waiting at the depot. We define e^{s_i} as a vector in X such $e^{s_i}_{s_j} = \mathbb{1}_{\{i=j\}}$. For any $x \in X$, define

$$\mathcal{L}(x) = \{s \in \mathcal{S} \mid x_s > 0,\ s \neq s^*\} \tag{2.3}$$

as the set of delivery locations for requests waiting at the depot; it will be convenient to also define $\mathcal{L}(x)$ as $\sum_{s \in \mathcal{L}(x)} e^s$ depending on context.

We define the action space $\mathcal{A}$ as

$$\mathcal{A} = \{\text{Remain}\} \cup \left\{\text{Depart}_{x'} \mid x' \in X \setminus \{\mathbf{0}\},\ \mathcal{L}(x) \subseteq \mathcal{L}(x')\right\}. \tag{2.4}$$

The decision epochs will correspond to delivery request arrivals and the courier returning to the depot.

Let $\Pi(x)$ be the tour that begins at s^*, visits all locations in $\mathcal{L}(x)$, and then returns to s^*, visiting each location in the same order as in Π. Choosing the action $\text{Depart}_{x'}$ in state x means that the courier will depart from the depot to deliver all waiting requests, following the tour $\Pi(x')$. At this time we will pay the costs associated with each delivery request being sent along the tour $\Pi(x')$, as well as paying the holding fees associated with any new delivery requests that arrive while we are away from the depot.

When we depart on a tour $\Pi(x')$ in state x, we will be gone for a random time Λ that is exponentially distributed with mean $\Pi(x')_{s^*}$. Since the arrival sequence is a Poisson process with rate λ, given the value of Λ the number of arrivals I is a Poisson random variable with parameter $\lambda\Lambda$. Let $\xi_1, \xi_2, \ldots, \xi_I$ be random variables describing the arrival times of delivery requests while we are away from the depot, measured from the departure time, and let $S_1, S_2, \ldots, S_I$ be the *i.i.d.* random variables describing the locations they are sent to. Conditional on Λ these I arrivals will arrive at times after we depart that are uniformly distributed over $[0, \Lambda]$. We charge each of these arrivals the mean waiting time they accrue, which will be $\Lambda/2$. Thus the total expected waiting time accrued is

$$\begin{aligned}
\mathrm{E}\left[\sum_{i=1}^{I}(\Lambda - \xi_i)\right] &= \mathrm{E}\left[\mathrm{E}\left[\sum_{i=1}^{I}(\Lambda - \xi_i)\middle|\,\Lambda\right]\right] \\
&= \mathrm{E}\left[\mathrm{E}\left[\frac{\Lambda}{2}I\middle|\,\Lambda\right]\right] \\
&= \frac{\lambda}{2}\,\mathrm{E}\left[\Lambda^2\right] \\
&= \frac{\lambda}{2}\cdot 2\Pi(x')_{s^*}^2 \\
&= \lambda\Pi(x')_{s^*}^2.
\end{aligned}$$

The cost of delivering the requests in x is simply $\sum_{s\in\mathcal{S}}\Pi(x')_s x_s$. Then we have transition probabilities P, fixed costs G, rate costs g, and transition rates ν of

$$P_{x,\,x''}(\text{Depart}_{x'}) = \Pr\left[x'' = \sum_{i\in[I]} e^{S_i}\right], \tag{2.5}$$

$$G(x, \text{Depart}_{x'}) = \lambda\Pi(x')_{s^*}^2 + \sum_{s\in\mathcal{S}}\Pi(x')_s x_s \tag{2.6}$$

$$g(x, \text{Depart}_{x'}) = 0, \tag{2.7}$$

$$\nu(x, \text{Depart}_{x'}) = \frac{1}{\Pi(x')_{s^*}}. \tag{2.8}$$

The transition probabilities P represent that we transition to the random state $\sum_{i\in[I]} e^{S_i}$. The fixed costs G represent the cost of delivering the request currently

waiting at the depot, along with the waiting time accrued by new delivery requests that arrive while we are gone. There are no rate costs, since the waiting time accrued by requests on the courier have been accounted for in the fixed costs. Finally, the transition rate ν represents that the delivery takes $\Pi(x')_{s^*}$ in expectation to complete.

Taking the Remain action in state x means that we will not depart from the depot until at least the time of the next delivery request arrival. During this time we pay the holding fees for each delivery request waiting at the depot. The next delivery request will occur in an exponentially distributed time with rate λ and will be sent to a destination in $\mathcal{S}$ according to distribution function $F_{\mathcal{S}}$. While we are waiting for this arrival all requests at the depot continue to accrue waiting time. We have no fixed costs. This gives us transition probabilities P, fixed costs G, rate costs g, and transition rates ν of

$$P_{x,\,x''}(\text{Remain}) = \begin{cases} f_{\mathcal{S}}(s_i), & x'' = x + e^{s_i} \\ 0, & \text{otherwise,} \end{cases} \tag{2.9}$$

$$G(x, \text{Remain}) = 0 \tag{2.10}$$

$$g(x, \text{Remain}) = \sum_{s \in \mathcal{S}} x_s, \tag{2.11}$$

$$\nu(x, \text{Remain}) = \lambda. \tag{2.12}$$

The transition probabilities P represent that the next state has exactly one additional delivery request. The rate costs g represent the additional waiting time accrued by the waiting delivery requests. The transition rate ν represents that the next delivery request arrives at an exponential rate with rate parameter λ.

Finally, we can use standard uniformization techniques from [20] to convert the CTMDP $\Psi = (\mathcal{S}, \mathcal{A}, g, G, \nu, P)$ to a discrete time MDP. The idea behind uniformization is to create a discrete-time MDP where each transition represents a

time step

$$\begin{aligned}\tau &= \inf_{x\in X,\ a\in\mathcal{A}} \nu(x,a)^{-1} \\ &= \min\left\{\frac{1}{\lambda},\ \Pi(\mathbf{1})_{s^*}\right\}\end{aligned} \tag{2.13}$$

the fastest mean transition time in Ψ, where $\mathbf{1} \in X$ is the ones vector. We compensate for slower transitions by increasing the probability of self transitions in those states. Applying these techniques gives us a cost function C and transition probabilities P^* defined as

$$C(x,a) = G(x,a)\nu(x,a) + g(x,a) \tag{2.14}$$

$$P^*_{x,\,x'}(a) = \begin{cases} \tau\nu(x,a)P_{x,\,x'}(a), & x \neq x' \\ 1 - \tau\nu(x,a), & x = x'. \end{cases} \tag{2.15}$$

It is worth expanding the definition of C, which gives us

$$\begin{aligned} C(x, \text{Remain}) &= \sum_{s\in\mathcal{S}} x_s \\ C\left(x, \text{Depart}_{x'}\right) &= \lambda\Pi(x')_{s^*} + \sum_{s\in\mathcal{S}} \frac{\Pi(x')_s}{\Pi(x')_{s^*}} x_s. \end{aligned} \tag{2.16}$$

With these we can define the MDP $\Phi = (\mathcal{S}, \mathcal{A}, C, P^*)$ as the discrete-time analogue of the CTMDP Ψ.

2.4.2 Structural Results

Our goal in this section will be to show that average cost optimal policies for Ψ are *threshold policies*.

Definition 3. A policy $\pi : X \to \mathcal{A}$ is a *threshold policy* if for any $x \in X$ such that $\pi(x) = \text{Depart}_{\mathcal{L}(x)}$, then for all $x \leq x'$ with $\mathcal{L}(x) = \mathcal{L}(x')$ we have that $\pi(x') = \text{Depart}_{\mathcal{L}(x)}$.

Our goal in this section will be to show that average cost optimal policies for Ψ are threshold policies.

Lemma 7. *Let $x, x' \in X$ be given with $\mathcal{L}(x) \subseteq \mathcal{L}(x')$. Then $\Pi(x)_s \leq \Pi(x')_s$ for all $s \in \mathcal{L}(x)$.*

Proof. We show that $\Pi(x)_s \leq \Pi\left(x + e^{s'}\right)$ for any $s' \in \mathcal{S}$; the result follows immediately. Let $s \in \mathcal{L}(x)$ and $s' \in \mathcal{S}$ be given. If $s \leq s'$, then $\Pi(x)_s = \Pi\left(x + e^{s'}\right)$ by construction. If $s > s'$, then $\Pi(x)_s \leq \Pi\left(x + e^{s'}\right)$, since $\Pi\left(x + e^{s'}\right)$ must make a nonnegative length detour from $\Pi(x)$ to visit s' before visiting s. □

Lemma 8. *Let $x_1, x_1', x_2, x_2' \in X$ be given with $\mathcal{L}(x_1) = \mathcal{L}(x_1') = \mathcal{L}(x_2) = \mathcal{L}(x_2')$. Then $P_{x_1,x''}\left(\text{Depart}_{x_1'}\right) = P_{x_2,x''}\left(\text{Depart}_{x_2'}\right)$ for all $x'' \in X$.*

Proof. From Equation (2.5),

$$P_{x_1,\,x''}\left(\text{Depart}_{x_1'}\right) = \Pr\left[x'' = \sum_{i \in [I]} e^{S_i}\right]$$
$$P_{x_2,\,x''}\left(\text{Depart}_{x_2'}\right) = \Pr\left[x'' = \sum_{i \in [I']} e^{S_i}\right],$$

where I conditional on Λ is a Poisson random variable with rate parameter $\lambda\Lambda$, Λ is an exponentially distributed random variable with rate parameter $\Pi(x_1')_{s^*}$, and I' and Λ' are defined analogously. Since $\mathcal{L}(x_1') = \mathcal{L}(x_2')$, from Lemma 7 we have that $\Pi(x_1')_{s^*} = \Pi(x_2')_{s^*}$, and so I and I' are identically distributed. This ensures that

$$P_{x_1,x''}\left(\text{Depart}_{x_1'}\right) = P_{x_2,x''}\left(\text{Depart}_{x_2'}\right),$$

as required. □

Lemma 9. *For the MDP Φ, $V_\alpha(x) \leq V_\alpha(x')$ for all $x \leq x'$ with $\mathcal{L}(x) \subseteq \mathcal{L}(x')$.*

Proof. We show that for any $s' \in \mathcal{S}$, $V_\alpha(x) \leq V_\alpha\left(x + e^{s'}\right)$; the result then immediately follows. We argue via coupling. Let π^* be a stationary policy realizing V_α as per Theorem 5, and let $s \in \mathcal{S}$ be given. Let T be the first random transition at which π^* does not choose the Remain action given that we begin in state $x + e^s$, noting that it is possible that $T = \infty$ if π^* never does so, and let $\hat{x} + e^{s'}$ be the random state that π^* observes at transition T. Define the non-stationary policy π that chooses the Remain action for the first $T - 1$ transitions, chooses the $\text{Depart}_{\hat{x}+e^{s'}}$ at transition T, and then follows π^* exactly for all remaining transitions. We show that π incurs no more cost than π^* along this sample path.

By construction both π and π^* take the Remain action for the first $T - 1$ transitions. During each of these transitions the cost incurred by π is strictly less than the cost incurred by π^*, since from Equation (2.16)

$$\begin{aligned} C\left(x, \text{Remain}\right) &= \sum_{s \in \mathcal{S}} x_s \\ &< \sum_{s \in \mathcal{S}} x_s + 1 \\ &= \sum_{s \in \mathcal{S}} \left(x + e^{s'}\right)_s \\ &= C\left(x + e^{s'}, \text{Remain}\right) \end{aligned}$$

for any $x \in X$. At transition T both π and π^* choose the $\text{Depart}_{\hat{x}+e^{s'}}$ action, and follow the tour $\Pi\left(\hat{x} + e^{s'}\right)$. Since

$$\begin{aligned} C(\hat{x}, \text{Depart}_{\hat{x}+e^{s'}}) &= \lambda \Pi \left(\hat{x} + e^{s'}\right)_{s^*} + \sum_{s \in \mathcal{S}} \frac{\Pi(\hat{x})_s}{\Pi\left(\hat{x} + e^{s'}_{s^*}\right)} \hat{x}_s \\ &\leq \lambda \Pi \left(\hat{x} + e^{s'}\right)_{s^*} + \sum_{s \in \mathcal{S}} \frac{\Pi\left(\hat{x} + e^{s'}\right)_s}{\Pi\left(\hat{x} + e^{s'}\right)_{s^*}} \left(\hat{x}_s + e^{s'}\right) \\ &= C(\hat{x} + e^{s'}, \text{Depart}), \end{aligned}$$

π again incurs a cost of no more than that incurred by π^*. Both policies will return to the depot in the same state, having departed at the same time and

for the same duration. Since from this point onward π follows π^* exactly, both policies incur identical costs moving forward. Thus $V_\alpha(x) \leq V_\alpha\left(x + e^{s'}\right)$, as required. □

Lemma 10. *Let $\alpha \in (0,1)$ be given and define the set*

$$\hat{\mathcal{A}} = \{\text{Remain}\} \cup \left\{\text{Depart}_{\mathcal{L}(x)} \mid x \in X \setminus \{\mathbf{0}\}\right\}.$$

There exists an optimal policy π^* for Φ that uses only the actions in $\hat{\mathcal{A}}$.

Proof. Let an optimal policy π^* be given, and consider some state x where a policy chooses $\text{Depart}_{x'}$ for some $x' \in X$ with $\mathcal{L}(x) = \mathcal{L}(x')$. We show that the policy $\hat{\pi}^*$, defined as

$$\hat{\pi}^*(\hat{x}) = \begin{cases} \text{Depart}_{\mathcal{L}(x)}, & \hat{x} = x \\ \pi^*(\hat{x}), & \text{otherwise} \end{cases} \qquad \forall \hat{x} \in X,$$

is optimal as well. It then follows that there exists an optimal policy that does not choose actions outside those in $\hat{\mathcal{A}}$. From Theorem 5,

$$\begin{aligned} V_\alpha(x) &= C\left(x, \text{Depart}_{x'}\right) + \alpha \sum_{x'' \in X} P^*_{x,\,x''}\left(\text{Depart}_{x'}\right) V_\alpha(x'') \\ &= \min_{a \in \mathcal{A}_x} \left\{ C(x,a) + \alpha \sum_{x'' \in X} P^*_{x,\,x''}(a) V_\alpha(x'') \right\}. \end{aligned}$$

However, from Lemma 8 and Equation (2.16) we have that

$$\begin{aligned} V_\alpha(x) &= C\left(x, \text{Depart}_{x'}\right) + \alpha \sum_{x'' \in X} P^*_{x,\,x''}\left(\text{Depart}_{x'}\right) V_\alpha(x'') \\ &= C\left(x, \text{Depart}_{\mathcal{L}(x)}\right) + \alpha \sum_{x'' \in X} P^*_{x,\,x''}\left(\text{Depart}_{\mathcal{L}(x)}\right) V_\alpha(x'') \\ &= V_{\alpha,\hat{\pi}^*}(x), \end{aligned}$$

and so the policy π^* is no better than the policy $\hat{\pi}^*$. □

Theorem 4. *Let $\alpha \in (0,1)$ be given along with an optimal policy π^* as in Lemma 10. Then π^*_α is a threshold policy.*

Proof. Let an optimal policy π^* be given that restricts itself to actions in $\hat{\mathcal{A}}$, shown to exist by Lemma 10. Let $x \in X$ be given for which $\pi^*(x) = \text{Depart}_{\mathcal{L}(x)}$ along with $s' \in \mathcal{L}(x)$; if no such x exists, we are done. We want to show that $\pi\left(x + e^{s'}\right) = \text{Depart}_{\mathcal{L}(x)}$; if so, we have proved our claim. From Theorem 5 we have that

$$V_\alpha(x) = \min\left\{V_{\alpha,\text{ Remain}}(x),\ V_{\alpha,\text{ Depart}}(x)\right\},$$

where

$$V_{\alpha,\text{ Remain}}(x) = C\left(x, \text{Remain}\right) + \alpha \sum_{x'' \in X} P^*_{x,\,x''}(\text{Remain}) V_\alpha(x'')$$

$$V_{\alpha,\text{ Depart}}(x) = C\left(x, \text{Depart}_{\mathcal{L}(x)}\right) + \alpha \sum_{x'' \in X} P^*_{x,\,x''}\left(\text{Depart}_{\mathcal{L}(x)}\right) V_\alpha(x'').$$

From Lemma 7 we have that

$$\begin{aligned}
C\left(x + e^{s'}, \text{Depart}_{\mathcal{L}(x)}\right) &= \lambda \Pi\left(\mathcal{L}(x)\right)_{s^*} + \sum_{s \in \mathcal{S}} \frac{\Pi\left(x + e^{s'}\right)_s}{\Pi\left(\mathcal{L}(x)\right)_{s^*}} \left(x + e^{s'}\right)_s \\
&= \lambda \Pi(x)_{s^*} + \sum_{s \in \mathcal{S}} \frac{\Pi(x)_s}{\Pi(x)_{s^*}} \left(x + e^{s'}\right)_s \\
&= \lambda \Pi(x)_{s^*} + \sum_{s \in \mathcal{S}} \frac{\Pi(x)_s}{\Pi(x)_{s^*}} x_s + \frac{\Pi(x)_{s'}}{\Pi(x)_{s^*}} \\
&= C\left(x, \text{Depart}_{\mathcal{L}(x)}\right) + \frac{\Pi(x)_{s'}}{\Pi(x)_{s^*}}.
\end{aligned}$$

Combining this with Lemma 8, we have that

$$\begin{aligned}
V_{\alpha,\,\text{Depart}_{\mathcal{L}(x)}}\left(x+e^{s'}\right) &= C\left(x+e^{s'},\text{Depart}_{\mathcal{L}(x)}\right) \\
&\quad+\alpha\sum_{x''\in X}P^{*}_{x+e^{s'},\,x''}\left(\text{Depart}_{\mathcal{L}(x)}\right)V_{\alpha}(x'') \\
&= C\left(x,\text{Depart}_{\mathcal{L}(x)}\right)+\frac{\Pi(x)_{s'}}{\Pi(x)_{s^*}} \\
&\quad+\alpha\sum_{x''\in X}P^{*}_{x,\,x''}\left(\text{Depart}_{\mathcal{L}(x)}\right)V_{\alpha}(x'') \\
&= V_{\alpha,\,\text{Depart}}\left(x\right)+\frac{\Pi(x)_{s''}}{\Pi(x)_{s^*}} \\
&\leq V_{\alpha,\,\text{Depart}}\left(x\right)+1,
\end{aligned}$$

since by construction $\Pi\left(x\right)_{s'}\leq\Pi\left(x\right)_{s^*}$. Finally, since π^* is an optimal policy and chose to depart in x, we have that

$$\begin{aligned}
V_{\alpha,\,\text{Depart}}\left(x+e^{s'}\right) &\leq V_{\alpha,\,\text{Depart}}\left(x\right)+1 \\
&= V_{\alpha}\left(x\right)+1 \\
&\leq V_{\alpha,\,\text{Remain}}(x)+1 \\
&= V_{\alpha,\,\text{Remain}}\left(x+e^{s'}\right).
\end{aligned}$$

Thus $\pi^*(x)=\text{Depart}_{\mathcal{L}(x)}$ implies that $\pi\left(x+e^{s'}\right)=\text{Depart}_{\mathcal{L}(x)}$ for all $s'\in\mathcal{L}(x)$, as required. □

Thus we have shown that in the α-discounted setting, optimal policies for Φ exhibit a threshold structure. We now argue that this result holds in the undiscounted case using Theorem 6. We must first show that Φ satisfies the conditions of the theorem.

Lemma 11. *Consider the policy π defined as*

$$\pi(x)=\text{Depart}_{\mathbf{1}}. \tag{2.17}$$

Then $V_{\alpha,\pi}(x)\leq 2\lambda\Pi(\mathbf{1})_{s^*}/(1-\alpha)$.

Proof. We begin by providing an upper bound on the cost incurred in each state. From Equation (2.16), Lemma 7, and the fact that $\Pi(\mathbf{1})_s \leq \Pi(\mathbf{1})_{s^*}$ for all $s \in \mathcal{S}$, we have that

$$\begin{aligned} C(x, \pi(x)) &= \lambda \Pi(\mathbf{1})_{s^*} + \sum_{s \in \mathcal{S}} \frac{\Pi(\mathbf{1})_s}{\Pi(\mathbf{1})_{s^*}} x_s \\ &\leq \lambda \Pi(\mathbf{1})_{s^*} + \sum_{s \in \mathcal{S}} x_s \end{aligned}$$

Thus the cost incurred at each transition is upper bounded by a function that is linear in the number of delivery requests in the state. Applying Lemma 8 gives us that

$$P^*_{x,\, x''}\left(\text{Depart}_{x+\mathbf{1}}\right) = P^*_{\mathbf{1},\, x''}\left(\text{Depart}_{\mathbf{1}}\right),$$

for all $w'' \in X$, and so at each transition the transition probabilities are independent of the current state. In particular, we transition to the random state $\chi = \sum_{i \in [I]} e^{s_i}$, where I given Λ is a Poisson random variable with rate parameter $\lambda\Lambda$ and Λ is an exponential random variable with mean $\Pi(\mathbf{1})_{s^*}$. Combining this with Equation (1.1), we have

$$\begin{aligned} V_{\alpha,\pi}(\mathbf{0}) &\leq \sum_{t=0}^{\infty} \alpha^t \, \mathrm{E}\left[C(\chi, \text{Depart}_{\mathbf{1}})\right] \\ &\leq \sum_{t=0}^{\infty} \alpha^t \, \mathrm{E}\left[\lambda \Pi(\mathbf{1})_{s^*} + \sum_{s \in \mathcal{S}} \chi_s\right]. \end{aligned}$$

Straightforward calculation shows that

$$\begin{aligned} \mathrm{E}\left[I\right] &= \mathrm{E}\left[\mathrm{E}\left[I \mid \Lambda\right]\right] \\ &= \mathrm{E}\left[\lambda\Lambda\right] \\ &= \lambda \Pi(\mathbf{1})_{s^*}, \end{aligned}$$

and so

$$V_{\alpha,\pi}(\mathbf{0}) \leq 2\lambda\Pi(\mathbf{1})_{s^*} \sum_{t=0}^{\infty} \alpha^t$$
$$= \frac{2\lambda\Pi(\mathbf{1})_{s^*}}{1-\alpha}.$$

□

Lemma 12. *The MDP Φ satisfies SEN 1—3 of Theorem 6.*

Proof. We show that each of the assumptions holds in turn for Φ. We take the zero vector **0** as our distinguished state. Throughout this proof it is assumed that $\alpha \in (0, 1)$ is given.

SEN 1. Since all costs C are nonnegative it suffices to show that $(1-\alpha)V_\alpha(\mathbf{0})$ is bounded above. Lemma 11 immediately provides this bound, since any feasible policy provides an upper bound on the cost of the optimal policy.

SEN 2. We show that

$$h_\alpha(x) \leq \frac{2\lambda\Pi(\mathbf{1})_{s^*}}{P^*_{\mathbf{1},\,\mathbf{0}}(\text{Depart})},$$

or equivalently that

$$V_{\alpha,\pi}(x) = \frac{2\lambda\Pi(\mathbf{1})_{s^*}}{P^*_{\mathbf{1},\,\mathbf{0}}(\text{Depart})} + V_\alpha(\mathbf{0}).$$

Consider the non-stationary policy $\hat{\pi}$ that follows π from Equation (2.17) in states $x \neq \mathbf{0}$ until the first time it reaches state **0**, and after which it follows some optimal policy π^*. We show that this policy leads to the state **0** from $x \neq \mathbf{0}$ with finite cost.

Let $x \neq \mathbf{0}$ be given, and consider following $\hat{\pi}$. Note that $P^*_{x,\,\mathbf{0}}(\text{Depart}_{x+\mathbf{1}}) > 0$. From Lemma 8 we have that

$$P^*_{x,\,\mathbf{0}}(\text{Depart}_{x+\mathbf{1}}) = P^*_{x+\mathbf{1},\,\mathbf{0}}\left(\text{Depart}_{\mathbf{1}}\right) = P^*_{\mathbf{1},\,\mathbf{0}}\left(\text{Depart}_{\mathbf{1}}\right).$$

This means that at each transition the probability we transition to **0** is at least $P^*_{\mathbf{1},\,\mathbf{0}}(\text{Depart})$. Thus an upper bound on the number of transitions we need to take until we get to state **0** is X, where $X \in \{1, 2, \ldots\}$ is a geometric random variable with success probability $P^*_{\mathbf{1},\,\mathbf{0}}(\text{Depart})$. As shown in the proof of Lemma 11, this policy incurs a cost of at most $2\lambda\Pi(\mathbf{1})_{s^*}$ per transition. This gives us

$$\begin{aligned} V_{\alpha,\pi}(x) &\leq \mathrm{E}\left[\sum_{t=1}^{X} \alpha^{X-1} \cdot 2\lambda\Pi(\mathbf{1})_{s^*} + \alpha^X V_\alpha(\mathbf{0})\right] \\ &\leq \mathrm{E}\left[\sum_{t=1}^{X} 2\lambda\Pi(\mathbf{1})_{s^*} + V_\alpha(\mathbf{0})\right] \\ &= \frac{2\lambda\Pi(\mathbf{1})_{s^*}}{P^*_{\mathbf{1},\,\mathbf{0}}(\text{Depart})} + V_\alpha(\mathbf{0}), \end{aligned}$$

and so $h_\alpha(x)$ is bounded above as required.

SEN 3. We show that $0 \leq h_\alpha(x)$. Showing that $0 \leq h_\alpha(x)$ is equivalent to showing that $V_\alpha(\mathbf{0}) \leq V_\alpha(x)$, which follows immediately from Lemma 9. □

Lemma 13. *For any $\alpha \in (0, 1)$, let π^*_α be an α-discount optimal threshold policy as in Theorem 4. For all $x \in X$ such that $\sum_{s\in\mathcal{S}} x_s > 2\lambda\Pi(\mathbf{1})_{s^*}$, $\pi^*_\alpha(x) = \text{Depart}_{\mathcal{L}(x)}$.*

Proof. Let $x \in X$ be given with $\sum_{s\in\mathcal{S}} x_s > 2\lambda\Pi(\mathbf{1})_{s^*}$. Suppose for contradiction that $\pi^*_\alpha(x) = \text{Remain}$. Then

$$V_\alpha(x) = \sum_{s\in\mathcal{S}} x_s + \alpha(1 - \lambda\tau)V_\alpha(x) + \alpha \sum_{s\in\mathcal{S}} f_{\mathcal{S}}(s) V_\alpha\left(x + e^s\right).$$

From Lemma 9 we have that $V_\alpha\left(x + e^s\right) \geq V_\alpha(x)$, and so

$$\begin{aligned} V_\alpha(x) &\geq \sum_{s\in\mathcal{S}} x_s + \alpha(1 - \lambda\tau)V_\alpha(x) + \alpha \sum_{s\in\mathcal{S}} f_{\mathcal{S}}(s) V_\alpha\left(x\right) \\ &\geq \sum_{s\in\mathcal{S}} x_s + \alpha V_\alpha(x) \\ &\geq \frac{1}{1-\alpha} \sum_{s\in\mathcal{S}} x_s. \end{aligned}$$

By assumption we have that $\sum_{s\in\mathcal{S}} x_s > 2\lambda\Pi(\mathbf{1})_{s^*}$, and so

$$V_\alpha(x) > \frac{2\lambda\Pi(\mathbf{1})_{s^*}}{1-\alpha}.$$

However, from Lemma 11 we know that $V_\alpha(x) \leq 2\lambda\Pi(\mathbf{1})_{s^*}/(1-\alpha)$, a contradiction. □

Lemma 14. *For any $\alpha \in (0,1)$, let π^*_α be an α-discount optimal threshold policy as in Theorem 4. Then there exists $\alpha_1, \alpha_2, \ldots \in (0,1)$ with $\lim_{n\to\infty} \alpha_n \to 1$ such that*

$$\lim_{n\to\infty} \pi^*_{\alpha_n} \to \pi^*$$

exists where π^* is also a threshold policy.

Proof. Observe that a threshold policy may be fully characterized by the set of all states for which the Remain action is chosen, since for any state x not in that set by definition the $\text{Depart}_{\mathcal{L}(x)}$ action is taken. For any $\alpha \in (0,1)$ define

$$\mathcal{R}_\alpha = \{x \in X \mid \pi^*_\alpha(x) = \text{Remain}\}.$$

From Lemma 13 we have for any $\alpha \in (0,1)$ that $\pi^*_\alpha(x) = \text{Depart}_{\mathcal{L}(x)}$ for all $x \in X$ such that $\sum_{s\in\mathcal{S}} x_s > 2\lambda\Pi(\mathbf{1})_{s^*}$. Define

$$\mathcal{R} = \left\{ x \in X \mid \sum_{s\in\mathcal{S}} x_s \leq 2\lambda\Pi(\mathbf{1})_{s^*} \right\}.$$

This is a finite set, and by construction $\mathcal{R}_\alpha \subseteq \mathcal{R}$. Let a sequence $\alpha_1, \alpha_2, \ldots \in (0,1)$ with $\lim_{n\to\infty} \alpha_n \to 1$ be given. Then the sequence $\mathcal{R}_{\alpha_1}, \mathcal{R}_{\alpha_2}, \ldots$ is contained in the finite set $\mathcal{R}$ and so has some convergent subsequence $\mathcal{R}_{\beta_1}, \mathcal{R}_{\beta_2}, \ldots$ that converges to an element of $\mathcal{R}$, say $\mathcal{R}^*$. Thus

$$\lim_{n\to\infty} \pi^*_{\beta_n} \to \pi^*$$

exists and is a threshold policy, as required. □

2.5 Comparing the Two Settings

Computing an optimal threshold policy π^* for the CTMDP Ψ can be computationally expensive, whereas running Algorithm 1 on a given input requires only evaluating the β-approximation $\text{TSP}_\beta(\mathcal{S})$. For this reason we are interested in bounding the performance of Algorithm 1 relative to the performance of π^*, an optimal policy for Ψ, in the long-run average cost setting of Section 2.4. We consider the simple geometry where $\mathcal{S}$ is a finite subset of $[0, \Delta/2]$ where the depot is at 0; note that by construction the optimal TSP tour will depart from the depot, move to the furthest client at a distance $\Delta/2$, and then return, incurring a total distance of Δ. In this case we can consider $\beta = 1$ since the optimal tour is known.

In this simple geometry, the CTMDP of Section 2.4 can be reduced to a much smaller state space. In particular, we need only track the total number of delivery requests waiting for delivery, and the distance of the furthest client to which there is a delivery request. With these two pieces of information the tour we take when we depart and the costs incurred are known. To construct this equivalent CTMDP we move all delivery charges from the depart action to the remain action, noting that over any sequence of remain actions followed by a depart action we incur the same cost. We can then use any solution method to compute the long-run average cost incurred, for example, via value iteration [20].

Figure 2.1 shows the ratio between the performance of Algorithm 1 relative to the optimal threshold policy when $\mathcal{S} = \{0, 1/11, 2/11, \ldots, 10/11\}$. We considered uniform and (discretized) Beta distributions for the client distribution $f_{\mathcal{S}}$. As shown, for any arrival rates the performance guarantees are quite good. Note that the competitive ratio in this setting is $3/2 \cdot (10/11)/(1/11) - 1 = 14$, which is far worse than any of these performance guarantees, suggesting that

Algorithm 1 can perform well in place of using a difficult-to-compute optimal threshold policy.

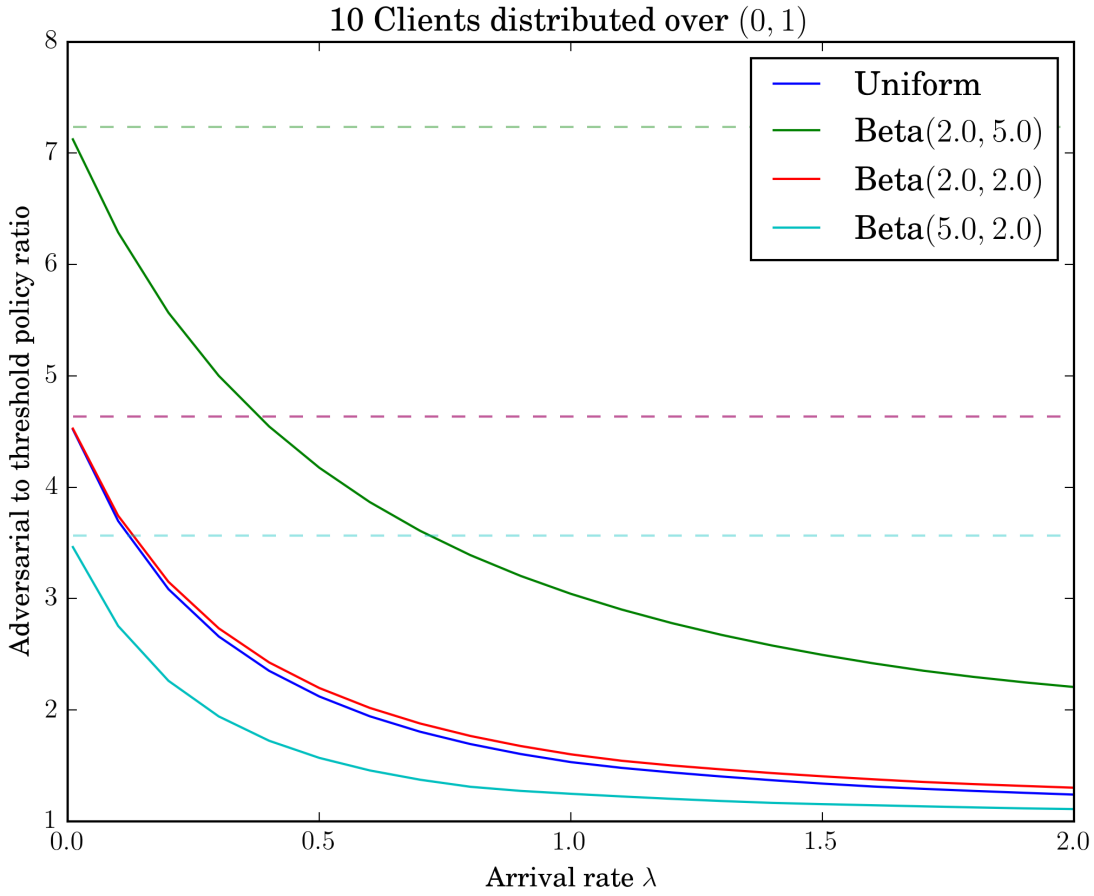

Figure 2.1: The ratio of the performance of Algorithm 1 relative to the optimal threshold policy for different arrival rates. The solid lines indicate the ratio for different arrival distributions. The associated dashed lines represent the worst-case ratio for any arrival rate. The worst-case guarantees are easily computed by evaluating the expected cost per delivery request of Algorithm 1, which depends only on the mean delivery request distance and not the arrival rate. The associated lower bound on the cost of optimal threshold policy is obtained by the structure of the cost functions.

CHAPTER 3

THE BASE SELECTION PROBLEM

3.1 Air Ambulance Routing at Ornge

The Ornge corporation offers air ambulance services for the Canadian province of Ontario [3]. Each day Ornge must transport a number of patients between locations across Ontario. Each patient, or *transfer request*, involves picking up a patient from one location and flying them to another. Local ground ambulances handle the transportation of the patient to and from any nearby hospital, so in this chapter we assume all patients are being transferred between airports.

Each of these transfer requests can have a number of healthcare-related side constraints. Constraints can include earliest pickup times, latest delivery times, or requiring the patient to be the only patient on board.

Ornge has access to a fleet of fixed-wing air ambulances which can be used to serve these requests. Different aircraft have different patient capacities, flight speeds, required ground time, and landing restrictions. Each aircraft has a home base where it begins its day and must return to at the end of the day. Thus when an aircraft is assigned to serve some set of requests it must choose a tour over pickup and delivery locations, starting and ending at its home base.

3.1.1 The Single-Day Problem

Each day Ornge is presented with a number of non-emergency transfer requests to serve. These transfer requests are fully known ahead of time, and so an offline schedule can be developed to serve them. Emergency requests are handled by Ornge separately from non-emergency requests, and so we will ignore emergency requests in this chapter.

Given a set of transfer requests to serve, Ornge's goal is to find a minimum-cost routing of all transfer requests utilizing its available fleet. Each transfer request must be served by an aircraft in a manner that satisfies its side constraints, while each aircraft has limits on how long it can operate each day and how many patients it can have on board at any time.

This problem is a variant of the dial-a-ride problem [14], where each transfer request has a number of additional side constraints. This problem can also be expressed as a set partitioning integer programming (IP) problem [7] when we do additional work to compute the objective coefficients [12, 13]. In this formulation, for each subset of transfer requests and for each aircraft we associate a binary variable. The objective coefficient of this variable is the optimal cost of serving the subset of requests with that aircraft. We must then select subset-aircraft pairs so that each request is in exactly one selected pair, and each aircraft is used in at most one selected pair.

Formally, let $\mathcal{L}$ be the set of airports that Ornge services. For each $\ell \in \mathcal{L}$ let $\mathcal{P}_\ell$ be the set of aircraft in the Ornge fleet with home base ℓ, and define $\mathcal{P} = \bigcup_{\ell \in \mathcal{L}} \mathcal{P}_\ell$ as the Ornge fleet. Let $\mathcal{R}$ be the set of transfer requests to be served, and let $\mathbf{P}(\mathcal{R})$ be the power set of $\mathcal{R}$. Each $r \in \mathcal{R}$ has an origin and destination in $\mathcal{L}$, along with a (possibly empty) set of side constraints to be respected. Let an aircraft $i \in \mathcal{P}$ and a set $j \in \mathbf{P}(\mathcal{R})$ be given. We define c_{ij} as the optimal cost of serving all requests in j with aircraft i, where $c_{ij} = \infty$ if there is no feasible schedule. See [13] for a practical treatment of how to compute such c_{ij}. Then the following integer program models the daily transfer request problem at Ornge.

$$
\begin{aligned}
\mathcal{Q}(\mathcal{P},\mathcal{R}) = \min \quad & \sum_{i\in\mathcal{P}} \sum_{j\in\mathbf{P}(\mathcal{R})} c_{ij}x_{ij} \\
\text{s.t.} \quad & \sum_{i\in\mathcal{P}} \sum_{j\in\mathbf{P}(\mathcal{R}):r\ni j} x_{ij} = 1 \qquad \forall r \in \mathcal{R} \\
& \sum_{j\in\mathbf{P}(\mathcal{R})} x_{ij} \leq 1 \qquad \forall i \in \mathcal{P} \\
& x_{ij} \in \{0,1\} \quad \forall i \in \mathcal{P},\ j \in \mathbf{P}(\mathcal{R}).
\end{aligned}
\tag{3.1}
$$

The variable x_{ij} indicates whether we serve subset $j \in \mathbf{P}(\mathcal{R})$ with aircraft i. The first set of constraints ensures that each request is served by some subset-aircraft pair. The second set of constraints ensures that each aircraft serves at most one set of requests.

This formulation has a number of useful properties. First, for problem instances that actually arise at Ornge this integer program can be solved to optimality in an acceptable amount of time. (One practical constraint enforced is that we only consider subsets of requests of size at most four. Although larger subsets can be considered, it increases the time required to solve the problem dramatically and offers little benefit, since the duty day constraints on aircraft will typically be violated serving such subsets.) Second, the varied side constraints on both the aircraft and transfer requests are easily enforced during the computation of the objective coefficients c_{ij}. A software tool solving this problem is used by Ornge to help plan transfer requests each day at Ornge [13].

3.2 Base Selection

Ornge owns a number of air ambulances with which it serves both emergency and non-emergency calls. However there are also a number of *standing agreement* (SA) aircraft that can be used by Ornge for a certain number of hours per

year. Each SA aircraft has a home base that it begins its day and must return to at the end of each day, just as the Ornge aircraft do. These SA contracts are periodically renegotiated, offering the possibility of choosing SA aircraft at different locations. Table 3.1 lists the current SA aircraft available. Our goal is to decide where to locate SA aircraft in order to minimize the long-run expected operating cost for Ornge.

Carrier	Count	Home base
Air Bravo	1	Barrie-Orillia
Air Bravo	1	Thunder Bay
NAS	1	Thunder Bay
NAS	1	Muskoka
SkyCare	2	Sioux Lookout
Thunder Air	2	Timmins
Thunder Air	2	Thunder Bay
Wabusk	1	Moosonee

Table 3.1: The fleet of 11 SA aircraft currently available to Ornge. There are 32 additional potential SA aircraft from various locations in Ontario to choose from.

3.2.1 Stochastic Programming Formulation

Our objective is to choose a set of SA aircraft locations, subject to some budget constraint, that minimizes the average cost of operations over some time horizon. We can formulate this problem as a 2-stage stochastic optimization problem. In the first stage we choose a set of SA aircraft to operate, and in the second stage we solve a number of single-day problems to compute the total cost implied by that selection.

As before let $\mathcal{L}$ be the set of airports that Ornge services. We now augment $\mathcal{P}$ and $\mathcal{P}_\ell$ for each $\ell \in \mathcal{L}$ to include the potential new SA aircraft, and denote $\hat{\mathcal{P}} \subseteq \mathcal{P}$ as the set of aircraft that will be guaranteed to remain open after the first stage decisions. We are allowed to choose at most $N_{\mathcal{P}}$ aircraft and at most $N_{\mathcal{L}}$ base

locations. Finally, let $\mathbf{R}$ be a random variable over the set of transfer requests to be served on a given day, including over all associated side constraints. This gives us the following 2-stage stochastic integer program, where $\mathcal{Q}$ is defined in (3.1).

$$\begin{aligned}
\min \quad & \sum_{i \in \mathcal{P}} F_i y_i + \mathrm{E}\left[Q(\{i \in \mathcal{P} \mid z_i = 1\}, \mathbf{R})\right] \\
\text{s.t.} \quad & \sum_{i \in \mathcal{P}} z_i \leq N_{\mathcal{P}} \\
& \sum_{\ell \in \mathcal{L}} y_\ell \leq N_{\mathcal{L}} \\
& \sum_{i \in \mathcal{P}_\ell} z_i \leq |\mathcal{P}_\ell| \, y_\ell \quad \forall \ell \in \mathcal{L} \\
& z_i = 1 \qquad \forall i \in \hat{P} \\
& z_i \in \{0,1\} \quad \forall i \in \mathcal{P} \setminus \hat{\mathcal{P}} \\
& y_\ell \in \{0,1\} \quad \forall \ell \in \mathcal{L}.
\end{aligned} \tag{3.2}$$

Here the variable z_i indicates whether or not aircraft $i \in \mathcal{P}$ will be utilized, and variable y_ℓ indicates whether or not location $\ell \in \mathcal{L}$ will be utilized by a chosen aircraft. The first constraint ensures that we choose at most $N_{\mathcal{P}}$ aircraft, and the second constraint ensures that we choose at most $N_{\mathcal{L}}$ base locations. The third constraint ensures that aircraft can only be chosen if their corresponding base has been opened. The fourth constraint is used to force us to choose all aircraft in $\hat{\mathcal{P}}$, as these aircraft are not up for renegotiation. Finally, the fifth and sixth constraints ensure that all choices are binary.

3.2.2 Extensive Form Formulation

The random variable $\mathbf{R}$ is high-dimensional. In addition to a pickup location, a delivery location, an earliest pickup time, and a latest delivery time, each re-

quest can also have additional side constraints. To overcome the need to model the distribution of $\mathbf{R}$ explicitly we rely on previously observed data to form an empirical distribution. Let $\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_n$ be request sets from n previously observed days. Assuming that $\mathbf{R}$ has the empirical distribution associated with $\mathcal{R}_1, \ldots, \mathcal{R}_n$, then we can write (3.2) in extensive form as follows [10].

$$
\begin{aligned}
\min \quad & \sum_{i \in \mathcal{P}} F_i y_i + \frac{1}{n} \sum_{t \in [n]} \sum_{i \in \mathcal{P}} \sum_{j \in \mathcal{R}_t} c_{tij} x_{tij} \\
\text{s.t.} \quad & \sum_{i \in \mathcal{P}} \sum_{j \in \mathbf{P}(\mathcal{R}_t): r \in j} x_{tij} = 1 && \forall t \in [n]\,,\ r \in \mathcal{R}_t \\
& \sum_{j \in \mathbf{P}(\mathcal{R}_t)} x_{tij} \leq z_i && \forall t \in [n]\,,\ i \in \mathcal{P} \\
& \sum_{i \in \mathcal{P}} z_i \leq N_{\mathcal{P}} \\
& \sum_{\ell \in \mathcal{L}} y_\ell \leq N_{\mathcal{L}} \\
& \sum_{i \in \mathcal{P}_\ell} z_i \leq y_\ell && \forall \ell \in \mathcal{L} \\
& z_i = 1 && \forall i \in \hat{P} \\
& z_i \in \{0,1\} && \forall i \in \mathcal{P} \setminus \hat{\mathcal{P}} \\
& y_\ell \in \{0,1\} && \forall \ell \in \mathcal{L} \\
& x_{tij} \in \{0,1\} && \forall t \in [n]\,,\ i \in \mathcal{P},\ j \in \mathbf{P}(\mathcal{R}_t).
\end{aligned}
\tag{3.3}
$$

Here the first, second, and final constraints are time-indexed variants of the constraints in program (3.1). The first constraint ensures that for each sub-problem $t \in [n]$ that all requests in $\mathcal{R}_t$ are served. The second constraint ensures that each aircraft can be only utilized if that aircraft has been chosen for use. The third through eighth constraints serve the same role as in program (3.2). The final constraint ensures that for each sub-problem all decisions are binary.

3.3 Results

Ornge provided 198 records of daily request sets from May 6, 2015 to April 25, 2016. We solved the program (3.3) using 60 request sets randomly-chosen from this collection as the scenarios in the objective function. The choice to use only a subset of the available data is due to practical constraints; Table 3.4 shows that solving the IP even with only 60 request sets uses a large amount of memory. Solving the IP with additional request sets is computationally infeasible. We considered 43 different aircraft to open across 22 base locations. We set $F_i = 0$ for all aircraft, as the cost of choosing an aircraft is either unknown or uncertain. Instead we solve the program for a number of choices of $N_{\mathcal{L}}$ and $N_{\mathcal{P}}$. In particular, we focused on the following scenarios listed in Table 3.2.

Scenario	$\lvert\hat{\mathcal{P}}\rvert$	$N_{\mathcal{P}}$	$N_{\mathcal{L}}$
(Current)	11	11	9
1.1	11	12	9
1.2	11	13	9
1.3	11	14	9
2.1	9	10	9
2.2	9	11	9
2.3	9	12	9
2.4	9	13	9
2.5	9	14	9
2.6	9	15	9
(All)	44	44	22

Table 3.2: List of scenarios considered for the SA base selection problem. The forced aircraft column represents the size of $\hat{P}$. The total aircraft column indicates the aircraft budget $N_{\mathcal{P}}$, while the total bases column indicates the base budget $N_{\mathcal{L}}$. The scenario "(Current)" represents Ornge's current fleet, while the scenario "(All)" represents using all 44 potential aircraft. The "(All)" scenario is infeasible from Ornge's perspective, but provides a useful bound on the cost on what can be achieved.

As a pre-processing step we compute the objective coefficients for each of the 198 available days and store them in a database, as these coefficients are

identical across the different scenarios.

3.3.1 Direct Computation with Gurobi

We created a Gurobi [15] model in C++ to directly solve the extensive form IP from Equation (3.3). Each scenario was then constructed by querying the objective coefficients from the database, assembling the model, and passing it to Gurobi to be solved. All but one scenario, Scenario 2-2, was solved to optimality. Scenario 2-2 failed to solve, with the machine running out of memory. This happened even when warm starting Gurobi using the optimal solution from Scenario 2-1. The costs associated with each solution are listed in Table 3.3, while the run time characteristics are described in Table 3.4.

Scenario	$\mathrm{E}[\phi_1]$	$\mathrm{E}[\phi_2]$	$\mathrm{E}[\phi_3]$
(Current)	2100.75	5.13	1.29
1.1	2087.31	5.09	1.28
1.2	2066.58	5.04	1.27
1.3	2046.71	4.99	1.26
2.1	2133.06	5.21	1.32
2.2	—	—	—
2.3	2058.66	5.02	1.27
2.4	2041.26	4.98	1.26
2.5	2031.35	4.95	1.25
2.6	2027.98	4.94	1.25
(All)	1987.36	4.86	1.23

Table 3.3: The costs associated with each scenario. Here ϕ_1, ϕ_2, and ϕ_3 are random variables describing cost per distance requested, the cost per distance requested, and the distance flown per distance requested for a day. The distance requested in a single day is the sum over all requests of the origin to destination distance. Costs are expressed in dollars, and distances are expressed in kilometers. Scenario 2-2 failed to solve with Gurobi running out of memory, even when warm-starting the solve using the solution from Scenario 2.1.

Scenario	CPU Time (hours)	Wall time (hours)	Max Memory (GB)
1-1	20.45	2.86	17.06
1-2	15.51	2.24	17.07
1-3	5.92	1.00	15.59
2-1	16.12	2.28	12.68
2-2	12.27	1.85	30.73
2-3	8.64	1.32	17.53
2-4	4.19	0.77	16.45
2-5	4.79	0.84	15.94
2-6	11.35	1.66	20.22

Table 3.4: The computing resources required to solve each scenario. Scenario 2-2 was terminated after running out of available memory, even when warm started with a solution from Scenario 2-1. The "(Current)" and "(All)" scenarios have no first stage decisions, and so were solved entirely during the pre-processing step.

3.3.2 Decomposition

An alternate strategy to directly solving the extensive form IP in Equation (3.3) is to utilize the integer L-shaped method for stochastic programming. For a detailed explanation of the integer L-shaped method, see Section 8.1 of [10].

The linear programs used in the L-shaped method, specialized to our problem, are shown in Section A.2. It is useful to explore the constraint cuts generated by the L-shaped method.

The first type of cuts generated are feasibility cuts. These cuts take the form

$$\sum_{i \in \mathcal{P}} \sigma_i z_i \geq \sum_{r \in \mathcal{R}_t} \sigma_r, \tag{3.4}$$

where σ is a dual optimal solution to the linear program in Equation A.2 for some sub-problem $t \in [n]$, and σ is indexed according to the constraints in the

program. The dual of the linear program can be written as

$$\begin{aligned} \max \quad & \sum_{r \in \sigma_r} \sigma_r - \sum_{i \in \mathcal{P}} z_i \sigma_i \\ \text{s.t.} \quad & \sum_{r \in j} \sigma_r \leq \sigma_i \qquad \forall i \in \mathcal{P},\ j \in \mathbf{P}(\mathcal{R}_t) \\ & \sigma_r \in [0,1] \quad \forall r \in \mathcal{R}_t \\ & \sigma_i \in [0,1] \quad \forall i \in \mathcal{P}. \end{aligned} \tag{3.5}$$

Note that the feasibility cut enforces that the dual objective must be non-positive, which coincides with the feasibility LP having a zero objective value. Additionally, the dual variables are bounded between zero and one, and so the feasibility cut can be interpreted as requiring the first-stage LP to open enough planes to satisfy the total number of transfer requests that were unable to be served.

The second type of cuts generated are optimality cuts. These cuts take the form

$$\frac{1}{n} \sum_{t \in [n]} \sum_{i \in \mathcal{P}} \pi_{ti} z_i + \theta \geq \frac{1}{n} \sum_{t \in [n]} \sum_{r \in \mathcal{R}_t} \pi_{tr}, \tag{3.6}$$

where π_t is a dual optimal solution to the linear program in Equation A.3 corresponding to the tth sub-problem, and π_t is indexed according to the constraints in the program. The dual of the linear program can be written as

$$\begin{aligned} \max \quad & \sum_{r \in \mathcal{R}_t} \pi_{tr} - \sum_{i \in \mathcal{P}} z_i \pi_{ti} \\ \text{s.t.} \quad & \sum_{r \in j} \pi_{tr} \leq \pi_{ti} + c_{tij} \quad \forall i \in \mathcal{P},\ j \in \mathbf{P}(\mathcal{R}_t) \\ & \pi_{tr} \geq 0 \qquad \forall r \in \mathcal{R}_t \\ & \pi_{ti} \geq 0 \qquad \forall i \in \mathcal{P}. \end{aligned} \tag{3.7}$$

Note that by taking an average over the objective functions of all sub-problems, the optimality cut ensures that the expected second stage cost θ is at least the average of the second stage objective values. For each $r \in \mathcal{R}_t$, the dual variable

π_{tr} can be interpreted as the contribution of transfer request r to the objective value. For each $i \in \mathcal{P}$, the dual variable π_{ti} can be interpreted as the savings achieved by utilizing aircraft i.

We implemented the integer L-shaped method for the base selection problem, using Gurobi as the underlying linear programming solver. While small problems were able to be solved to optimality, problems with even a few days of sub-problems took tens of hours to complete or exhausted all available memory. The majority of the work (and memory utilization) went into the branch-and-bound tree. I conjecture that this is due in part to the lack of integrality cuts such as Gomory cuts [8]. The difficulty in incorporating such cuts in the decomposition technique is due to the first stage decisions influencing the right-hand side of the constraints in the second stage, where the bulk of the integer variables are.

It is tempting to rely on Gurobi's impressive abilities as an IP solver to solve the second stage problems to optimality in one pass. However, doing so discards all dual information used to generate feasibility and optimality cuts in the first stage problem.

Another tempting avenue to speed up the decomposition routine would be to warm-start algorithm with a good-quality integer solution we can use to prune the branch-and-bound tree. A natural approach to this would be to utilize randomized rounding. Unfortunately, we have the following result.

Lemma 15. *The IP in Equation* (3.3) *has an unbounded integrality gap.*

Proof. Consider an instance of this IP with 2 aircraft, 1 sub problem, and 4 transfer requests. Each aircraft can feasibly serve one of the following subsets of requests: $\{1,2\}, \{3,4\}, \{1,3\}, \{2,4\}$. Let $j = 1,2,3,4$ index these subsets, re-

spectively. Aircraft 1 has objective coefficients

$$c_{111} = a \qquad c_{112} = a$$
$$c_{113} = b \qquad c_{114} = b,$$

while aircraft 2 has objective coefficients

$$c_{121} = b \qquad c_{122} = b$$
$$c_{123} = a \qquad c_{124} = a,$$

where $a < b$. Then the optimal fractional solution is to choose $x_{111} = x_{112} = 1/2$ and $x_{123} = x_{124} = 1/2$, achieving an objective value of $2a$. However, the only two feasible integer solutions are $x_{111} = x_{122} = 1$ or $x_{113} = x_{124}$, up to symmetry in the aircraft, all with a cost of $a + b$. Thus the integrality gap is at least $(a + b)/(2a) = 1/2 + b/a$. Since we are free to choose b large and a small, there is no finite integrality gap for this family of instances. □

CHAPTER 4

ONLINE EMERGENCY TRANSPORTATION DISPATCHING

4.1 Air Ambulance Routing at BCEHS

BC Emergency Health Services (BCEHS) offers both air and ground ambulance service in the Canadian province of British Columbia [2]. Each day BCEHS must transfer patients between locations across British Columbia. BCEHS has a fleet of land ambulances, fixed-wing (FW) aircraft, and rotor-wing (RW, helicopter) aircraft to provide these services. FW aircraft must land at airports, and so patients are moved to and from the airport using land ambulances, whereas RW aircraft can land at airports, helipads, and other sufficiently open locations such as the locations of roadside accidents.

Unlike Ornge, BCEHS does not serve urgent and non-urgent calls with separate fleets. This is in part because they have a smaller fleet available. Ornge operates 11 FW standing agreement aircraft just for non-urgent calls, and has a larger fleet of FW and RW aircraft for urgent calls, while BCEHS currently has nine aircraft available in total; see table 4.1 for details.

Name	Type	Count	Home base
Cessna Citation	FW (Jet)	1	Vancouver
Beechcraft King Air	FW (Turboprop)	1	Vancouver
Beechcraft King Air	FW (Turboprop)	2	Kelowna
Beechcraft King Air	FW (Turboprop)	1	Prince George
Sikorsky S76	RW	2	Vancouver
Sikorsky S76	RW	1	Prince Rupert
Bell 412	RW	1	Kamloops

Table 4.1: The fleet of 9 aircraft available to BCEHS.

Our goal is to identify policies with which BCEHS can schedule its fleet to serve all urgent and non-urgent calls. Calls arrive in an online fashion, each with an origin, destination, and urgency. Urgent calls typically involve patients

with time-sensitive, life-threatening conditions. These calls will often require a RW aircraft for pickup as the location of the associated accident may be far from an airport. Non-urgent calls typically indicate a transfer request to or from a hospital, and generally utilize FW aircraft. We ignore other healthcare-specific side constraints.

There are several objective functions we can consider. In particular, we can consider minimizing any of the following: pickup latency, time between a call arriving and the patient being boarded onto a transport; (total) latency, the time between the arrival of a call and the delivery of the patient to the destination; or the float time, the total time between the arrival of a call and the delivery of the patient to the destination minus the required transport time. We can additionally consider each of these quantities broken down by the urgency of calls being served.

Unlike in Chapter 3, we do not choose as our objective the total cost of service, as assigning a cost associated with degrading the quality of service for an urgent call is difficult. Similarly, it is not clear how much we should value the quality of service for urgent calls over non-urgent calls, and so we will look for Pareto efficient solutions.

4.1.1 Notation

To simulate policies we need to keep track of the state of the world. We say that a *world* W is a full description of the status of all known calls and transports, including locations, histories, and planned actions. We define $\mathcal{W}$ as the space of all possible worlds. A call R consists of an origin, a destination, and an urgency. We define $\mathcal{R}$ as the space of all possible calls.

4.1.2 Policies

We will be interested in exploring policies to govern the behavior of both FW and RW aircraft in the presence of a mixed stream of urgent and non-urgent calls. A policy maps the state of the world to an action to be taken. In particular, when presented with a world a policy must choose exactly one of the following options.

1. Schedule a pickup for a call on a transport, choosing where in the transports' queue the pickup occurs.
2. Schedule a delivery for a call on a transport, choosing where in the transports' queue the delivery occurs.
3. Schedule a transport to fly to another airport, choosing where in the transports' queue the flight occurs.
4. Request to be asked to provide an action at some later time.
5. Take no action.

We consider the following functions to build up candidate policies.

Greedy(f)**.** Perhaps the simplest policy imaginable considers each call as it arrives, and schedules it on the transport and in the position that will minimize the objective f for this call. Note that this policy might assign a call ahead of existing calls, delaying them.

GreedyNon(f)**.** This policy is identical to Greedy(f), except that calls will always be scheduled after already-scheduled calls.

Threshold(k, p, q)**.** This policy serves urgent calls with policy p and non-urgent calls with policy q. However, all non-urgent calls will not be scheduled until there are at least k transports available.

ReturnToBase(p)**.** This policy first allows the policy p to react to the state of the world. If p takes no action, then this policy will schedule an idle transport that is away from its home base to return to its home base.

In particular, we explore the policies Greedy, GreedyNon, Threshold(k, p, q) for each $k \in \{0, 1, ..., 9\}$ and each $p, q \in \{\text{Greedy}, \text{GreedyNon}\}$. Additionally, for each of these policies p, we consider ReturnToBase(p). We will use the notation of Table 4.2 to refer to such policies.

Notation	Explanation
GP	A greedy preemptive policy
GNP	A greedy non-preemptive policy
Tk (p, q)	A threshold policy with threshold k, urgent policy p, and non-urgent policy q
R	(As a suffix) return to base when idle

Table 4.2: Naming conventions used to label policies. For example, the label "T1 (GNP, GP) R" represents a threshold policy that has a threshold of 1, uses a greedy non-preemptive policy for urgent calls, a greedy preemptive policy for non-urgent calls, and finally sends transports back to their home base when idle.

4.2 Computational Results

We simulated daily operations for all calls from 2014. The data available from BCEHS only includes urgent calls that began as non-urgent calls. Thus the results in this section should be interpreted only as a first approximation of the true behavior of the system, as there should be more urgent calls. As it stands we considered a total of 17083 calls, 7409 of which are non-urgent and 9674

of which are urgent. Arrival times of the calls were drawn from the empirical call distribution from operations at Ornge, since the arrival times of calls in the BCEHS data are not available.

The performance of various policies for non-urgent and urgent calls are shown in Tables 4.3 and 4.4, respectively. The policies shown include the best 10 and worst 10 policies, as measured by the total latency of non-urgent calls. The confidence intervals listed are computed by batch means, assuming each month of calls is approximately independent, and represent the 95% confidence level.

Somewhat surprisingly, the very simple greedy non-preemptive policy which returns idle planes to their home base does quite well for both urgent and non-urgent calls. However, the actual policy used does not seem to matter as long as the return-to-base heuristic is also used. This suggests that geography is playing a large role in the performance of these policies. In particular, BCEHS has been operating for years, and so the home base of each aircraft are likely chosen to offer good service for typical demand patterns. This in turn suggests that careful base selection, as in Chapter 3, could provide a large benefit to BCEHS.

The utilization of each aircraft under each policy are tabulated in Section C.1. It is important to recall that the demand distribution we simulated does not account for all the calls that BCEHS receives, and so should not be interpreted as providing a perfectly accurate analysis. Rather, these results should be used to guide the exploration of a variety of policies that incorporate geography more explicitly.

4.3 Conclusions and Future Work

As discussed above, the results from this work are largely preliminary. The simulation model captures the core behavior of the air ambulances, but omits many

Policy	Pickup Latency	Total Latency	Float
GNP R	0.22 ± 0.01	1.21 ± 0.03	0.28 ± 0.02
GP R	0.23 ± 0.01	1.30 ± 0.03	0.37 ± 0.02
T1 (GNP, GNP) R	0.34 ± 0.01	1.42 ± 0.02	0.40 ± 0.02
GNP	0.34 ± 0.01	1.42 ± 0.02	0.40 ± 0.02
T0 (GNP, GNP)	0.34 ± 0.01	1.42 ± 0.02	0.40 ± 0.02
T1 (GNP, GNP)	0.34 ± 0.01	1.42 ± 0.02	0.40 ± 0.02
T0 (GNP, GNP) R	0.34 ± 0.01	1.42 ± 0.02	0.40 ± 0.02
T7 (GNP, GNP)	0.37 ± 0.01	1.45 ± 0.03	0.43 ± 0.02
T7 (GNP, GNP) R	0.37 ± 0.01	1.45 ± 0.03	0.43 ± 0.02
T7 (GNP, GP)	0.37 ± 0.01	1.48 ± 0.03	0.50 ± 0.02
T8 (GP, GP)	0.61 ± 0.02	1.85 ± 0.04	0.86 ± 0.03
T8 (GP, GP) R	0.61 ± 0.02	1.85 ± 0.04	0.86 ± 0.03
T9 (GNP, GP) R	1.44 ± 0.08	2.48 ± 0.09	1.52 ± 0.07
T9 (GNP, GP)	1.44 ± 0.08	2.48 ± 0.09	1.52 ± 0.07
T9 (GNP, GNP)	1.44 ± 0.08	2.48 ± 0.09	1.52 ± 0.07
T9 (GNP, GNP) R	1.44 ± 0.08	2.48 ± 0.09	1.52 ± 0.07
T9 (GP, GNP)	1.60 ± 0.06	2.76 ± 0.07	1.82 ± 0.06
T9 (GP, GP) R	1.60 ± 0.06	2.76 ± 0.07	1.82 ± 0.06
T9 (GP, GNP) R	1.60 ± 0.06	2.76 ± 0.07	1.82 ± 0.06
T9 (GP, GP)	1.60 ± 0.06	2.76 ± 0.07	1.82 ± 0.06

Table 4.3: Performance measures for a selection of policies for non-urgent calls. The total latency measures time between a request arriving in the system and the request being served. The pickup latency measures the time between a request arriving in the system and the request being boarded on an aircraft. The float time is the total latency minus the required travel time for each request. All units are hours.

Policy	Pickup Latency	Total Latency	Float
GNP R	0.43 ± 0.01	1.41 ± 0.01	0.52 ± 0.01
GP R	0.42 ± 0.01	1.42 ± 0.01	0.53 ± 0.01
T1 (GNP, GNP) R	0.59 ± 0.01	1.41 ± 0.02	0.69 ± 0.01
GNP	0.59 ± 0.01	1.41 ± 0.02	0.69 ± 0.01
T0 (GNP, GNP)	0.59 ± 0.01	1.41 ± 0.02	0.69 ± 0.01
T1 (GNP, GNP)	0.59 ± 0.01	1.41 ± 0.02	0.69 ± 0.01
T0 (GNP, GNP) R	0.59 ± 0.01	1.41 ± 0.02	0.69 ± 0.01
T7 (GNP, GNP)	0.59 ± 0.01	1.41 ± 0.02	0.70 ± 0.02
T7 (GNP, GNP) R	0.59 ± 0.01	1.41 ± 0.02	0.70 ± 0.02
T7 (GNP, GP)	0.59 ± 0.01	1.41 ± 0.02	0.70 ± 0.01
T8 (GP, GP)	0.59 ± 0.01	1.44 ± 0.02	0.73 ± 0.01
T8 (GP, GP) R	0.59 ± 0.01	1.44 ± 0.02	0.73 ± 0.01
T9 (GNP, GP) R	0.60 ± 0.01	1.41 ± 0.02	0.71 ± 0.02
T9 (GNP, GP)	0.60 ± 0.01	1.41 ± 0.02	0.71 ± 0.02
T9 (GNP, GNP)	0.60 ± 0.01	1.41 ± 0.02	0.71 ± 0.02
T9 (GNP, GNP) R	0.60 ± 0.01	1.41 ± 0.02	0.71 ± 0.02
T9 (GP, GNP)	0.59 ± 0.01	1.43 ± 0.02	0.74 ± 0.01
T9 (GP, GP) R	0.59 ± 0.01	1.43 ± 0.02	0.74 ± 0.01
T9 (GP, GNP) R	0.59 ± 0.01	1.43 ± 0.02	0.74 ± 0.01
T9 (GP, GP)	0.59 ± 0.01	1.43 ± 0.02	0.74 ± 0.01

Table 4.4: Performance measures for a selection of policies for urgent calls. The total latency measures time between a request arriving in the system and the request being served. The pickup latency measures the time between a request arriving in the system and the request being boarded on an aircraft. The float time is the total latency minus the required travel time for each request. All units are hours.

medically relevant details. The available data also omit an important class of calls, namely urgent calls that did not begin as non-urgent calls. The omission of such call is one likely explanation for the low utilization of the RW aircraft as shown in Table C.1. However, these results still point to the conclusion that the home base location of aircraft is an important factor influencing service quality. Although the results from Chapter 3 suggest that Ornge can reduce their costs by only a few percent by carefully choosing base locations, these improvements are to a schedules that are planned with full information available. In the on-line system of BCEHS it is possible (and the simulation data agree) that base locations are at least as important.

APPENDIX A

MISCELLANEOUS THEOREMS AND EQUATIONS

A.1 Useful Theorems

The following are useful theorems we rely on in Chapter 2.

Theorem 5 (Theorem 4.1.4 of [20]). *For an α-discounted MDP $\Phi = (\Omega, \mathcal{A}, C, P)$, the discounted value function V_α is the minimum nonnegative solution of*

$$V_\alpha(\omega) = \min_{a \in \mathcal{A}} \left\{ C(\omega, a) + \alpha \sum_{\omega' \in \Omega} P_{\omega,\,\omega'}(a) V_\alpha(\omega') \right\}, \qquad \forall \omega \in \Omega.$$

Theorem 6 (Theorem 7.2.3 of [20]). *Consider the following conditions for the MDP $\Phi = (\Omega, \mathcal{A}, C, P)$, and define $h_\alpha(\omega) = V_\alpha(\omega) - V_\alpha(\omega^*)$ for some distinguished $\omega^* \in \Omega$.*

SEN1. The quantity $(1-\alpha)V_\alpha(\omega^*)$ is bounded for all $\alpha \in (0,1)$.

SEN2. There exists $M \geq 0$ such that $h_\alpha(\omega) \leq M$ for all $\omega \in \Omega$ and all $\alpha \in (0,1)$.

SEN3. There exists $L \geq 0$ such that $-L \leq h_\alpha(\omega)$ for all $\omega \in \Omega$ and all $\alpha \in (0,1)$.

Let π_α^* be the α-discount optimal policy for Φ. If Φ satisfies SEN 1—3 and there exists a sequence $\alpha_n \to 1$ such that $\lim_{\alpha_n \to 1} \pi_{\alpha_n} = \pi$, then π is average cost optimal.

Theorem 7 (Theorem 8.6 of [11], Yao's principle). *Here we consider request-answer systems for which there is no a priori bound on the number of requests, and so the cost of the game may be unbounded. Let ALG be any randomized online algorithm for the request-answer system. Let $\sigma^{(n)}$ be a random variable representing n-length arrival sequences with distribution $\mu^{(n)}$ for all $n \geq 1$. Let $\{ALG_{n,i}\}$ be the set of all deterministic algorithms for serving request sequences of length n. If*

$$\liminf_{n \to \infty} \frac{\inf_i \mathrm{E}_{\mu^{(n)}} \left[ALG_{n,i}(\sigma^{(n)}) \right]}{\mathrm{E}_{\mu^{(n)}} \left[OPT(\sigma^{(n)}) \right]} \geq c,$$

and

$$\limsup_{n\to\infty} \mathrm{E}_{\mu^{(n)}}\left[OPT(\sigma^{(n)})\right] = \infty,$$

then the competitive ratio of the algorithm is at least c.

A.2 Linear Programs for the L-Shaped Method

Refer to the notation used in Chapter 3. The master linear program is

$$\begin{aligned}
\min \quad & \sum_{i\in\mathcal{P}} F_i y_i + \theta \\
\text{s.t.} \quad & \sum_{i\in\mathcal{P}} z_i \le N_{\mathcal{P}} \\
& \sum_{\ell\in\mathcal{L}} y_\ell \le N_{\mathcal{L}} \\
& \sum_{i\in\mathcal{P}_\ell} z_i \le y_\ell && \forall \ell \in \mathcal{L} \\
& z_i = 1 && \forall i \in \hat{P} \\
& z_i \ge 0 && \forall i \in \mathcal{P}\setminus\hat{\mathcal{P}} \\
& y_\ell \ge 0 && \forall \ell \in \mathcal{L} \\
& \sum_{i\in\mathcal{P}} \sigma_i^k z_i \ge \sum_{r\in\mathcal{R}_{t_k}} \sigma_r^k && \forall k \in [k_1] \\
& \frac{1}{n}\sum_{t\in[n]}\sum_{i\in\mathcal{P}} (\pi^t)_i^k z_i + \theta \ge \frac{1}{n}\sum_{t\in[n]}\sum_{r\in\mathcal{R}_t} (\pi^t)_r^k && \forall k \in [k_2]\,,
\end{aligned} \tag{A.1}$$

where k_1 represents the number of feasibility cuts, t_k is the index of the feasibility linear program used in the constraint, and k_2 represents the number of

optimality cuts. The feasibility linear programs are

$$\begin{aligned}
\min \quad & \sum_{r\in\mathcal{R}_t}\left(v_r^+ + v_r^-\right) + \sum_{i\in\mathcal{P}}\left(v_i^+ + v_i^-\right) \\
\text{s.t.} \quad & \sum_{i\in\mathcal{P}}\sum_{j\in\mathbf{P}(\mathcal{R}_t):r\in j} x_{tij} + v_r^+ - v_r^- = 1 \quad \forall r\in\mathcal{R}_t \\
& \sum_{j\in\mathbf{P}(\mathcal{R}_t)} x_{tij} + v_i^+ - v_i^- \le z_i \quad \forall i\in\mathcal{P} \\
& x_{tij} \ge 0 \quad \forall i\in\mathcal{P},\ j\in\mathbf{P}(\mathcal{R}_t) \\
& v_i^+, v_i^- \ge 0 \quad \forall i\in\mathcal{P} \\
& v_r^+, v_r^- \ge 0 \quad \forall r\in\mathcal{R}_t
\end{aligned} \tag{A.2}$$

for each sub-problem $t\in[n]$, and the optimality linear programs are

$$\begin{aligned}
\min \quad & \sum_{i\in\mathcal{P}}\sum_{j\in\mathcal{R}_t} c_{tij}x_{tij} \\
\text{s.t.} \quad & \sum_{i\in\mathcal{P}}\sum_{j\in\mathbf{P}(\mathcal{R}_t):r\in j} x_{tij} = 1 \quad \forall r\in\mathcal{R}_t \\
& \sum_{j\in\mathbf{P}(\mathcal{R}_t)} x_{tij} \le z_i \quad \forall i\in\mathcal{P} \\
& x_{tij} \ge 0 \quad \forall i\in\mathcal{P},\ j\in\mathbf{P}(\mathcal{R}_t)
\end{aligned} \tag{A.3}$$

for each sub-problem $t\in[n]$.

APPENDIX B

DETAILED BASE SELECTION RESULTS

B.1 Base Choices

The following figures show the actual base choices associated with the results of Table 3.3. We show only the scenarios considered most useful by Ornge. The maps utilized in these plots were created by [4]

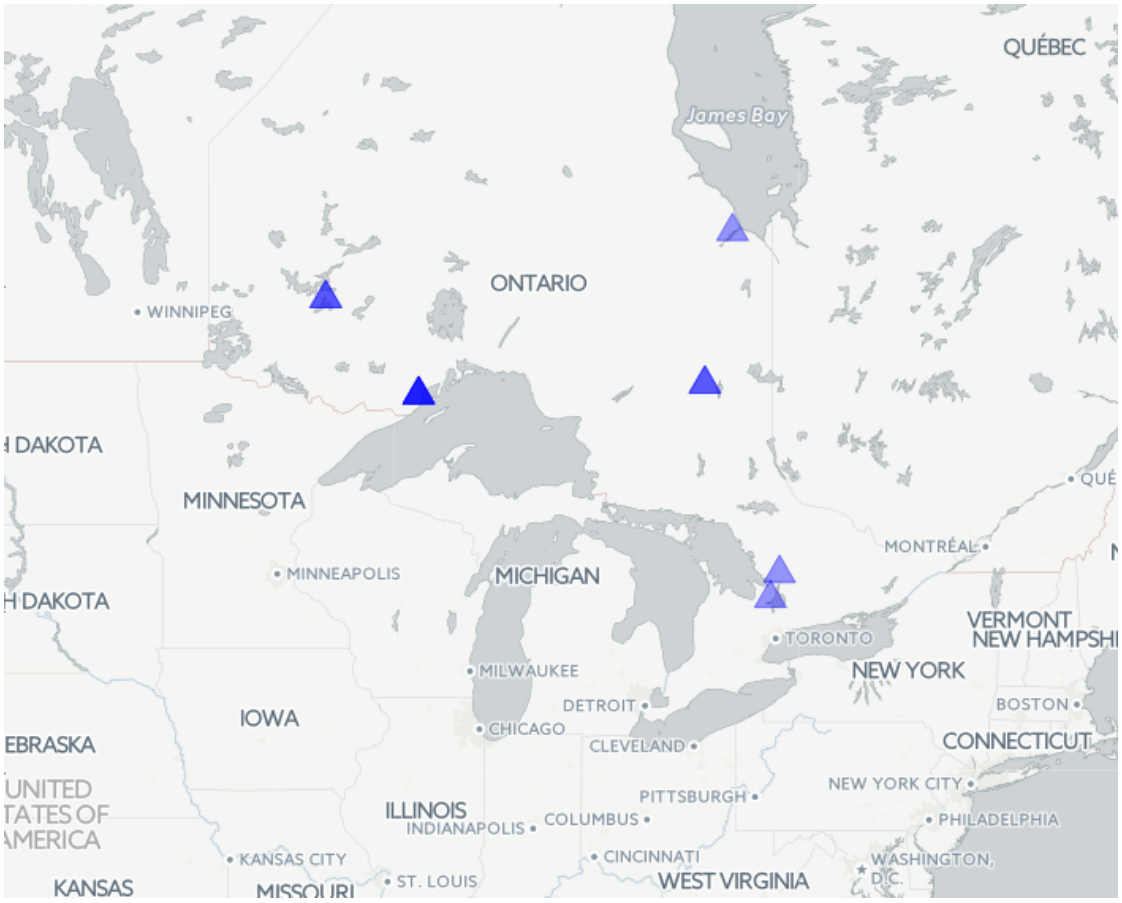

Figure B.1: The current location of SA aircraft at Ornge. Each triangle represents a base location; darker triangles indicate that multiple aircraft are located at that base.

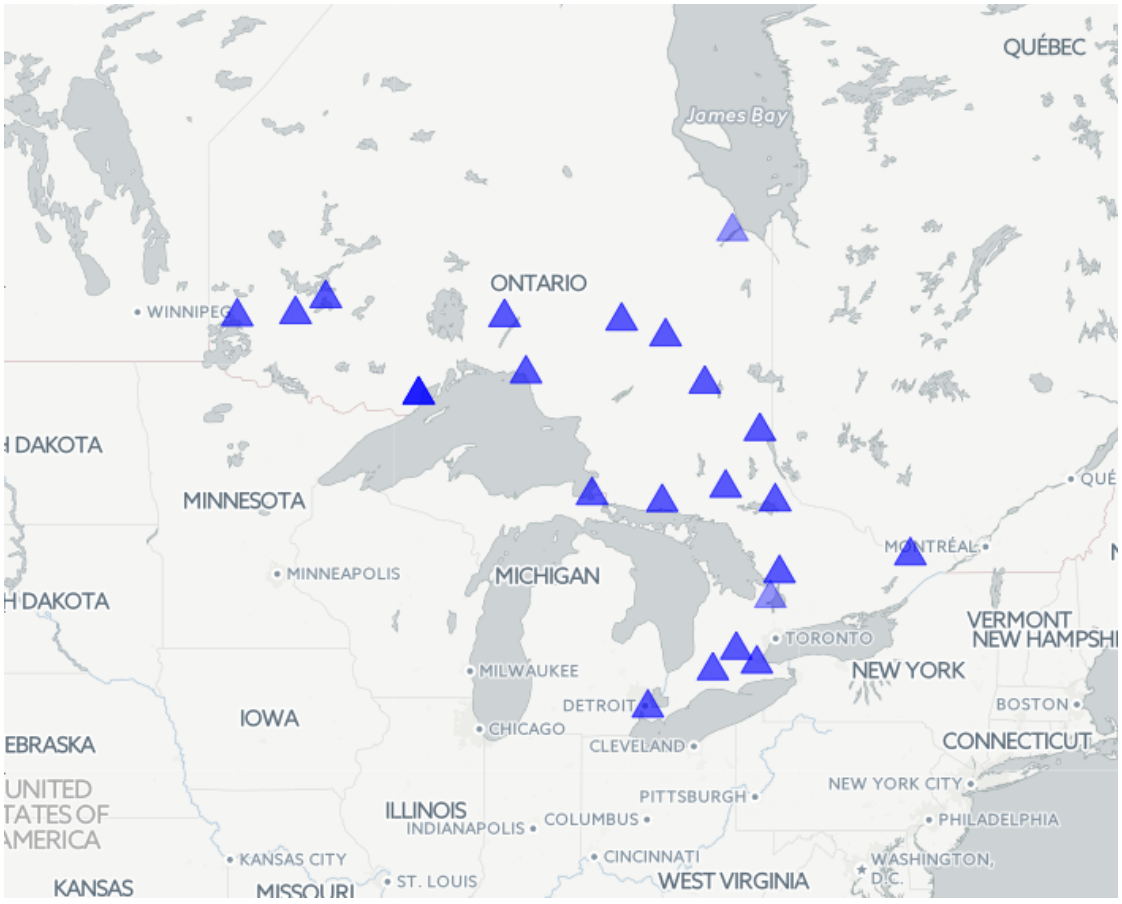

Figure B.2: The set of candidate SA base locations.

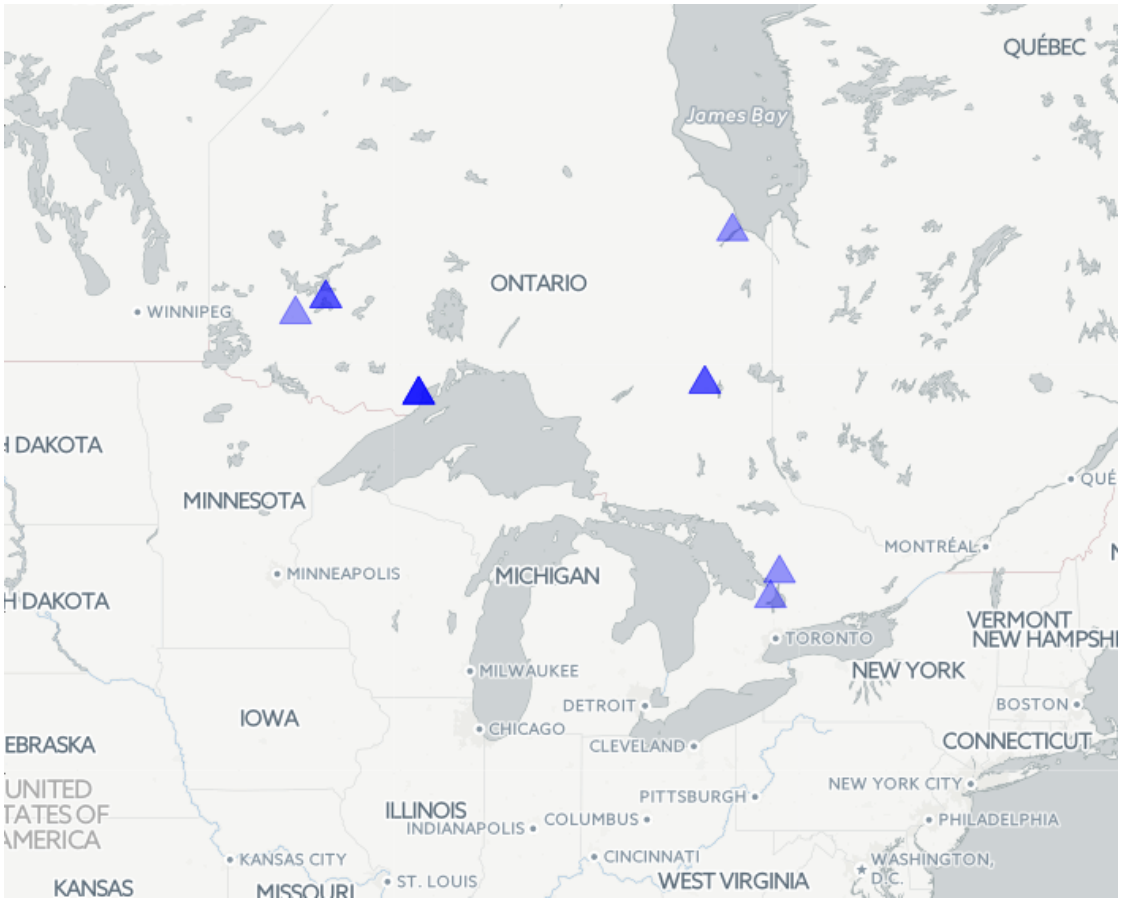

Figure B.3: The aircraft chosen in scenario 1-1.

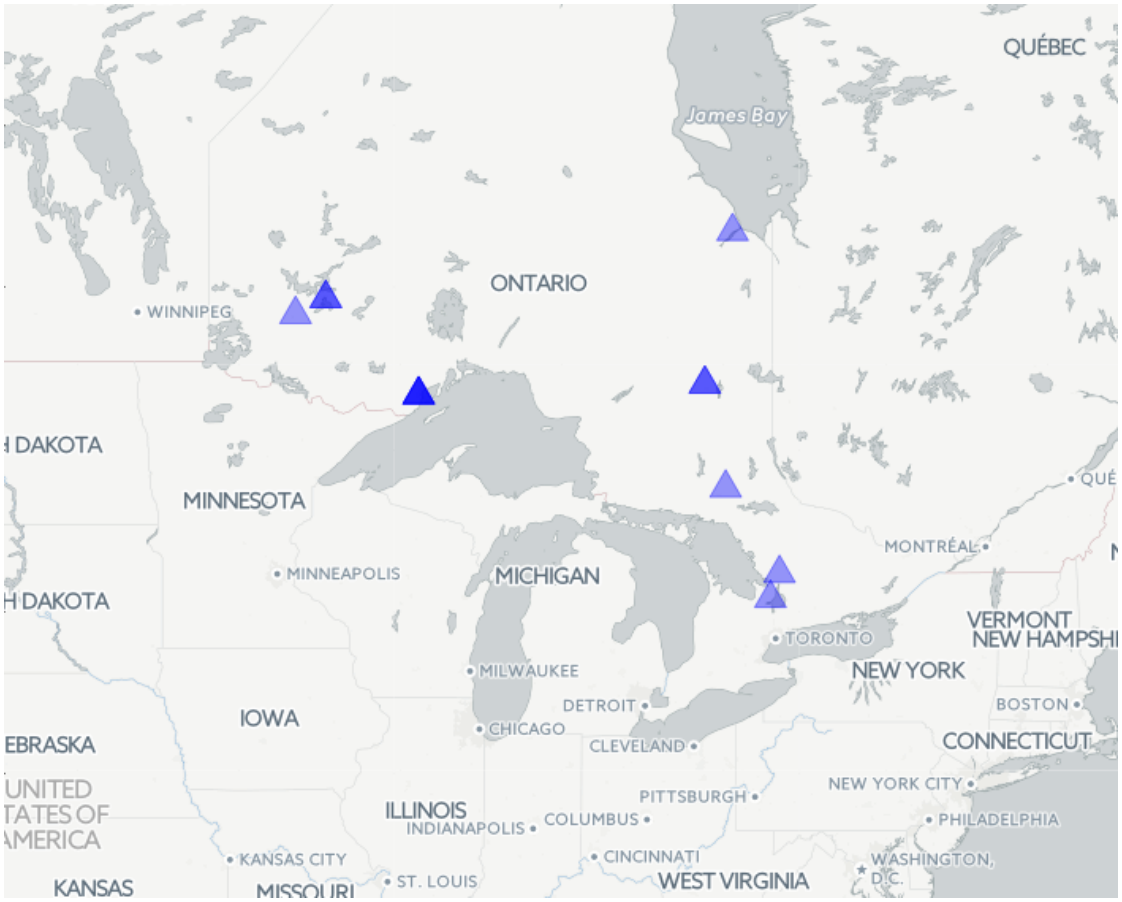

Figure B.4: The aircraft chosen in scenario 1-2.

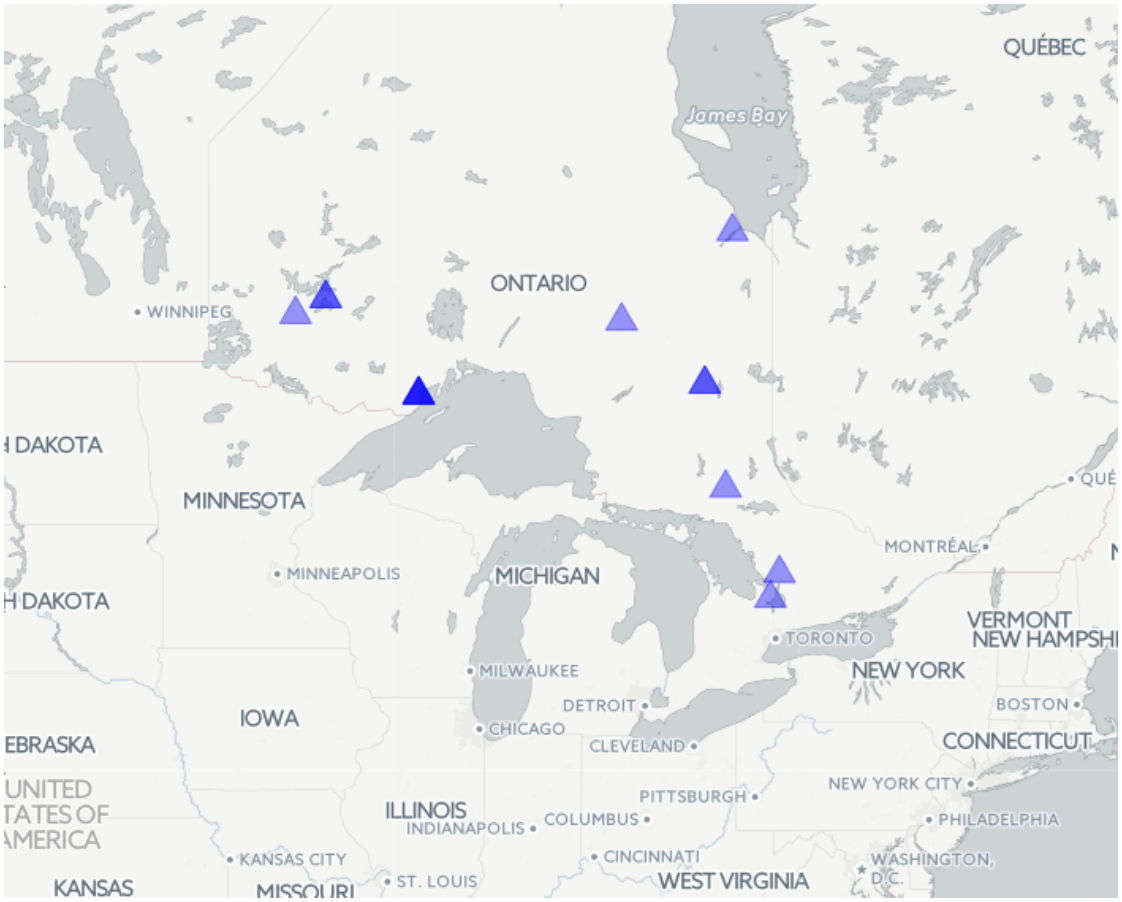

Figure B.5: The aircraft chosen in scenario 1-3.

APPENDIX C

BCEHS SIMULATION

C.1 Aircraft Utilization

The following tables detail the utilization of each aircraft under selected policies, as measured by the fraction of time the aircraft was available. Since we ignored duty day constraints, each aircraft is available 24 hours a day. The title of each table is the policy being detailed; see Table 4.2 for details.

GNP

Transport	Total	Active	Patient
Cessna Citation	0.36	0.20	0.23
Beechcraft King Air	0.11	0.07	0.07
Beechcraft King Air 1	0.10	0.07	0.07
Beechcraft King Air 2	0.09	0.06	0.06
Beechcraft King Air	0.07	0.05	0.05
Sikorsky S76	0.05	0.05	0.05
Sikorsky S76	0.03	0.03	0.03
Sikorsky S76	0.03	0.03	0.03
Bell 412	0.03	0.02	0.02

Table C.1: The time flown by each aircraft, expressed as a percentage of the total time the aircraft is in service. Recall that aircraft did not have restricted duty days. The active time represents time with at least one patient onboard. The patient time represents the total time spent by patients onboard, counting each patient separately. Note that the RW aircraft (the Sikorsky S76 and Bell 412) have very low utilization, likely due to the omission of most urgent calls as discussed in Section 4.2.

GNP R

Transport	Total	Active	Patient
Cessna Citation	0.27	0.13	0.14
Beechcraft King Air	0.07	0.03	0.04
Beechcraft King Air 1	0.17	0.08	0.08
Beechcraft King Air 2	0.04	0.02	0.02
Beechcraft King Air	0.38	0.18	0.19
Sikorsky S76	0.01	0.01	0.01
Sikorsky S76	0.00	0.00	0.00
Sikorsky S76	0.35	0.17	0.17
Bell 412	0.05	0.03	0.03

Table C.2: The time flown by each aircraft, expressed as a percentage of the total time the aircraft is in service. Recall that aircraft did not have restricted duty days. The active time represents time with at least one patient onboard. The patient time represents the total time spent by patients onboard, counting each patient separately.

GP

Transport	Total	Active	Patient
Cessna Citation	0.37	0.22	0.26
Beechcraft King Air	0.13	0.09	0.10
Beechcraft King Air 1	0.10	0.07	0.07
Beechcraft King Air 2	0.09	0.06	0.07
Beechcraft King Air	0.06	0.04	0.04
Sikorsky S76	0.03	0.03	0.03
Sikorsky S76	0.03	0.02	0.02
Sikorsky S76	0.04	0.03	0.03
Bell 412	0.03	0.03	0.03

Table C.3: The time flown by each aircraft, expressed as a percentage of the total time the aircraft is in service. Recall that aircraft did not have restricted duty days. The active time represents time with at least one patient onboard. The patient time represents the total time spent by patients onboard, counting each patient separately.

GP R

Transport	Total	Active	Patient
Cessna Citation	0.27	0.13	0.15
Beechcraft King Air	0.07	0.04	0.04
Beechcraft King Air 1	0.17	0.08	0.08
Beechcraft King Air 2	0.04	0.02	0.02
Beechcraft King Air	0.38	0.19	0.20
Sikorsky S76	0.01	0.01	0.01
Sikorsky S76	0.00	0.00	0.00
Sikorsky S76	0.35	0.17	0.17
Bell 412	0.05	0.03	0.03

Table C.4: The time flown by each aircraft, expressed as a percentage of the total time the aircraft is in service. Recall that aircraft did not have restricted duty days. The active time represents time with at least one patient onboard. The patient time represents the total time spent by patients onboard, counting each patient separately.

T9 (GP, GP)

Transport	Total	Active	Patient
Cessna Citation	0.39	0.23	0.28
Beechcraft King Air	0.12	0.08	0.08
Beechcraft King Air 1	0.10	0.07	0.07
Beechcraft King Air 2	0.09	0.06	0.06
Beechcraft King Air	0.05	0.03	0.04
Sikorsky S76	0.05	0.04	0.04
Sikorsky S76	0.02	0.02	0.02
Sikorsky S76	0.02	0.02	0.02
Bell 412	0.02	0.02	0.02

Table C.5: The time flown by each aircraft, expressed as a percentage of the total time the aircraft is in service. Recall that aircraft did not have restricted duty days. The active time represents time with at least one patient onboard. The patient time represents the total time spent by patients onboard, counting each patient separately.

T9 (GP, GP) R

Transport	Total	Active	Patient
Cessna Citation	0.39	0.23	0.28
Beechcraft King Air	0.12	0.08	0.08
Beechcraft King Air 1	0.10	0.07	0.07
Beechcraft King Air 2	0.09	0.06	0.06
Beechcraft King Air	0.05	0.03	0.04
Sikorsky S76	0.05	0.04	0.04
Sikorsky S76	0.02	0.02	0.02
Sikorsky S76	0.02	0.02	0.02
Bell 412	0.02	0.02	0.02

Table C.6: The time flown by each aircraft, expressed as a percentage of the total time the aircraft is in service. Recall that aircraft did not have restricted duty days. The active time represents time with at least one patient onboard. The patient time represents the total time spent by patients onboard, counting each patient separately.

BIBLIOGRAPHY

[1] Amazon flex. https://flex.amazon.com. Accessed on December 2, 2015.

[2] Bcehs. http://www.bcehs.ca/about. Accessed on December 6, 2016.

[3] Ornge. http://www.ornge.ca/AboutOrnge. Accessed on November 9, 2016.

[4] Stamen Design LLC. https://stamen.com/about.

[5] Uber rush. https://rush.uber.com/how-it-works. Accessed on December 2, 2015.

[6] Giorgio Ausiello, Esteban Feuerstein, Stefano Leonardi, Leen Stougie, and Maurizio Talamo. Algorithms for the on-line travelling salesman. *Algorithmica*, 29(4):560–581, 2001.

[7] Egon Balas and Manfred W Padberg. Set partitioning: A survey. *SIAM review*, 18(4):710–760, 1976.

[8] Dimitris Bertsimas and John N Tsitsiklis. *Introduction to linear optimization*, volume 6. Athena Scientific Belmont, MA, 1997.

[9] Dimitris J Bertsimas, Patrick Jaillet, and Amedeo R Odoni. A priori optimization. *Operations Research*, 38(6):1019–1033, 1990.

[10] John R Birge and Francois Louveaux. *Introduction to stochastic programming*. Springer Science & Business Media, 2011.

[11] Allan Borodin and Ran El-Yaniv. *Online computation and competitive analysis*. cambridge university press, 1998.

[12] Timothy Carnes. Approximation algorithms via the primal-dual schema: Applications of the simple dual-ascent method to problems from logistics. 2010.

[13] Timothy A Carnes, Shane G Henderson, David B Shmoys, Mahvareh Ahghari, and Russell D MacDonald. Mathematical programming guides air-ambulance routing at ornge. *Interfaces*, 43(3):232–239, 2013.

[14] Jean-François Cordeau and Gilbert Laporte. The dial-a-ride problem: models and algorithms. *Annals of Operations Research*, 153(1):29–46, 2007.

[15] Gurobi Optimization, Inc. Gurobi optimizer reference manual, 2016.

[16] Patrick Jaillet. A priori solution of a traveling salesman problem in which a random subset of the customers are visited. *Operations research*, 36(6):929–936, 1988.

[17] Patrick Jaillet and Michael R Wagner. Online vehicle routing problems: A survey. In *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 221–237. Springer, 2008.

[18] Sujin Kim, Raghu Pasupathy, and Shane G Henderson. A guide to sample average approximation. In *Handbook of simulation optimization*, pages 207–243. Springer, 2015.

[19] Gilbert Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(3):345–358, 1992.

[20] Linn I Sennott. *Stochastic dynamic programming and the control of queueing systems*, volume 504. John Wiley & Sons, 2009.

[21] David Shmoys and Kunal Talwar. A constant approximation algorithm for the a priori traveling salesman problem. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 331–343. Springer, 2008.

[22] Eric W Weisstein. Lambert w-function. 2002.

[23] David P Williamson and David B Shmoys. *The design of approximation algorithms*. Cambridge university press, 2011.