UNIVERSITY OF CALIFORNIA
Santa Barbara

Task Allocation and Vehicle Routing in Dynamic Environments

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Mechanical Engineering

by

Stephen Leslie Smith

September 2009

The dissertation of Stephen Leslie Smith is approved.

Mustafa Khammash

João P. Hespanha

Bassam Bamieh

Francesco Bullo, Committee Chair

University of California, Santa Barbara

June 2009

Acknowledgments

First of all, I would like to thank Francesco Bullo for being an extraordinary advisor. I have learned a great deal from his guidance, his enthusiasm and his approach to research. Among the many lessons I have learned from him, one which sticks out and has affected me greatly is the simple concept that research should be fun.

I would like to thank Emilio Frazzoli for providing me with the opportunity to visit the Massachusetts Institute of Technology. It was a great experience to work in his group and get a different perspective on my research. I also want to thank Marco Pavone for the great collaboration. It was a pleasure working with him and I hope that we will be able to work together more in the future.

I want to thank George J. Pappas and Ali Jadbabaie for providing me with the opportunity to visit the GRASP Lab at the University of Pennsylvania. In addition, I want to thank Michael Zavlanos for making my stay there comfortable and helping me to get to know the research group and the city.

I want to thank João P. Hespanha for providing excellent insight into some of my research problems I have been working on. I would like to thank Shaunak D. Bopardikar for an excellent collaboration and for being a good friend. I want to thank Karl J. Obermeyer for some great discussions about research and advice on color choices, and John Enright for showing me around Boston during my visit. I also want to thank Bruce A. Francis and Mireille E. Broucke for advising me in the work that led to this thesis, and for the encouragement during my transition from the University of Toronto to UCSB.

Finally, I would like to thank all of the members of the research group, my friends at UCSB and elsewhere, and most of all, my family for supporting me during my seemingly endless years of education.

Abstract

Task Allocation and Vehicle Routing in Dynamic Environments

by

Stephen Leslie Smith

Autonomous vehicles and sensors have the potential to transform many aspects of societal infrastructure, from transportation networks and assisted living, to emergency response systems and military operations. In fact, the transformation is already occurring in areas such as reconnaissance and environmental monitoring. However, in these applications autonomous vehicles are typically deployed in small numbers, tightly coupled with human control. To realize the full potential of such systems there is a required shift towards large groups of networked and highly autonomous vehicles, capable of performing complex and evolving tasks. This shift calls for vehicles that can adapt to dynamic environments, utilizing newly acquired information to re-allocate resources, and re-plan routes.

This thesis addresses problems in distributed task allocation and in dynamic vehicle routing. In task allocation we consider a target assignment problem in which a group of vehicles must divide a set of targets (tasks) among themselves. In dynamic vehicle routing—where vehicles must complete spatially distributed tasks that arrive sequentially in time—we consider several problems. First, we consider a problem in which the vehicles have different capabilities, and each task requires a team of vehicles for its completion. Second, we consider a problem in which tasks have different levels of urgency, and thus the vehicles must prioritize the tasks, completing urgent tasks with minimal delay while simultaneously considering and completing less urgent tasks. Finally, we consider a problem in which task locations are non-stationary, and a variation wherein a vehicle must guard a boundary from approaching targets. Our technical approach to each of these problems follows the same basic steps. We show that the problem exhibits an underlying structure that can be exploited to determine fundamental limits on the achievable performance. Then, we design novel and provably efficient algorithms for solving the problem. The solutions combine aspects of combinatorial optimization, stochastic processes, and distributed control.

Table of Contents

List of Figures

List of Tables

List of Algorithms

Chapter One

Introduction

Systems of networked autonomous vehicles[1] have the potential to transform many aspects of our societal infrastructure, from transportation networks and assisted living, to emergency response systems and military operations. In fact, the transformation is already occurring in several areas. One area is environmental monitoring, where autonomous vehicles are currently being deployed for ocean sampling (see Leonard et al., 2007), and for estimating the boundaries of wildfires (see Casbeer et al., 2006) or oil spills (see Clark and Fierro, 2007). Another area is in reconnaissance, where there have been significant advances in using autonomous vehicles to gather information in remote or dangerous environments (see Beard et al., 2006). However, in these applications, autonomous vehicles are typically deployed in small numbers, tightly coupled with human control. To realize the full potential of such systems, there is a required shift towards large groups of networked and highly autonomous vehicles, capable of performing complex and evolving tasks. This shift calls for vehicles that can adapt to dynamic environments, utilizing newly acquired information to re-allocate resources, and re-plan routes; that is, vehicles that can dynamically perform *task allocation* and *vehicle routing*. It is these two problems—task allocation and vehicle routing—that will be the focus of this thesis.

Broadly speaking, task allocation is concerned with the following problem:

> Given a group of a vehicles and a set of tasks, decide which vehicle(s) should perform which task(s).

An illustration of task allocation is shown in Figure 1.1. We are typically thinking of tasks that consist of a location (or set of locations) and a service to be provided at that location. In the settings of environmental monitoring and reconnaissance, a task may consist of a location at which an unmanned aerial vehicle is needed to take a picture or some video footage. After determining a task allocation, the problem of vehicle routing is as follows:

> Given an allocation of tasks to vehicles, determine routes that allow each vehicle to complete its tasks.

[1]We use the word "vehicle" to refer to any form of robotic agent capable of motion.

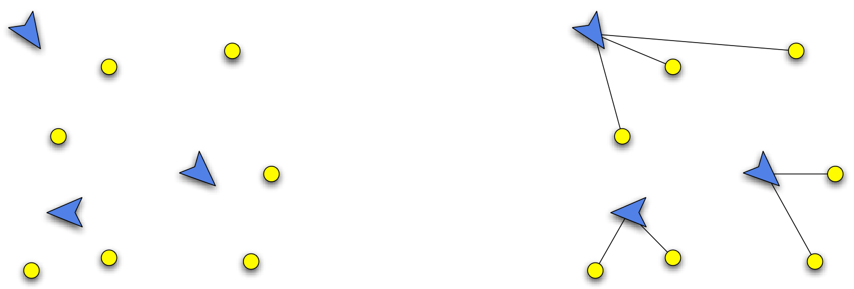


Figure 1.1: An illustration of task allocation. Vehicles are denoted by chevrons, and task locations by disks. The left-figure shows a group of vehicles and a set of tasks and the right-figure shows an allocation of tasks to vehicles.

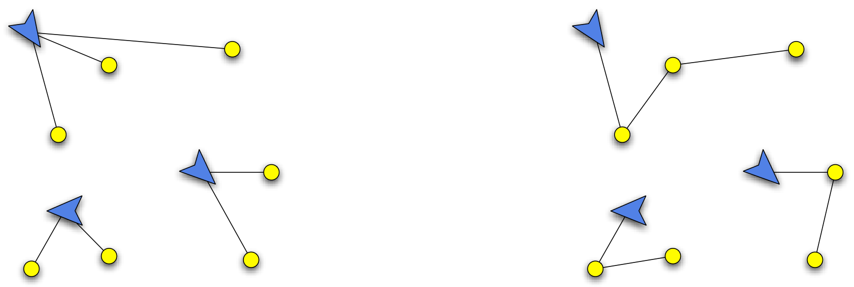


Figure 1.2: An illustration of vehicle routing. Vehicles are denoted by chevrons, and task locations by disks. The left-figure shows a task allocation and the right-figure shows a set of routes for completing the tasks.

An illustration of vehicle routing is shown in Figure 1.2. In task allocation and in vehicle routing we are often interested in minimizing a metric, or cost function, such as the amount of time until all tasks are completed. Notice, however, that the problems of allocation and routing are interconnected; to equitably divide tasks among vehicles, one needs to know the amount of time it will take for a vehicle to complete a set of tasks. However, there are many important applications for which the problems can be decoupled, and thus task allocation and vehicle routing can be studied independently.

As stated above, many future applications call for highly autonomous vehicles that are capable of performing complex and evolving tasks. Thus, we would like to address the problems of task allocation and vehicle routing in the distributed setting, where allocation and routing decisions are made by the vehicles rather than a supervisor. In addition, we are interested in dynamic environments, where new task information becomes available in real-time. Thus, the focus of this thesis is on the areas of distributed task allocation and dynamic vehicle routing. In the following sections we will review the existing literature in allocation and routing and describe the contributions of this thesis.

1.1 Literature Synopsis

In this section we give an overview of the existing literature in the areas of task allocation and vehicle routing. This will allow for a more concrete statement of the contributions of the thesis. However, each chapter of the thesis deals with a specific problem in the area. Thus, at the beginning of each chapter we review the literature related directly to the chapter's topic.

1.1.1 Centralized task allocation

In a centralized setting, there is a single processor (supervisor) is given complete information of the vehicles and tasks, and must determine an efficient allocation. In many cases such a problem can be cast in the combinatorial optimization framework. For example, if there are an equal number of vehicles and tasks, and the goal is simply to assign one vehicle to each task, then the problem is known as the *maximum matching problem* (Korte and Vygen, 2005). If one wishes to capture the notion of efficiency, then we many be interested in finding a maximum matching that minimizes a cost function. If the cost function is the sum of distances from each vehicle to its assigned task, then the problem is known as the *sum assignment problem* (Korte and Vygen, 2005). Another choice of cost function is to minimize the maximum distance between vehicles and their assigned targets. This problem is commonly referred to as the *bottleneck assignment problem* (Burkard, 2002). There exist efficient polynomial time algorithms for the solution of all these problems (Hopcroft and Karp, 1973), (Kuhn, 1955), (Burkard, 2002). The *auction algorithm* (Bertsekas and Tsitsiklis, 1997) also provides a solutions to the sum assignment problem, and has the advantage that can be implemented on a parallel processing machine.

1.1.2 Decentralized task allocation

In decentralized task allocation the goal is generally to assign vehicles to spatially distributed tasks while maximizing the "score" of the mission. Gerkey and Mataric (2004) give a taxonomy of task allocation problems, dividing problems into groups based on, among other things, the number of tasks a vehicle can execute, and the number of vehicles required for a task. As an example, Parker (1998) designed the ALLIANCE architecture for fault-tolerant multi-vehicle coordination, which provides a method for task allocation. In papers such as (Jin. et al., 2003; Schumacher et al., 2003; Godwin et al., 2006; Alighanbari and How, 2008; Brunet et al., 2008), advanced heuristic methods are developed, and their effectiveness is demonstrated through simulation or real-world implementation. Bethke et al. (2008) provide a nice review of task allocation for unmanned aerial vehicles, with a focus on hardware implementation. There have been other interesting approaches to task allocation. For example, Moore and Passino (2007) adapt the auction algorithm to solve a task allocation problem in the presence of communication delays. Castañón and Wu (2003) study the problem of dynamically reassigning vehicles as new tasks arrive and old tasks expire. Arslan et al. (2007) formulate a task allocation problem as a multi-player game and seek to optimize a global utility. Finally, Zavlanos and Pappas (2008) develop a task allocation algorithm based on hybrid systems tools.

1.1.3 Classic vehicle routing

Vehicle routing problems are concerned with planning optimal vehicle routes for providing service to a given set of customers. The routes are planned with complete information of the customers, and thus the optimization is static, and typically combinatorial (Toth and Vigo, 2001). Arguably, the most famous vehicle routing problem is the traveling salesperson problem (TSP) (see Applegate et al., 2006) [2]. In the TSP, there are a fixed number of customer locations (or tasks), and the goal is to find the shortest tour which visits each location exactly once, and returns to the starting point. A more thorough review of the results on the TSP and related problems is given in Chapter 2.

Other interesting vehicle routing problems include the TSP for the Dubins vehicle (Savla, Frazzoli and Bullo, 2008) and for the double integrator (Savla et al., 2009); the deadline-TSP, where each location has a deadline before which it must be visited (Bansal et al., 2004); vehicle routing with time-windows (Solomon, 1987), where locations may also have a start time before which they cannot be visited; the Dial-a-ride problem (Psaraftis, 1980) and its many variations (Cordeau and Laporte, 2003), where each demand for service consists of a pick-up location and a destination; orienteering problems (Blum et al., 2007) where there is a prize at each location, and the vehicle must maximize the value of prizes collected in a fixed time horizon; and, the minimum latency problem (Blum et al., 1994), where the vehicle seeks to minimize the sum of service latencies for each demand. An in-depth characterization of vehicle routing problems is given by Toth and Vigo (2001) and Golden et al. (2008).

Recently, researchers have looked at the Euclidean TSP with moving objects. In (Chalasani and Motwani, 1999) the authors consider objects moving on straight lines in the y-direction and focus on the case when the objects are slower than

[2]This problem is also commonly referred to as the traveling salesman problem.

the vehicle and when the vehicle moves parallel to the x- or y-axis. The same problem is studied in (Hammar and Nilsson, 2002), but with arbitrary vehicle motion, and it is called the translational TSP. The authors of (Hammar and Nilsson, 2002) propose a polynomial-time approximation scheme to catch all objects in minimum time. Another variation of this problem with object motion on piece-wise straight line paths, and with different but finite object speeds has been addressed in (Asahiro et al., 2008). Other variants of the moving TSP in which the points are allowed to move in different directions have been addressed in (Helvig et al., 2003).

1.1.4 Dynamic vehicle routing

In contrast to the classic vehicle routing problems discussed above, dynamic vehicle routing (DVR) considers scenarios in which not all customer information is known *a priori*, and thus routes must be re-planned as new customer information becomes available (Powell et al., 1995; Larsen, 2000). These types of problems have been studied in the context of urban operations research for many years (Larson and Odoni, 1981). Dynamic vehicle routing addresses problems where autonomous vehicles are deployed in complex and uncertain environments. Because of this, DVR provides a model for applications such as environmental monitoring, surveillance and perimeter defence, where the vehicles must re-plan their motion as they acquire information on their surroundings. In addition, DVR models scenarios where remote operators may add tasks to, or remove tasks from, the vehicles' mission in real-time.

Dynamic traveling repairperson problem: The first DVR problem was arguably the dynamic traveling repairperson Problem (DTRP) (Psaraftis, 1988; Bertsimas and van Ryzin, 1991), in which customers (frequently called *demands for service* or simply *demands*), arrive sequentially in a region, and a service vehicle seeks to serve them by reaching each demand location and spending some amount of on-site service time. The goal is for the service vehicle to minimize the expected delay between a demands' arrival, and its service completion. The DTRP was first introduced on a graph by Psaraftis (1988), and for the Euclidean Plane by Bertsimas and van Ryzin (1991). In Bertsimas and van Ryzin (1991), the authors study the DTRP for a single service vehicle, a Poisson demand arrival process, and a uniform spatial density of demand arrivals. Optimal DTRP policies are proposed in light load conditions (i.e., when the fraction of time the service vehicles spends performing on-site service approaches zero) and policies within a constant factor of the optimal are proposed for heavy load conditions (i.e., when the fraction of time the service vehicles spend performing on-site service approaches one). In Bertsimas and van Ryzin (1993*a*) the DTRP was extended to multiple vehicles with finite service capacity (i.e., vehicles must go to a central depot to recharge). In addition, an improved policy was given for the heavy load conditions. Xu (1995, page 23) developed a stronger policy independent lower bound for heavy load conditions, which showed that the improved policy of Bertsimas and van Ryzin (1993*a*) is optimal in heavy load conditions. In Bertsimas and van Ryzin (1993*b*) the DTRP was generalized to nonuniform spatial density and any renewal arrival process. Finally, Papastavrou (1996) developed a single vehicle routing policy which provides near optimal performance in both light and heavy load conditions.

Relation to spatial queueing: The DTRP can be thought of as a spatial queuing problem. Instead of customers arriving at a fixed queuing location, they are distributed throughout space, and thus service is composed of both a travel time and an on-site service time. Because of this relation, the analysis of the DTRP relies heavily on results in classical queueing theory (Kleinrock, 1975, 1976). The DTRP is also related to polling system problems (Takagi, 1986), where customers arrive only at fixed locations in space.

Online algorithms and DVR In the computer science community there has been considerable interest in online algorithms (Borodin and El-Yaniv, 1998). An online algorithm is one that operates based on input information given up to the current time. Thus, these algorithms are designed to operate in scenarios where the entire input is not known at the outset, and new pieces of the input should be incorporated as they become available. Typically, the performance of an online algorithm is characterized by comparing it to the performance of a corresponding offline algorithm (i.e., an algorithm that has *a priori* knowledge of the entire input).

Dynamic vehicle routing problems have been studied from the perspective of online algorithms. A well studied online problem is called the k-server problem. This problem is equivalent to the DTRP, but with the added constraint that demands must be served in the order in which they arrive (Manasse et al., 1990; Koutsoupias and Papadimitriou, 1995). More recently the DTRP has been studied in the online framework under the name of the online traveling repairman problem (Krumke et al., 2003; Jaillet and Wagner, 2006). For this problem, constant factor competitive ratios have been obtained, and the online algorithms generally utilize solvers for vehicle routing with time-windows (Solomon, 1987).

Decentralized DTRP: Most policies designed for the DTRP are centralized; a central dispatcher notifies each vehicle of its route. Recently, there has been research on solving the DTRP in a distributed manner, where the vehicles divide the demands amongst themselves. The problem was addressed for uniform spatial density by Frazzoli and Bullo (2004) and for nonuniform spatial density by Pavone, Frazzoli and Bullo (2009). The idea behind this work is that vehicles must agree on an equitable partition (Pavone, Arsie, Frazzoli and Bullo, 2008) of the environment, such that the workload (i.e., the expected amount of time required to service all customers that arrive over a fixed time interval), of each region is equal. One vehicle then works in each region of the partition. Enright and Frazzoli (2006) studied a variation of the DTRP where the vehicles have a limited sensor range, and can only see customers within that range.

Dynamic pick-up and delivery problem: There have been several other variations on DVR problems, and a thorough review is given by Larsen (2000). One example is the dynamic pick-up and delivery problem (DPDP), which is a dynamic version of the dial-a-ride problem (Psaraftis, 1980). The DPDP was introduced for a single vehicle with a fixed capacity by Swihart and Papastavrou (1999), and provides a model for a taxi company, where service requests are generated in real-time, and consist of a pick-up location and a destination. Waisanen et al. (2008), considered a variation of the DPDP motivated by wireless networks, where the service vehicle can be thought of as a mobile relay, picking up messages from source locations, and delivering them to their destinations. For many of these problems, policies have been proposed which provide expected customer delays that are within a constant factor of this lower bound.

Recent DVR variations: Recently, there has been an increased interest in DVR among researchers in robotic motion planning, as it provides a powerful method for completing spatially distributed tasks that are generated in real-time. Some interesting research includes DTRP problems for the Dubins vehicle (Savla, Frazzoli and Bullo, 2008; Enright et al., 2009), where the service vehicle's motion is subject to a nonholonomic constraint. Pavone, Bisnik, Frazzoli and Isler (2008) study a DVR problem with impatient customers, where demands for service leave the system if they are not serviced within a certain time window. Another variant is the dynamic traveling repairperson problem for dynamical systems (Itani et al., 2008). A related problem is dynamic data harvesting (Le Ny, Dahleh, Feron and Frazzoli, 2008), where a service vehicle must collect data from wireless data stations, and its signal strength for a given station (and thus service rate) depends on distance from the service vehicle to the station. Savla, Temple and Frazzoli (2008) study a variation of the DTRP where the service time of a particular demand is dictated by a human operator. This work models situations where there is a "human-in-the-loop." Le Ny, Dahleh and Feron (2008) study a problem where demands are modeled as a two-state Markov chain, and they must be periodically visited by a service vehicle while in one of the two states. Another related problems is a dynamic variant of the assignment problem (Spivey and Powell, 2004). A discussion of other recent DVR variations is given by Golden et al. (2008).

1.2 Contributions of the Thesis

In this thesis we look at a problem in distributed task allocation, and several problems in the area of dynamic vehicle routing. Chapter 3 looks at a task allocation problem called the target assignment problem. Chapters 4, 5, 6, and 7 look at different problems within the area of dynamic vehicle routing.

Technical approach: Fundamental theory and algorithms: Our technical approach for each problem relies on developing a model which captures the essential features of the problem, and then following the three basic steps. They can be summarized as follows:

(i) For each problem we identify **underlying problem structure**. This typically consists of intrinsic regimes in the space of problem parameters.

(ii) Leveraging the problem structure, we determine **fundamental limits on the achievable performance** of any algorithm for solving the problem of interest.

(iii) Utilizing the fundamental limits and problem structure, we design **provably efficient algorithms**. Generally, we seek algorithms that are guaranteed to perform within a constant factor of the optimal algorithm.

In using this approach will are able to not only learn about the limits on achievable performance, but also provide provably efficient solutions to each problem.

Contributions of each chapter: The organization of the thesis, and the contributions of each chapter can be summarized as follows.

Chapter 2: In this chapter we review some results in combinatorial optimization, including the TSP problem and related variations, matching problems, and vertex coloring. The results will be applied in the subsequent chapters for both the design of algorithms and their analysis.

Chapter 3: In this chapter we study a problem in distributed task allocation which consists of an equal number of vehicles and distinct target locations dispersed in an environment. Each vehicle has a limited communication range and either (1) knowledge of every target position, or (2) a finite-range sensor capable of acquiring target positions and no a priori knowledge of target positions. The problem is to design a distributed algorithm with which the vehicles divide the targets among themselves and, simultaneously, move to their unique target. We evaluate an algorithm's performance by characterizing its worst-case asymptotic time to complete the target assignment; that is the completion time as the number of vehicles (and targets) increases, and the size of the environment scales to accommodate them. We introduce the intuitive class of *monotonic algorithms*, and give a lower bound on its worst-case completion time. We design and analyze two algorithms within this class: the ETSP Assgmt algorithm which works under assumption (1), and the Grid Assgmt algorithm which works under either assumption (1) or (2). In "sparse environments," where communication is infrequent, the ETSP Assgmt algorithm is within a constant factor of the optimal monotonic algorithm for worst-case initial conditions. In "dense environments," where communication is more prevalent, the Grid Assgmt algorithm is within a constant factor of the optimal monotonic algorithm for worst-case initial conditions. In addition we characterize the performance of the Grid Assgmt algorithm for uniformly distributed targets and vehicles, and for the case when there are more vehicles than targets.

The work in this chapter is based on the journal article (Smith and Bullo, 2009*b*) and the preliminary conference papers (Smith and Bullo, 2007*a*) and (Smith and Bullo, 2007*b*).

Chapter 4: In this chapter we introduce a DVR problem called the dynamic team forming problem (DTFP). The DTFP for a heterogeneous group of vehicles is described as follows. Each vehicle is capable of providing a specific service. Tasks arrive sequentially over time, assume random locations in the environment, and require several different services. A task is completed when a team of vehicles travels to the task location and provides the required services. The goal is to minimize the expected *delay* between a task's arrival and its completion. We restrict our attention to unbiased policies for the DTFP, i.e., policies for which the expected delay is the same for all tasks. We introduce three intuitive policies, and in certain asymptotic regimes we analyze their *delay* as a function of the arrival rate of tasks (or *throughput*). For each policy we show that there is a broad class of system parameters for which the policy's performance is within a constant factor of the optimal.

The work in this chapter is based on the journal article (Smith and Bullo, 2009*a*) and the preliminary conference paper (Smith and Bullo, 2008).

Chapter 5: In Chapter 4 we restricted our attention to unbiased policies. In this chapter we explore the concept of biasing service by introducing a DVR problem in which there are multiple vehicles and multiple priority classes of service demands. Service demands of each priority class arrive in the environment randomly over time and require a random amount of on-site service that is characteristic of the class. The quality of service provided to each class is given by the expected delay between the arrival of a demand in the class, and that demand's service completion. The goal is to design a routing policy for the service vehicles which minimizes a convex combination of the delays for each class. First, we provide a lower bound on the achievable values of the convex combination of delays. Then, we propose a novel routing policy and analyze its performance under heavy load conditions (i.e., when the fraction of time the service vehicles spend performing on-site service approaches one). The policy performs within a constant factor of the lower bound, where the constant depends only on the number of classes, and is independent of the number of vehicles, the arrival rates of demands, the on-site service times, and the convex combination coefficients.

The work in this chapter is based on the journal article (Smith, Pavone, Bullo and Frazzoli, 2009) and the preliminary conference papers (Smith et al., 2008) and (Pavone, Smith, Bullo and Frazzoli, 2009).

Chapter 6: In this chapter we introduce a DVR problem in which demands arrive stochastically on a line segment, and upon arrival, move with a fixed velocity perpendicular to the segment. We design a receding horizon service policy for a vehicle with speed greater than that of the demands, based on the translational minimum Hamiltonian path (TMHP). We consider Poisson demand arrivals, uniformly distributed along the segment. For a fixed segment width and fixed vehicle speed, the problem is governed by two parameters; the demand speed and the arrival rate. We establish a necessary condition on the arrival rate in terms of the demand speed for the existence of any stabilizing policy. We derive a sufficient condition on the arrival rate in terms of the demand speed that ensures stability of the TMHP-Based policy. When the demand speed tends to the vehicle speed, every stabilizing policy must service the demands in the first-come-first-served (FCFS) order; and of all such policies, the TMHP-Based policy minimizes the expected time before a demand is serviced. When the demand speed tends to zero, the sufficient condition on the arrival rate for stability of the TMHP-Based policy is within a constant factor of the necessary condition for stability of any policy. Finally, when the arrival rate tends to zero for a fixed demand speed, the TMHP-Based policy minimizes the expected time before a demand is serviced. We numerically validate our analysis and empirically characterize the region in the parameter space for which the TMHP-Based policy is stable.

The work in this chapter is based on the journal article (Bopardikar, Smith, Bullo and Hespanha, 2009*a*) and the preliminary conference papers (Bopardikar, Smith, Bullo and Hespanha, 2009*b*) and (Smith, Bopardikar, Bullo and Hespanha, 2009).

Chapter 7: In this chapter we introduce a DVR problem in which a service vehicle seeks to defend a deadline from dynamically arriving mobile targets. The environment is a rectangle and the deadline is one of its edges. Targets arrive continuously over time on the edge opposite the deadline, and move towards the deadline at a fixed speed. The goal for the vehicle is to maximize the fraction of targets that are captured before reaching the boundary. We consider two cases; when the service vehicle is faster than the targets; and, when the service vehicle is slower than the targets. In the first case we develop a novel vehicle policy based on computing longest paths in a directed acyclic graph. We give a lower bound on capture fraction of the policy and show that it is optimal when the distance between the target arrival edge and boundary becomes very large. We present numerical results which suggest that the policy performs very near the optimal even for moderate distances between the arrival edge and the boundary. In the second case, when the targets are slower than the vehicle, we propose a policy based on servicing fractions of the translational minimum Hamiltonian path. In the limit of low target speed and high arrival rate, the capture fraction of this policy is within a small constant factor of the optimal.

The work in this chapter is based on the recent conference paper (Smith, Bopardikar and Bullo, 2009).

Chapter 8: This chapter outlines some broad directions for future research in the areas of dynamic and distributed task allocation and vehicle routing, and concludes the thesis.

Chapter Two

Preliminaries in Combinatorics and Queueing

In this chapter we review results on the shortest paths and tours through sets of points and on some basics of queueing theory. We begin by introducing some notation which will be used consistently throughout the remaining chapters.

2.1 Notation

We let $\mathbb{R}$, $\mathbb{R}_{\geq 0}$ and $\mathbb{N}$ denote the set of real numbers, the set of non-negative real numbers, and the set of positive integers, respectively. Given a finite set A, we let $|A|$ denote its cardinality, and given an infinite set $A \subset \mathbb{R}^d$, $d \geq 2$ we let $|A|$ denote its d-dimensional volume. For two functions $f, g : \mathbb{N} \to \mathbb{R}_{>0}$, we write $f(n) \in O(g)$ (respectively, $f(n) \in \Omega(g)$) if there exist $N \in \mathbb{N}$ and $c \in \mathbb{R}_{>0}$ such that $f(n) \leq cg(n)$ for all $n \geq N$ (respectively, $f(n) \geq cg(n)$ for all $n \geq N$). If $f(n) \in O(g)$ and $f(n) \in \Omega(g)$, then we say $f(n) \in \Theta(g)$. We say that event $A(n)$ occurs *with high probability* (w.h.p.) if the probability of $A(n)$ occurring tends to one as $n \to +\infty$.

2.2 Combinatorics

In this section we review results combinatorial optimization which will be useful in the algorithm designs in subsequent chapters.

2.2.1 Basic Graph Definitions

A **directed graph** $G = (V(G), E(G))$ consists of a set of **vertices** $V(G)$ and a set of directed **edges** $E(G) \subset V(G) \times V(G)$. An edge $(v, w) \in E(G)$ is directed from vertex v to vertex w. A subgraph of a graph $G = (V(G), E(G))$ is a graph $H = (V(H), E(H))$ such that $V(H) \subset V(G)$ and $E(H) \subset E(G)$.

A graph is **undirected** if $(v, w) \in E(G)$ implies that $(w, v) \in E(G)$, and in this case we write $\{v, w\} \in E(G)$. For an undirected graph an edge $\{v, w\} \in E$ is **incident** to v and w, and v and w are **neighbors**. The *degree* of $v \in V$ is the number of edges incident to v. The **complete graph** is an undirected graph G such that for every $u, v \in V(G)$, the edge $\{u, v\} \in E(G)$.

A **path** in G is a subgraph $P = (\{v_1, \dots, v_{k+1}\}, \{e_1, \dots, e_k\})$ such that $v_i \neq v_j$ for all $i \neq j$, and $e_i = (v_i, v_{i+1})$ for each $i \in \{1, \dots, k\}$. We say that the path starts at v_1 and ends at v_{k+1}, and at times we will simply identify a path by its vertex sequence $v_1, \dots, v_{k+1}$. A **cycle** or **closed path** is a path in which the first and last vertex in the sequence are the same, i.e., $v_1 = v_{k+1}$. A graph G is **acyclic** if it contains no cycles. A **Hamiltonian path** P in a graph G is a path such that $V(P) = V(G)$. Similarly, a **Hamiltonian tour** T in a graph G is a cycle such that $V(T) = V(G)$. A **weighted graph** is a graph G along with a cost function $w : E(G) \to \mathbb{R}$ which assigns a weight to each edge. If a graph is unweighted, then the **length** of a path P is given by $|V(P)|$, and if the graph is weighted, then the length of the path is $\sum_{e \in E(P)} w(e)$.

2.2.2 Traveling Salesperson and Shortest Path Problems

We begin by introducing the Traveling Salesperson (Salesman) Problem. In the terminology of the previous section this problem is also known as the minimum Hamiltonian tour problem.

> **The traveling salesperson problem (TSP):** Given a complete graph G and weights $w : E(G) \to \mathbb{R}_{>0}$, find a Hamiltonian tour T (i.e., a closed path which visits each vertex in $V(G)$ exactly once) whose weight $\sum_{e \in E(T)} w(e)$ is minimum.

In many instances we are interested in the TSP with weights on edges are given by the Euclidean distances between vertices.

> **The Euclidean traveling salesperson problem (ETSP):** Given a finite set $V \subset \mathbb{R}^2$, find a Hamiltonian tour T (i.e., a closed path which visits each point in V exactly once) in the complete graph on V such that the total length $\sum_{\{v,w\} \in E(T)} \|v - w\|_2$ is minimum.

If we are interested in finding shortest paths that visit all vertices in graph rather than shortest tours (that is, if the salesperson is not required to return to their start vertex), then the problems are called the **minimum Hamiltonian path problem (MHP)** and the **Euclidean minimum Hamiltonian path problem (EMHP)**, respectively.

Finally, a variation on the EMHP problem which we will be interested in is stated as follows

> **The Constrained EMHP:** Given a set $V = \{\mathbf{s}, \mathcal{Q}, \mathbf{f}\} \subset \mathbb{R}^2$ consisting of a start point $\mathbf{s}$, a finite set of points $\mathcal{Q}$, and a finish point $\mathbf{f}$, determine the shortest EMHP on V which starts at $\mathbf{s}$, and terminates at $\mathbf{f}$.

Approximate solutions for the TSP and MHP

The well known negative result is that the TSP is *NP*-hard, and for general weights no approximation algorithms exist unless $P = NP$ (Sahni and Gonzalez, 1976). The ETSP is still *NP*-hard, but the structure added by having weights which satisfy the triangle inequality results in the existence of several efficient approximation algorithms. In particular, the *Christofides' algorithm* (Christofides, 1976) is guaranteed to find a tour that is no more than 3/2 times longer than the optimal tour. In addition, the algorithm runs in $O(|V|^3)$ computation time. Another algorithm known as the *double-tree algorithm* produces tours that are no longer than twice the optimal in $O(|V|^2)$ computation time (Korte and Vygen, 2005). There also exist polynomial time approximation schemes. For example, it is shown by Arora (1998) and Mitchell (1999) that a tour no longer than $(1 + \epsilon)$ times the shortest one can be found in $|V|(\log|V|)^{O(1/\epsilon)}$ computation time. Thus, the ETSP can be approximated in polynomial time to any desired degree of accuracy. However, the constants involved in the run time are quite large and thus the approximation schemes appear to have little practical value.

The *Lin-Kernighan algorithm* is a local search heuristic which does not have an approximation guarantee but usually produces solutions within a few percent of the optimal (Lin and Kernighan, 1973). In addition, it is reported to have an empirical running time of about $O(|V|^{2.2})$, and thus is often the algorithm of choice for producing efficient and high quality solutions in many applications. Finally the *Concorde TSP solver*[1], see (Applegate et al., 2006), is a computer code for determining exact solutions of the TSP. It has been used to obtain the optimal solution to problems as large as $15,112$ vertices.

In the subsequent chapters, all numerical experiments involving a solution to the TSP or ETSP will utilize the `linkern` implementation of the Lin-Kernighan algorithm. The `linkern` takes as an input an instance of the TSP, but it can also be used to produce solutions to MHP and EMHP problems. To transform a constrained EMHP problem into a TSP, we can simply replace the distance between the start and end points with a large negative number, ensuring that this edge is included in resulting tour.

[1]The TSP heuristic `linkern` and the solver `Concorde` are freely available for academic research use at `http://www.tsp.gatech.edu/concorde.html`.

The length of Euclidean TSP tour and MHP

An instance of the ETSP (or EMHP) is defined by the set $V \subset \mathcal{E}$, where $\mathcal{E}$ is a compact subset of $\mathbb{R}^2$ with area $|\mathcal{E}|$. Let $N := |V|$, denote the number of points in V. Additionally, let ETSP(V) denote the length of the shortest tour on V and let EMHP(V) denote the length of the shortest path. The following results characterize some aspects of the length of ETSP(V).

Worst-case length: We begin by presenting an upper bound on the length of the EMHP through a set of points in the unit square (Few, 1955). We extend the bound to the case of a rectangular region, as it will prove useful in the subsequent chapters. For completeness, we have included the proof in the Appendix.

Lemma 2.1 (EMHP worst-case length, Few (1955))**.** *Given N points in a $1 \times h$ rectangle in $\mathbb{R}^2$, where $h \in \mathbb{R}_{>0}$, there exists a path that starts from a unit length edge of the rectangle, passes through each of the n points exactly once, and terminates on the opposite unit length edge, having length upper bounded by*

$$\sqrt{2hN} + h + 5/2.$$

The previous lemma has the following consequence.

Theorem 2.2 (ETSP tour worst-case length, Few (1955))**.** *If V is a set of N points in a compact square region $\mathcal{E}$, then* ETSP$(V) \in O(\sqrt{n|\mathcal{E}|})$.

Note that the previous theorem can be equivalently written as follows: There exists $\beta > 0$ such that for every set V of N points in $\mathcal{E}$, ETSP$(\mathcal{Q}) \leq \beta\sqrt{N|\mathcal{E}|}$. In addition, one can show that there exists a set of points (for example, space the points evenly on a grid) such that the length of the ETSP is in $\Omega(\sqrt{N|\mathcal{E}|})$. This implies that the worst-case length of the ETSP is in $\Theta(\sqrt{N|\mathcal{E}|})$. In fact, these results can be extended to higher dimensional spaces. For a compact d-cube $\mathcal{E} \subset \mathbb{R}^d$, $d \geq 2$, with d-volume $|\mathcal{E}|$, the worst-case length of the ETSP is in $\Theta\big((N|\mathcal{E}|)^{1-1/d}\big)$.

Length for randomly distributed points: Assume that the locations of the N points in V are random variables independent and identically distributed (i.i.d.) in a compact set $\mathcal{E}$. The following result characterizes the length of this stochastic ETSP when the points are distributed uniformly.

Theorem 2.3 (Asymptotic ETSP length, Beardwood et al. (1959))**.** *If a set V of N points are distributed independent and uniformly in a compact region $\mathcal{E}$, then there exists a constant β_{TSP} such that, with probability one,*

$$\lim_{N\to+\infty} \frac{\text{ETSP}(V)}{\sqrt{N}} = \beta_{\text{TSP}}\sqrt{|\mathcal{E}|}. \tag{2.1}$$

The constant β_{TSP} has been estimated numerically as $\beta_{\mathrm{TSP}} \approx 0.7120 \pm 0.0002$, (Percus and Martin, 1996). The bound in equation (2.1) holds for all compact sets $\mathcal{E}$, and the shape of $\mathcal{E}$ only affects the convergence rate to the limit. In (Larson and Odoni, 1981), the authors note that if $\mathcal{E}$ is "fairly compact [square] and fairly convex", then equation (2.1) provides an adequate estimate of the optimal TSP tour length for values of N as low as 15.

Theorem 2.3 also characterizes the asymptotic length of the constrained EMHP. More precisely, let EMHP$(\mathbf{s}, \mathcal{Q}, \mathbf{f})$ denote the length of the constrained EMHP on $V := \{\mathbf{s}, \mathcal{Q}, \mathbf{f}\}$. If $\mathbf{s}$, $\mathcal{Q}$ and $\mathbf{f}$ are contained in a compact set, and if the points in $\mathcal{Q}$ are uniformly distributed in the set, then with probability one

$$\lim_{n\to+\infty} \frac{\mathrm{EMHP}(\mathbf{s}, \mathcal{Q}, \mathbf{f})}{\mathrm{ETSP}(\mathcal{Q})} = 1,$$

The result in Theorem 2.3 has been generalized to $\mathbb{R}^d$, $d \geq 2$, and to nonuniform point distributions in (Beardwood et al., 1959) and (Steele, 1990). The result takes the following form: For N i.i.d. random variables with compact support, with probability one

$$\lim_{n\to+\infty} \frac{\mathrm{ETSP}(V)}{n^{1-1/d}} = \beta_{\mathrm{TSP},d} \int_{\mathcal{E}} \bar{f}(\mathbf{q})^{1-1/d}\, d\mathbf{q}, \tag{2.2}$$

where $\bar{f}$ is the density of the absolutely continuous part of the distribution of the points. From equation (2.2), one can readily show that the asymptotic cost of the stochastic ETSP for uniform point distributions is an upper bound on the asymptotic cost for general point distributions.

2.2.3 Translational Minimum Hamiltonian Path Problem

One can think of the length of the shortest ETSP tour (or the shortest EMHP) as the amount of time required for a salesperson moving at unit speed to visit all points in V. An interesting variation on the ETSP and EMHP problems is when the points in V are moving. In particular, consider the case when all points in V translate in the same direction at a fixed speed $v \in]0, 1[$. The goal is determine the shortest path for a unit speed salesperson to visit all moving points. In this case a path consists of an ordering of the points in V and a way of moving between consecutive points.

The problem was proposed and solved by Hammar and Nilsson (2002), and can be stated as follows.

The translational minimum Hamiltonian path (TMHP) problem: Given initial coordinates; $\mathbf{s}$ of a start point, $\mathcal{Q} := \{\mathbf{q}_1, \dots, \mathbf{q}_N\}$

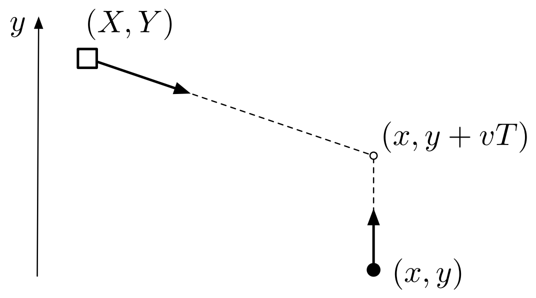


Figure 2.1: Constant bearing control. The salesperson (square) moves towards the point $(x, y + vT)$, where (x, y) is in initial position of the destination point, v is its speed, and T is the travel time in (2.3).

> of a set of points, and $\mathbf{f}$ of a finish point, all moving with speed $v \in]0,1[$ in the positive y-direction, determine a minimum length path for a unit speed salesperson that starts at time zero from point $\mathbf{s}$, visits all points in the set $\mathcal{Q}$ and ends at the finish point $\mathbf{f}$.

The solution is given by the Convert-to-EMHP method:

(i) Define the map $\mathrm{cnvrt}_v : \mathbb{R}^2 \to \mathbb{R}^2$ by

$$\mathrm{cnvrt}_v(x, y) = \left(\frac{x}{\sqrt{1 - v^2}}, \frac{y}{1 - v^2} \right).$$

(ii) Compute the constrained EMHP that starts at $\mathrm{cnvrt}_v(\mathbf{s})$, passes through $\{\mathrm{cnvrt}_v(\mathbf{q}_1), \dots, \mathrm{cnvrt}_v(\mathbf{q}_N)\} =: \mathrm{cnvrt}_v(\mathcal{Q})$ and ends at $\mathrm{cnvrt}_v(\mathbf{f})$.

(iii) To reach a translating point with initial position (x, y) from the initial position (X, Y), move towards the point $(x, y + vT)$, where

$$T = \frac{\sqrt{(1 - v^2)(X - x)^2 + (Y - y)^2}}{1 - v^2} - \frac{v(Y - y)}{1 - v^2}. \tag{2.3}$$

We call the motion described in step (ii) of the Convert-to-EMHP method *constant bearing control*. Isaacs (1965) showed that constant bearing control gives the minimum time motion for moving between successive points. The motion is shown in Figure 2.1.

The length $\mathrm{TMHP}_v(\mathbf{s}, \mathcal{Q}, \mathbf{f})$ of the shortest translational Hamiltonian path is as follows.

Lemma 2.4 (TMHP length, Hammar and Nilsson (2002))**.** *Let the initial coordinates of the start and finish points be* $\mathbf{s} = (x_{\mathbf{s}}, y_{\mathbf{s}})$ *and* $\mathbf{f} = (x_{\mathbf{f}}, y_{\mathbf{f}})$*, and the speed of the points be* $v \in]0,1[$*. Then,*

$$\text{TMHP}_v(\mathbf{s}, \mathcal{Q}, \mathbf{f}) = \text{EMHP}(\text{cnvrt}_v(\mathbf{s}), \text{cnvrt}_v(\mathcal{Q}), \text{cnvrt}_v(\mathbf{f})) + \frac{v(y_{\mathbf{f}} - y_{\mathbf{s}})}{1 - v^2}.$$

This lemma implies the following result: given a start point, a set of points and an end point all of whom translate in the positive vertical direction at speed $v \in]0,1[$, the order of the points followed by the optimal TMHP solution is the same as the order of the points followed by the optimal EMHP solution through a set of static locations equal to the locations of the moving points at initial time converted via the map cnvrt_v.

2.2.4 Longest paths in Directed Acyclic Graphs

We have defined the TSP problem on a complete and undirected graph. In this case there exists a Hamiltonian tour (path). However, when the graph is directed and is not complete, then there may not exist a tour in the graph which visits each vertex exactly once. In this case, an interesting problem is to find the longest path in a graph G.

> **The longest path problem:** Given an unweighted directed graph G, find a path P whose length $|V(P)|$ is maximum.

In general the longest path problem is NP-hard. This can be seen by noting that a solution to the problem would result in a test for determining whether or not a graph contains a Hamiltonian path – a well known NP-complete problem. However, if the graph is a directed acyclic graph (DAG), then the longest path problem has an efficient dynamic programming solution (Christofides, 1975) with complexity $O(|V| + |E|)$, that relies on topologically sorting (Cormen et al., 2001) the vertices.

2.2.5 Maximum and Weighted Matching Problems

To discuss matching problems let us introduce bipartite graphs. In an undirected **bipartite graph** $G = (V(G), E(G))$ we can partition the vertex set $V(G) = A \cup B$, such that $A \cap B = \emptyset$ and for every edge $\{u, v\} \in E(G)$, $u \in A$ and $v \in B$. A **matching** or **assignment** in a bipartite graph is a set of pairwise disjoint edges M (i.e., the endpoints are all different). That is, each vertex can be a member of at most one edge in M. We can think of a matching as a one-to-one assignment of persons in A to jobs in B. The matching M is a **maximum** matching if for every matching $\tilde{M}$, we have $|\tilde{M}| \leq |M|$. If $|A| = |B|$ then we say that a matching is **perfect** when $|M| = |A|$. The matching M is **maximal** if there does not exist a matching $\tilde{M}$, such that $\tilde{M}$ is a strict superset of M. There are several polynomial time algorithms for determining a maximum matching (Hopcroft and Karp, 1973).

In a weighted maximum matching problem, the edges of the bipartite graph G are weighted according to $w : E(G) \to \mathbb{R}$. Two common weighted problems are the following:

> **The sum matching problem:** Given a bipartite graph G with weights $w : E(G) \to \mathbb{R}$, find the matching M whose weight $\sum_{e \in M} w(e)$ is minimum.

> **The bottleneck matching problem:** Given a bipartite graph G with weights $w : E(G) \to \mathbb{R}$, find the matching M whose weight $\max_{e \in M} w(e)$ is minimum.

Assuming that the bipartite graph G has partition $V(G) = A \cup B$ where $|A| = |B| = N$, then sum matching problem (or sum assignment problem) can be solved in $O(N^3)$ computation time using methods such as the *Hungarian method* (Kuhn, 1955) or the *auction algorithm* (Bertsekas and Tsitsiklis, 1997). The sum matching (assignment) problem can also be written as a integer linear program with a totally unimodular constraint matrix (Korte and Vygen, 2005). This implies that one can solve the sum matching problem by solving a linear program obtained by relaxing the integer constraints.

In the following chapters we will require a standard algorithm, called Maximal Match, for computing a maximal matching. The algorithm chooses the edge with lowest cost, adds it to the matching, removes the edge from the problem, and repeats. In the case in which the bipartite graph is complete (i.e., for every $u \in A$ and $v \in B$, $\{u, v\} \in E$), this algorithm determines a perfect, and thus maximum, matching.

Algorithm 2.1: Maximal Match

```
  Input: A bipartite graph $G = (A(G) \cup B(G), E(G))$
  Output: A maximal matching $M$
1 Initialize $M := \emptyset$, and $E := E(G)$.
2 while $E$ is nonempty do
3     Find the lowest cost edge $\{u, v\} \in E$ and add it to $M$.
4     Remove every edge from $E$ that has an endpoint $u$ or endpoint $v$.
```

2.2.6 Vertex Coloring

Here we review the concept of a vertex coloring on a graph, which will be useful in producing non-conflicting schedules. A *vertex-coloring* of an undirected graph G is a mapping $f : V(G) \to \mathbb{N}$ with $f(v) \neq f(w)$ for all $\{v, w\} \in E(G)$. The number $f(v)$ is the *color* of v. Finding the minimum vertex coloring (i.e., a vertex coloring for which the number of colors is minimum) is NP-hard, and no approximation algorithms exist. However, the following theorem gives an upper bound on the number of colors required.

Theorem 2.5 (Vertex coloring, Korte and Vygen (2005))**.** *Let G be an undirected graph with N vertices and with maximum degree α. Then G has a vertex coloring with at most $\alpha + 1$ colors, and such a coloring can be found in $O(N)$ computation time using the* Greedy Color *algorithm.*

Algorithm 2.2: Greedy Color

```
  Input: An undirected graph $G = (\{v_1, \dots, v_N\}, E)$
  Output: A color $f(v_i)$ for each $i \in \{1, \dots, N\}$.
1 for $i$ from 1 to $N$ do
2     Set $f(v_i)$ to the minimum color $k \in \mathbb{N}$ such that $k \neq f(v_j)$ for all
      neighboring vertices $v_j$, $j < i$.
```

2.3 Queueing Theory

In this section we give a very brief review of queueing models and the Poisson process. For a more detailed review we refer the reader to (Kleinrock, 1975, 1976).

2.3.1 Queueing system models

A *queueing system* can be described as customers (or demands) arriving for service, waiting for service, and leaving the system after being serviced. While waiting for service the customers enter a queue, which can be described as a set $Q(t)$ with cardinality $N(t)$. When a customer reaches the head of line, it is removed from $Q(t)$ and served period of time, after which it leaves the system. That is, $N(t)$ represents the number of customers waiting in the queue at time t. Generally, the arrival of new customers is modeled as a stochastic process with independent and identically distributed (i.i.d.) inter-arrival times, and average arrival rate λ. The service time required for each customer is an i.i.d. random variable with mean $\bar{s}$. If there are n servers, then the *traffic intensity* or **load factor** is defined as

$$\varrho := \frac{\lambda \bar{s}}{n}.$$

The load factor captures the fraction of time the n servers are busy providing service.

We are often interested in determining the steady-state length of the queue, defined as $\bar{N} := \lim_{t\to+\infty} \mathbb{E}[N(t)]$. It turns out that when $\varrho > 1$, the queue never settles down to a steady-state and the $\bar{N}$ is not defined (i.e., it is $+\infty$). Thus, a necessary condition for the existence of a finite steady-state queue length is that $\varrho \leq 1$. If a queue reaches a steady-state then we say that it is **stable**. In many cases, it is difficult to study the properties of a queueing system for all values of $\varrho \in [0, 1)$, and a common technique is to focus on the limiting regimes of $\varrho \to 1^-$, referred to as the **heavy load** regime, and $\varrho \to 0^+$, referred to as the **light load** regime.

Another quantity of interest is the amount of time each customer spends waiting in the queue, which is called the **wait-time**, and for the ith customer it is denoted $\mathcal{W}_i$. If the queue reaches a steady-state, then we can define the steady-state wait-time to be

$$\mathcal{W} := \lim_{i\to+\infty} \mathbb{E}[\mathcal{W}].$$

Similarly, we can define the **service delay** of the ith customer to be $D_i = \mathcal{W}_i + s_i$, where s_i is the service time of customer i. The steady-state service delay is then simply $D := \mathcal{W} + \bar{s}$.

A useful result in queueing theory is called **Little's law**, which relates the steady-state queue length to the steady-state wait-time:

$$\bar{N} = \lambda \mathcal{W}.$$

Thus, in a stable system, if one determines the steady-state wait-time, this yields the steady-state queue length.

2.3.2 The G/G/1 queueing system

A G/G/1 system is a queueing system with a general arrival process, generally distributed service times, and a single server. More specifically, we consider i.i.d. inter-arrival times with expected value $1/\lambda$ and variance σ_a^2; and i.i.d. service times with expected value $\bar{s}$ and variance σ_s^2. For such a system the wait-time satisfies (Kleinrock, 1976)

$$\mathcal{W} \leq \frac{\lambda(\sigma_a^2 + \sigma_s^2)}{2(1-\varrho)}. \tag{2.4}$$

In addition, in heavy load as $\varrho \to 1^-$ the upper bound is exact.

2.3.3 The Poisson process

A common model for the arrival of customers is the Poisson process. Its defining feature is that the inter-arrival times are exponentially distributed. More formally, we define the process as follows. A *Poisson process* with intensity λ is an arrival counting process $\{N(t) : t \geq 0\}$ where $N(t)$ denotes the total number of arrivals up to time t, with $N(0) = 0$, and which satisfies the following three assumptions:

(i) If $t_1 < t_2$ then $N(t_1) \leq N(t_1)$;

(ii) In a small time interval $h > 0$ the number of arrivals satisfies[2]:

$$\mathbb{P}\left[N(t+h) = j_1 + j_2 | N(t) = j_1\right] = \begin{cases} \lambda h + o(h) & \text{if } j_2 = 1, \\ o(h) & \text{if } j_1 > 1, \\ 1 - \lambda h + o(h) & \text{if } j_2 = 0, \end{cases}$$

(iii) If $t_1 < t_2$, then the number $N(t_2) - N(t_1)$ of arrivals in the interval $]t_1, t_2]$ is independent of the times of arrivals in $[0, t_1]$;

It can be shown that the random variable $N(t)$ has the Poisson distribution with parameter λt, which implies that

$$\mathbb{P}\left[N(t) = j\right] = \frac{(\lambda t)^j}{j!} \mathrm{e}^{-\lambda t}, \quad j = 0, 1, 2, \ldots. \tag{2.5}$$

A consequence of equation (2.5) is that in a time interval $[t_1, t_2]$ where $t_2 > t_1$, we have

$$\mathbb{E}\left[N(t_2) - N(t_1)\right] = \lambda(t_2 - t_1).$$

2.3.4 Spatial queueing systems

In a spatial queueing system, customers arrive at distinct locations in space, and the server must travel to a customers' location in order to provide service to it. In such systems the total service requirement of a customer, say customer i, consists not only of the on-site service time s_i, but also the travel time from the customer

[2]We say that $f(h) \in o(g(h))$ if $f(h)/g(h) \to 0^+$ as $h \to 0^+$.

serviced prior to i, to customer i's location. This difference is very significant as it implies that the total service requirements of the customers are not i.i.d. random variables (Bertsimas and van Ryzin, 1991). Because of this, the results in standard queueing theory such as equation (2.4), do not apply. In addition, in much of queueing theory, the policy by which the server attends to customers is simply first-come-first-served. However, in spatial queueing the server may wish to attend to customers based on their proximity rather than on their order of arrival. Thus, for spatial queueing the problem arises of determining an efficient order in which to attending to customers.

To highlight the differences between classic queueing and spatial queueing, consider customers arriving according to a Poisson process with rate λ. Upon arrival each customer assumes a i.i.d. position uniformly in a compact environment $\mathcal{E}$. The on-site service requirement of each customer is i.i.d. with expected value $\bar{s}$. Suppose there is a single server that can move in $\mathcal{E}$ with speed $v_{\max}$. Then, for any policy by which the vehicle services the customers, the steady-state wait time satisfies

$$\mathcal{W} \geq \gamma^2 \frac{\lambda|\mathcal{E}|}{v_{\max}^2(1-\varrho)^2} - \frac{\bar{s}}{2\varrho},$$

where $\gamma = 2/(3\sqrt{2\pi}) \approx 0.266$. What is striking about the above bound is that it shows that in heavy load conditions as $\varrho \to 1^-$, the wait-time in spatially queueing must increase with $(1-\varrho)^{-2}$. In contrast, for a classic queueing system, equation (2.4) tells us that the wait-time scales as $(1-\varrho)^{-1}$.

The above problem is known as the dynamic traveling repairperson problem (DTRP). In Chapter 4 we will review the DTRP more extensively.

Chapter Three

Monotonic Target Assignment

In this chapter we consider a task allocation problem that consists of a group of n mobile robotic agents (henceforth called *vehicles*), equipped with wireless transceivers for limited range communication, dispersed in an environment $\mathcal{E} \subset \mathbb{R}^2$ which contains n target (or task) locations. In addition, we consider two scenarios: (1) each vehicle is given a list containing all target positions (the positions may be given as GPS coordinates); or (2) each vehicle has no initial target information, but has a finite-range target sensor to acquire target positions. The task is for the vehicles to divide the targets among themselves so that in minimum time, each target location is occupied by an vehicle. Since no *a priori* assignment of target-vehicle pairs has been given, the vehicles must solve the problem through communication and motion. We call this the *target assignment problem*. This problem has many applications in UAV surveillance and exploration, or mobile sensor networks. The first scenario could arise when a high-altitude, sensory-rich aircraft communicates a large number of target positions to a correspondingly large group of smaller, slower, autonomous aircraft at lower altitudes. The second (local sensing) scenario could arise in exploration tasks where a group of UAVs are sent into a region to find, and provide service to, spatially distributed tasks.

Unlike the prior work in the area of task allocation (see Chapter 1 for a review), in this chapter we study the scalability properties of the minimum-time target assignment problem. We assume that each vehicle has limited communication capabilities and either (1) *full target knowledge* (i.e., each vehicle knows the position of every target), or (2) *local target sensing* (i.e., each vehicle has a finite-range target sensor to acquire target positions). We focus on characterizing the completion time as the number of vehicles n grows, and the square environment $\mathcal{E}(n)$ grows to accommodate them.[1]

The contributions of this chapter are: a novel and concise statement of the minimum-time target assignment problem for vehicle networks; a broad

[1]The size of the environment $\mathcal{E}$ is a function of n, and thus we write $\mathcal{E}(n)$. If the environment size were independent of n, then the density of robots would become arbitrarily large as the task size n became large, which is not realistic. Thus, either the environment should grow with n (as is assumed here), or the robot's attributes should shrink with n (as discussed in Section 3.7.3).

class of distributed algorithms for solving this problem; lower bounds on the worst-case performance achievable by any algorithm in this class; and algorithms which perform within a constant factor of the optimal monotonic algorithm for worst-case initial conditions. In Section 3.3.1 we introduce the class of *monotonic algorithms*, which provides an intuitive approach for solving the target assignment problem. We show that in "sparse environments," that is when $|\mathcal{E}(n)|/n \to +\infty$, for every monotonic algorithm there exists a (worst-case) set of initial target and vehicle positions such that the completion time is in $\Omega(\sqrt{|\mathcal{E}(n)|n})$.[2] In "dense environments," that is when $|\mathcal{E}(n)|/n \to 0^+$, every algorithm in the class has worst-case completion time in $\Omega(|\mathcal{E}(n)|)$. In Section 3.4, we assume full target knowledge and present a monotonic algorithm, called the ETSP Assgmt algorithm, with worst-case completion time in $O(\sqrt{|\mathcal{E}(n)|n})$. In this algorithm, each vehicle computes an ETSP tour through the n targets, turning the cloud of target points into an ordered ring. Vehicles then move along the ring, looking for the next available target. When vehicles communicate, they exchange information on the location of the next available target along the ring. Then, in Section 3.5 we present a monotonic algorithm, called the Grid Assgmt algorithm, which operates under either the full target knowledge assumption, or the local target sensing assumption as long as the sensing range is at least $\sqrt{2/5}$ times the communication range. Under either assumption, the Grid Assgmt algorithm has worst-case completion time in $O(|\mathcal{E}(n)|)$. In this algorithm, the vehicles partition the environment into cells, and determine local maximum assignments in the cell which they occupy. A leader is elected in each cell, and through communication between leaders of adjacent cells, local assignments are merged into a global and complete assignment. These two algorithms are complementary in terms of worst-case performance: in "sparse environments," the ETSP Assgmt algorithm is within a constant factor of the optimal monotonic algorithm, and is "dense environments," the Grid Assgmt algorithm is within a constant factor of the optimal monotonic algorithm.

We also characterize the stochastic properties of the Grid Assgmt algorithm in "dense environments." If the vehicles and targets are uniformly distributed, then the completion time belongs to $O(\sqrt{|\mathcal{E}(n)|})$ with high probability. Additionally, if there are n vehicles and only $n/\log n$ targets, then the completion time belongs to $O(1)$ with high probability. In Section 3.7 we discuss extensions of the ETSP Assgmt and Grid Assgmt algorithms to higher dimensional spaces and to the case of n vehicles and m targets, $n \neq m$.

[2] $|\mathcal{E}(n)|$ denotes the area of $\mathcal{E}(n)$, and $\Omega(\cdot)$ is the asymptotic notation for lower bounds as reviewed in Section 3.1.

3.1 Preliminary Material

In this section we review a few useful results on occupancy problems and random geometric graphs.

3.1.1 Occupancy Problems

Occupancy problems, or "bins and balls" problems, are concerned with randomly distributing m balls into n equally sized bins. The following results will be useful in our analysis. We say that event $A(n)$ occurs *with high probability* (w.h.p.) if $\mathbb{P}[A(n)] \to 1^-$ as $n \to +\infty$ (i.e., the probability of event $A(n)$ occurring tends to one as n tends to infinity).

Theorem 3.1 (Bins and balls properties, (Motwani and Raghavan, 1995; Xue and Kumar, 2004))**.** *Consider uniformly randomly distributing m balls into n bins and let γ be any function such that $\gamma(n) \to +\infty$ as $n \to +\infty$. The following statements hold:*

(i) if $m = n$, then w.h.p. each bin contains at most $O\left(\frac{\log n}{\log\log n}\right)$ balls;

(ii) if $m = n\log n + \gamma(n)n$, then w.h.p. there exist no empty bins;

(iii) if $m = n\log n - \gamma(n)n$, then w.h.p. there exists an empty bin;

(iv) if $m = Kn\log n$, where $K > 1/\log(4/\mathrm{e})$, then w.h.p. every bin contains $\Theta(\log n)$ balls.

We will be interested in partitioning a square environment into equally sized and openly disjoint square bins such that the area of each bin is "small." To do this, we require the following simple fact.

Lemma 3.2 (Dividing the environment)**.** *Given $n \in \mathbb{N}$ and $r_{\mathrm{comm}} > 0$, consider a square environment $\mathcal{E}(n)$. If $\mathcal{E}(n)$ is partitioned into b^2 equally sized and openly disjoint square bins, where*

$$b := \left\lceil \frac{\sqrt{5|\mathcal{E}(n)|}}{r_{\mathrm{comm}}} \right\rceil, \tag{3.1}$$

then the area of each bin is no more than $r_{\mathrm{comm}}^2/5$. Moreover, if $x, y \in \mathcal{E}(n)$ are in the same bin or in adjacent bins, then $\|x - y\| \leq r_{\mathrm{comm}}$.

3.1.2 Random Geometric Graphs

In Chapter 2 we introduced directed and undirected graphs. Here we introduce an undirected graph whose vertices are embedded in the plane, and whose edges represent geometric relations between edges. For $n \in \mathbb{N}$ and $r_{\text{comm}} \in \mathbb{R}_{>0}$, a planar *geometric graph* $G(n, r_{\text{comm}})$ consists of n vertices in $\mathbb{R}^2$, and undirected edges connecting all vertex pairs $\{x, y\}$ with $\|x - y\| \leq r_{\text{comm}}$. We also refer to this as the r_{comm}-geometric graph. If the vertices are randomly distributed in some subset of $\mathbb{R}^2$, then we call the graph a *random geometric graph*.

Theorem 3.3 (Connectivity of geometric graphs, (Penrose, 2003))**.** *Consider the random geometric graph $G(n, r_{\text{comm}})$ obtained by uniformly randomly distributing n points in the square environment $\mathcal{E}(n)$ with*

$$\frac{\pi r_{\text{comm}}^2}{|\mathcal{E}(n)|} = \frac{\log n + \gamma(n)}{n}.$$

Then $G(n, r_{\text{comm}})$ is connected w.h.p. if and only if $\gamma(n) \to +\infty$ as $n \to +\infty$.

This theorem will be important for understanding some of our results, as it provides a bound on the environment size necessary for the communication graph of n randomly deployed vehicles to be asymptotically connected.

3.2 Network Model and Problem Statement

In this section we formalize our vehicle and target models and define the sparse and dense environments.

3.2.1 Vehicle Network Model

Consider n vehicles in an environment $\mathcal{E}(n) := [0, \ell(n)]^2 \subset \mathbb{R}^2$, where $\ell(n) > 0$ (that is, $\mathcal{E}(n)$ is a square with side length $\ell(n)$). The environment $\mathcal{E}(n)$ is compact for each n but its size depends on n. A robotic vehicle, $\mathcal{V}^{[i]}$, $i \in \mathcal{I} := \{1, \ldots, n\}$, is described by the tuple

$$\mathcal{V}^{[i]} := \{\text{UID}^{[i]}, \mathbf{p}^{[i]}, r_{\text{comm}}, r_{\text{sense}}, \mathbf{u}^{[i]}, \mathcal{M}^{[i]}\},$$

where the quantities are as follows: Its unique identifier (UID) is $\textsc{uid}^{[i]}$, taken from the set $I_{\textsc{uid}} \subset \mathbb{N}$. Note that each vehicle does not know the set of UIDs being used and thus does not initially know the magnitude of its UID relative to those of other vehicles. Its position is $\mathbf{p}^{[i]} \in \mathcal{E}(n)$. Its communication range is $r_{\text{comm}} > 0$, i.e., two vehicles, $\mathcal{V}^{[i]}$ and $\mathcal{V}^{[k]}$, $i, k \in \mathcal{I}$, can communicate if and only if $\|\mathbf{p}^{[i]} - \mathbf{p}^{[k]}\| \leq r_{\text{comm}}$. Its target sensing range is r_{sense}. With this sensor vehicle i can determine the relative position of targets within distance r_{sense} of $\mathbf{p}^{[i]}$. Its continuous time velocity input is $\mathbf{u}^{[i]}$, corresponding to the kinematic model $\dot{\mathbf{p}}^{[i]} = \mathbf{u}^{[i]}$, where $\|\mathbf{u}^{[i]}\| \leq v_{\max}$ for some $v_{\max} > 0$. Finally, its memory is $\mathcal{M}^{[i]}$ and is of cardinality (size) $|\mathcal{M}^{[i]}|$. From now on, we simply refer to vehicle $\mathcal{V}^{[i]}$ as vehicle i.

The vehicles move in continuous time and communicate according to a synchronous discrete time schedule consisting of an increasing sequence $\{t_k\}_{k\in\mathbb{N}}$ of time instants with no accumulation points. We assume $|t_{k+1} - t_k| \leq t_{\max}$, for all $k \in \mathbb{N}$, where $t_{\max} \in \mathbb{R}_{>0}$. We also assume that the time between communication rounds $t_{\max}$ is much smaller than $r_{\text{comm}}/v_{\max}$, the amount of time taken to travel the distance r_{comm}. At each communication round, vehicles can exchange messages of length $O(\log n)$.[3] Communication round k occurs at time t_k, and all messages are sent and received instantaneously at t_k. Motion then occurs from t_k until t_{k+1}. It should be noted that in this setup we are emphasizing the time complexity due to the motion of the vehicles.

3.2.2 The Target Assignment Problem

Let $\mathcal{Q} := \{\mathbf{q}_1, \ldots, \mathbf{q}_n\} \subset \mathcal{E}(n)$ be a set of distinct target locations. In this chapter we make one of two assumptions:

Full target knowledge: Each vehicle knows the position of every target. Thus, vehicle i's memory, $\mathcal{M}^{[i]}$, contains a copy of $\mathcal{Q}$, which we denote $\mathcal{Q}^{[i]}$. To store $\mathcal{Q}^{[i]}$ the size of each vehicles' memory, $|\mathcal{M}^{[i]}|$, must be in $\Omega(n)$.

Local target sensing: Each vehicle has no initial target information (i.e., $\mathcal{Q}^{[i]} = \emptyset$), but can acquire target positions through its target sensor of range r_{sense}.

Our goal is to solve the *target assignment problem*:

> Determine an algorithm for $n \in \mathbb{N}$ vehicles, with attributes as described above, satisfying the following requirement; there exists a time $T \geq 0$ such that for each target $\mathbf{q}_j \in \mathcal{Q}$, there is a unique vehicle $i \in \mathcal{I}$, with $\mathbf{p}^{[i]}(t) = \mathbf{q}_j$ for all $t \geq T$.

If the task begins at time $t = 0$, then the *completion time* of the target assignment task is the minimum $T \geq 0$, such that for each $\mathbf{q}_j \in \mathcal{Q}$, there is a unique $i \in \mathcal{I}$, with $\mathbf{p}^{[i]}(t) = \mathbf{q}_j$ for all $t \geq T$. In this chapter we seek algorithms that minimize this completion time. Note that in the local target sensing assumption the vehicles have less target information than in the full target knowledge assumption. Because

[3]The number of bits required to represent an ID, unique among n vehicles, is directly proportional to the logarithm of n.

of this, an algorithm's performance under the local target sensing assumption can be no better than its performance under the full target knowledge assumption.

Remark 3.4 (Consistent target knowledge)**.** *Another possible assumption on the target sets, $\mathcal{Q}^{[i]}$, which still ensures the existence of a complete matching, is the* consistent target knowledge *assumption: For each $\mathcal{K} \subseteq \mathcal{I}$, $\left| \cup_{k\in\mathcal{K}} \mathcal{Q}^{[k]} \right| \geq |\mathcal{K}|$. In fact, it was proved by Frobenius in 1917 and by Hall in 1935 that this is the necessary and sufficient condition for the existence of a complete matching (Korte and Vygen, 2005).* •

3.2.3 Sparse, Dense, and Critical Environments

We wish to study the scalability of a particular approach to the target assignment problem; that is, how the completion time increases as we increase the number of vehicles, n. The velocity $v_{\max}$ and communication range r_{comm} of each vehicle are independent of n. However, we assume that the size of the environment increases with n in order to accommodate an increase in vehicles. Borrowing terms from the random geometric graph literature (Penrose, 2003), we say that the environment is *sparse* if, as we increase the number of vehicles, the environment grows quickly enough that the density of vehicles (as measured by the sum of their communication footprints) decreases; we say the environment is *critical*, if the density is constant, and we say the environment is *dense* if the density increases. Formally, we have the following definition.

Definition 3.5 (Dense, critical and sparse environments)**.** *The environment $\mathcal{E}(n)$ is*

(i) sparse *if $|\mathcal{E}(n)|/n \to +\infty$ as $n \to +\infty$;*

(ii) critical *if $|\mathcal{E}(n)|/n \to \text{const} \in \mathbb{R}_{>0}$ as $n \to +\infty$;*

(iii) dense *if $|\mathcal{E}(n)|/n \to 0^+$, as $n \to +\infty$.*

It should be emphasized that a dense environment does not imply that the communication graph between vehicles is dense. On the contrary, from Theorem 3.3 we see that the communication graph at random vehicle positions in a dense environment may not even be connected.

3.3 Classes of Algorithms

In this section we introduce a class of algorithms for the target assignment problem that provides the structure for algorithms developed in this chapter.

We will provide a lower bound on the classes performance using the full target knowledge assumption. Necessarily this also provides a lower bound for the problem using the local target sensing assumption.

3.3.1 Monotonic Algorithms

We introduce a class of algorithms which provides an intuitive approach to target assignment.

Definition 3.6 (Monotonic algorithms)**.** *A target assignment algorithm is* monotonic *if it is deterministic and has the following property: If a subset of vehicles $\mathcal{J} \subset \mathcal{I}$ are all located at target $\mathbf{q}_j$ at time t_1 (i.e., $\mathbf{p}^{[i]}(t_1) = \mathbf{q}_j$, $\forall\, i \in \mathcal{J}$), then at least one vehicle in $\mathcal{J}$ remains located at $\mathbf{q}_j$ for all $t > t_1$ (i.e., $\exists\, i \in \mathcal{J}$ such that $\mathbf{p}^{[i]}(t) = \mathbf{q}_j$, $\forall\, t > t_1$).*

We call these algorithms "monotonic" since occupied targets remain occupied for all time, and thus the number of occupied targets monotonically increases throughout the execution. We focus on monotonic algorithms for two reasons: First, monotonicity is a natural constraint for target assignment problems since in many scenarios the vehicles will begin servicing a target immediately upon arriving at its location—in non-monotonic algorithms, service will be halted as vehicles leave their targets. Second, monotonic algorithms provide a broad class of algorithms for which rigorous analysis remains tractable.

We are now ready to lower bound the worst-case asymptotic completion time of the target assignment problem for any monotonic algorithm. This bound holds under both the full target knowledge and local target sensing assumptions.

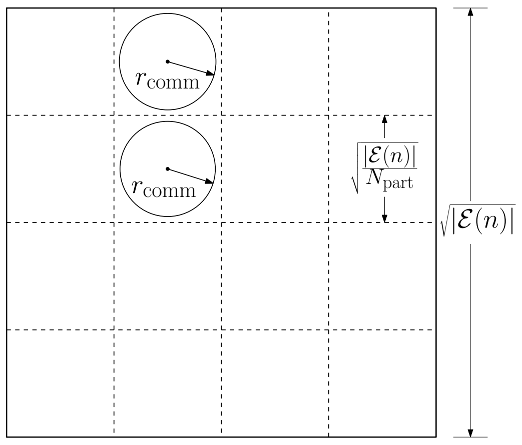
Theorem 3.7 (Time complexity of target assignment)**.** *Consider n vehicles, with communication range $r_{\text{comm}} > 0$, and n targets in $\mathcal{E}(n)$. For all monotonic algorithms the worst-case completion time of the target assignment problem is lower bounded as follows:*

(i) if $\mathcal{E}(n)$ is sparse, then the completion time is in $\Omega(\sqrt{n|\mathcal{E}(n)|})$;

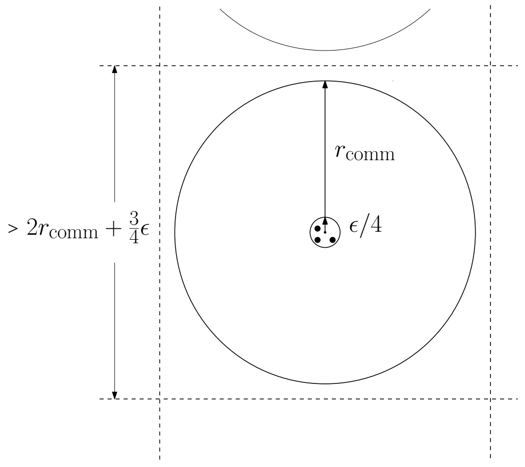
(ii) if $\mathcal{E}(n)$ is critical, then the completion time is in $\Omega(n)$;

(iii) if $\mathcal{E}(n)$ is dense, then the completion time is in $\Omega(|\mathcal{E}(n)|)$.

Proof. The proof proceeds by constructing a set of vehicle and target positions such that the lower bound is achieved. To do this, we place the targets in $\mathcal{E}(n)$ such that the r_{comm}-geometric graph, generated by the target positions, has a maximum number of disconnected components. Next we place vehicles $2, \dots, n$ so that they occupy targets $\mathbf{q}_2, \dots, \mathbf{q}_n$. We then place vehicle 1 in $\mathcal{E}(n) \setminus \mathcal{Q}$. If the


r_{comm}
r_{comm}
$\sqrt{\frac{|\mathcal{E}(n)|}{N_{\text{part}}}}$
$\sqrt{|\mathcal{E}(n)|}$

(a) Partitioning $\mathcal{E}(n)$ to construct target positions that generate an r_{comm}-geometric graph with a maximum number of disconnected components.



(b) The $\epsilon/4$-disk located at the center of one of the squares in the partition. Targets are shown in the disk, along with a lower bound on side-length.

Figure 3.1: Partitioning the environment $\mathcal{E}(n)$ for the proof of Theorem 3.7.

vehicles run a monotonic algorithm to solve the target assignment problem, then vehicles $2, \ldots, n$ will not move, and thus the assignment will not be complete until vehicle 1 reaches target $\mathbf{q}_1$. In the best case, when vehicle 1 comes within distance r_{comm} of a connected component, it immediately determines whether or not there is a free target in that component (i.e., whether or not $\mathbf{q}_1$ is in that component). However, vehicle 1 will not receive information about the availability of any targets outside of that component. So, vehicle 1 must come within distance r_{comm} of the connected component containing $\mathbf{q}_1$, before the assignment can be completed. Since the algorithm is deterministic, we can place the targets, and vehicles such that the connected component containing $\mathbf{q}_1$ is the last connected component that vehicle 1 will visit.

To create the maximum number of disconnected components, we partition the environment, $\mathcal{E}(n)$ into N_{part} equally sized, and openly disjoint squares, as shown in Fig. 3.1(a). We consider two cases, based on whether or not there exists an $\epsilon > 0$ such that $|\mathcal{E}(n)| \geq (2r_{\text{comm}} + \epsilon)^2 n$.

Case 1: [there exists $\epsilon > 0$ such that $|\mathcal{E}(n)| \geq (2r_{\text{comm}} + \epsilon)^2 n$] In this case we set $N_{\text{part}} := \lceil \sqrt{n} \rceil^2$ and place a target at the center of each square until there are no targets remaining. The area of each square is given by $|\mathcal{E}(n)|/N_{\text{part}}$, and thus the distance between any two targets is lower bounded by $\sqrt{|\mathcal{E}(n)|/N_{\text{part}}} \geq \sqrt{(2r_{\text{comm}} + \epsilon)^2 n / \lceil \sqrt{n} \rceil^2}$, which for sufficiently large n, is greater than $2r_{\text{comm}}$. Thus, we have created n disconnected components, as depicted in Fig. 3.1(a). The distance between r_{comm}-disks centered at any two targets is lower bounded by $\sqrt{|\mathcal{E}(n)|/\lceil\sqrt{n}\rceil^2} - 2r_{\mathrm{comm}}$, and we can place the vehicles and targets such that one vehicle must travel this distance $n-1$ times. Thus, the worst-case travel distance is at least

$$(n-1)\left(\sqrt{\frac{|\mathcal{E}(n)|}{\lceil\sqrt{n}\rceil^2}} - 2r_{\mathrm{comm}}\right) \in \Omega(\sqrt{|\mathcal{E}(n)|n}).$$

Since the vehicle travel at constant speed, the completion time is also in $\Omega(\sqrt{|\mathcal{E}(n)|n})$.

Case 2: [for every $\epsilon > 0$, $|\mathcal{E}(n)| < (2r_{\mathrm{comm}} + \epsilon)^2 n$] In this case we fix any $\epsilon > 0$ and set

$$N_{\mathrm{part}} := \left\lceil \sqrt{\frac{|\mathcal{E}(n)|}{(2r_{\mathrm{comm}} + \epsilon)^2}} \right\rceil^2.$$

We define a disk of radius $\epsilon/4$ at the center of each of the N_{part} squares. We then place $\lceil n/N_{\mathrm{part}}\rceil$ targets in each $\epsilon/4$-disk, until there are no targets remaining, as shown in Fig. 3.1(b). Note that for any $\alpha \in]0,1[$, we can find $N \in \mathbb{N}$ such that

$$N_{\mathrm{part}} < \frac{|\mathcal{E}(n)|}{(2r_{\mathrm{comm}} + \alpha\epsilon)^2}, \quad \text{for all } n \geq N.$$

Letting $\alpha = 3/4$, we find that for large n, the distance between the centers of any two squares is lower bounded by $\sqrt{\frac{|\mathcal{E}(n)|}{N_{\mathrm{part}}}} \geq 2r_{\mathrm{comm}} + 3\epsilon/4$, as shown in Fig. 3.1(b). So, the distance between any two $\epsilon/4$-disks is lower bounded by $2r_{\mathrm{comm}} + \epsilon/4$. Thus, we have created $\Omega(N_{\mathrm{part}})$ disconnected components. The distance between r_{comm}-disks centered at any two targets in different squares is lower bounded by $\epsilon/4$. Again, we can place the vehicles and targets such that one vehicle will have to travel this distance $\Omega(N_{\mathrm{part}})$ times. Thus, the worst-case distance is lower bounded by

$$\frac{\epsilon}{4}\Omega(N_{\mathrm{part}}) \in \Omega(|\mathcal{E}(n)|).$$

Since the vehicles travel at constant speed, the completion time is also in $\Omega(|\mathcal{E}(n)|)$.

Thus, if $|\mathcal{E}(n)|/n \to +\infty$ as $n \to +\infty$, then we are in Case 1 and the completion time is in $\Omega(\sqrt{|\mathcal{E}(n)|n})$. If $|\mathcal{E}(n)|/n \to \mathrm{const} \in \mathbb{R}_{>0}$ as $n \to +\infty$, then we may be in either Case 1 or Case 2, depending on the value of const, but in either case the completion time is in $\Omega(n)$. Finally, if $|\mathcal{E}(n)|/n \to 0^+$ as $n \to +\infty$, then we are in Case 2 and the completion time is in $\Omega(|\mathcal{E}(n)|)$. □

Remark 3.8 (Interpretation of lower bound). *In Theorem 3.7 we provided a worst-case lower bound. This should be interpreted as follows. For every* *monotonic algorithm there exists a set of initial target and vehicle position for which the completion time is no smaller than the lower bound. It should be noted that there are many initial positions for which the completion time is less than this worst-case lower bound (indeed, there are initial positions for which the completion time is zero).*

Also note that for a critical environment, the vehicle and target positions used in the proof of Theorem 3.7 give a completion time of $\Omega(n)$ for every monotonic algorithm. However, if a centralized solver were used to assign vehicles to targets from the same initial positions, then the motion time would be $O(1)$. Hence the distributed solutions given by a monotonic algorithm may severely under-perform when compared to solutions given by the optimal centralized solver. •

3.3.2 The Rendezvous Strategy and its Drawbacks

In this section we discuss another approach to solving the target assignment problem that we call the Rendezvous Strategy. The strategy, which works only under the full target knowledge assumption, can be described as follows.

```
Rendezvous Strategy (for vehicle $i$)
1 Compute a common meeting point, such as the centroid of the target
  positions.
2 Move to the meeting point and wait for all other vehicles to arrive.
3 Once all vehicles have arrived, broadcast UID$^{[i]}$ and $\mathbf{p}^{[i]}$, and receive UID$^{[k]}$
  and $\mathbf{p}^{[k]}$ from all other vehicles.
4 Compute a complete assignment of target-vehicle pairs using the Maximal
  Match algorithm and move to your assigned target.
```

Since every vehicle knows the position of all targets, the vehicles can compute a common meeting point. The time for an vehicle to reach any meeting point is bounded by $\sqrt{2}|\mathcal{E}(n)|/v_{\max}$, and thus each vehicle can determine when all other vehicles have arrived at the meeting point. Once all vehicles reach the meeting point the communication graph is complete and each vehicle can broadcast its UID and position to all other vehicles in one communication round. Then, each vehicle can use Maximal Match to solve a centralized assignment, and all vehicles end up with the same complete assignment. In addition, since the vehicles are co-located, this assignment is optimal. Each vehicle then moves to the target to which it has been assigned. Essentially, this approach turns the distributed problem into a centralized one.

Theorem 3.9 (Time bound for Rendezvous Strategy)**.** *Consider n vehicles and n targets in the environment $\mathcal{E}(n)$. In the worst-case, the* Rendezvous

Strategy *solves the target assignment problem in* $\Theta(\sqrt{|\mathcal{E}(n)|})$ *time. Moreover, if the targets and vehicles are uniformly randomly distributed in* $\mathcal{E}(n)$, *the completion time is in* $\Theta(\sqrt{|\mathcal{E}(n)|})$.

Proof. Since all information can be exchanged in one round, and we are not considering computation time, the completion time is given by the time to reach the meeting point plus the time to go from the meeting point to the assigned target. To see the worst-case, place all targets at one side of the environment, and all vehicles at the other side. Then each vehicle must travel a distance $\Theta(\sqrt{|\mathcal{E}(n)|})$. The distance from the meeting point back to any assigned target is also bounded by $O(\sqrt{|\mathcal{E}(n)|})$. Thus, the worst-case completion time is $\Theta(\sqrt{|\mathcal{E}(n)|})$.

If we uniformly randomly distribute n vehicles in $\mathcal{E}(n)$, then it is a well known fact (see, for example (Sharma et al., 2007)) that w.h.p., the maximum distance between vehicles, $\max_{i,j\in\mathcal{I}} \|\mathbf{p}^{[i]} - \mathbf{p}^{[j]}\|$, is in $\Theta(\sqrt{|\mathcal{E}(n)|})$. Thus, one vehicle must travel a distance of at least $\frac{1}{2}\max_{i,j\in\mathcal{I}} \|\mathbf{p}^{[i]} - \mathbf{p}^{[j]}\| \in \Theta(\sqrt{|\mathcal{E}(n)|})$. Hence, w.h.p., the completion time is in $\Theta(\sqrt{|\mathcal{E}(n)|})$. □

Remark 3.10 (Drawbacks of Rendezvous Strategy)**.** *From Theorem 3.9 we see that the* Rendezvous Strategy *has better worst-case performance than any monotonic algorithm. Thus, there may be applications in which this is the best algorithm for solving the target assignment problem. However, there are several drawbacks to the algorithm. First, this approach is not a distributed solution in the sense that it requires each vehicle to acquire information about all other vehicles in the group, and to solve a centralized assignment problem. Second, the process of meeting to exchange information creates a single point of failure for the system. Third, if we consider an initial configuration where m targets are occupied, then in the* Rendezvous Strategy *all of these targets become unoccupied as the vehicles travel to the meeting point. Thus, this is not a monotonic algorithm. In fact, if every target is occupied and we run the* Rendezvous Strategy, *all vehicles leave their targets, move to the meeting point, compute a complete assignment, and move to a new target. This is obviously not the desired behavior in this instance. Fourth, the* Rendezvous Strategy *is ill-suited for heterogeneous situations where vehicles have widely distinct speeds, or become active at different instants of time; in these situations the* Rendezvous Strategy *essentially reduces the performance of every vehicle to that of the slowest vehicle. Fifth, the* Rendezvous Strategy *does not work under the local target sensing assumption, whereas we will provide an algorithm later that does. Finally, in settings where more vehicle are available than targets, there is hope to complete the target assignment problem in time that is independent of n. The* Rendezvous Strategy *never achieves this time complexity, whereas we* *will prove this property for one of our proposed algorithms below.* •

Because of the drawbacks mentioned in the previous remark, in the remainder of this chapter, we look at distributed monotonic algorithms and their performance in solving the target assignment problem.

3.4 A Constant Factor Monotonic Algorithm in Sparse Environments

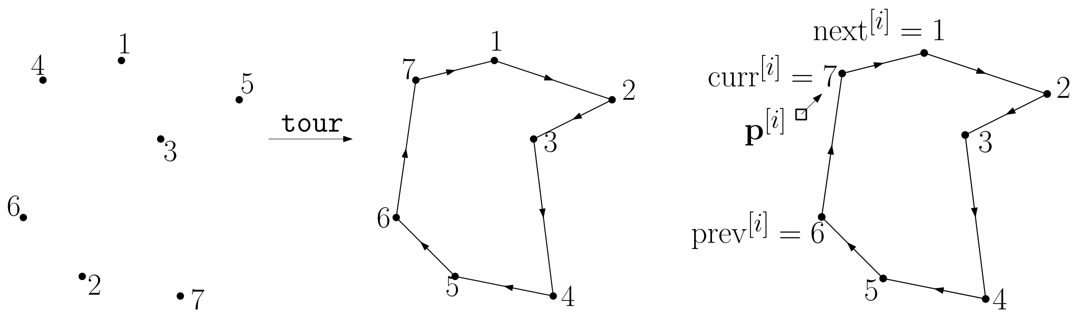
We begin by introducing a monotonic algorithm, called the ETSP Assgmt algorithm, for solving the target assignment problem. This algorithm operates only under the full target knowledge assumption. In this algorithm, each vehicle precomputes an optimal tour through the n targets, turning the cloud of target points into an ordered ring. Vehicles then move along the ring, looking for the next available target. When vehicles communicate, they exchange information on the next available target along the ring. We show that in sparse or critical environments, the ETSP Assgmt algorithm is within a constant factor of the optimal monotonic algorithm for worst-case initial conditions.

3.4.1 The ETSP Assgmt Algorithm

The ETSP Assgmt algorithm is designed under the full target knowledge assumption. In the following description it will be convenient to assume that the target positions are stored in each vehicles memory as an array, rather than as an unordered set. That is, we replace the target set $\mathcal{Q}$ with the target n-tuple $\mathbf{q} := (\mathbf{q}_1, \dots, \mathbf{q}_n)$, and the local target set $\mathcal{Q}^{[i]}$ with the n-tuple $\mathbf{q}^{[i]} := \mathbf{q}$.[4] The algorithm can be described as follows. For each $i \in \mathcal{I}$, vehicle i computes a constant factor approximation of the optimal ETSP tour of the n targets in $\mathbf{q}^{[i]}$ (as discussed in Section 2.2.2), denoted $\mathrm{tour}(\mathbf{q}^{[i]})$. We can think of tour as a permutation that reorders the entries of $\mathbf{q}^{[i]}$. This permutation is independent of i since all vehicles use the same method. An example is shown in Fig. 3.2(a).

Vehicle i then replaces its n-tuple $\mathbf{q}^{[i]}$ with $\mathrm{tour}(\mathbf{q}^{[i]})$. Next, vehicle i computes the index of the closest target in $\mathbf{q}^{[i]}$, and calls it $\mathrm{curr}^{[i]}$. Vehicle i also maintains the index of the next target in the tour that may be available, $\mathrm{next}^{[i]}$, and first target in the tour before $\mathrm{curr}^{[i]}$ that may be available, $\mathrm{prev}^{[i]}$. Thus, $\mathrm{next}^{[i]}$ is initialized to $\mathrm{curr}^{[i]} + 1 \pmod n$ and $\mathrm{prev}^{[i]}$ to $\mathrm{curr}^{[i]} - 1 \pmod n$. This is depicted in Fig. 3.2(b). Vehicle i also maintains the n-tuple, $\mathrm{status}^{[i]}$, which records whether a target is occupied by (assigned to) another vehicle or not.

[4]It is possible that the order of the targets in the local sets $\mathbf{q}^{[i]}$ may initially be different. However, given a set of distinct points in $\mathbb{R}^2$, it is always possible to create a unique ordering.



(a) ETSP tour of seven targets. (b) Vehicle i initialization.

Figure 3.2: The initialization process for the ETSP Assgmt algorithm.

Letting status$^{[i]}(j)$ denote the jth entry in the n-tuple, the entries are given by

$$\text{status}^{[i]}(j) = \begin{cases} 0, & \begin{array}{l}\text{if vehicle } i \text{ knows } \mathbf{q}_j^{[i]} \text{ is assigned} \\ \text{to another vehicle,}\end{array} \\ 1, & \text{otherwise.} \end{cases} \tag{3.2}$$

Thus, status$^{[i]}$ is initialized as the n-tuple $(1, \dots, 1)$. The initialization is summarized in Algorithm A.1 of Appendix A.

Vehicle i then moves toward the target curr$^{[i]}$ at constant speed $v_{\max} > 0$:

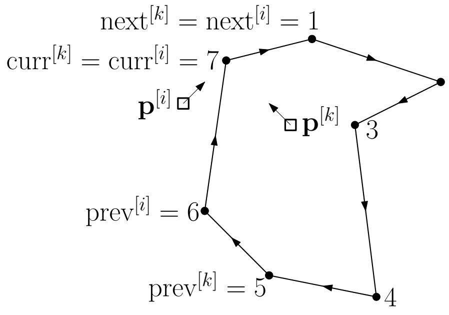
$$\dot{\mathbf{p}}^{[i]} = \begin{cases} v_{\max} \frac{\mathbf{q}^{[i]}_{\text{curr}^{[i]}} - \mathbf{p}^{[i]}}{\|\mathbf{q}^{[i]}_{\text{curr}^{[i]}} - \mathbf{p}^{[i]}\|}, & \text{if } \mathbf{q}^{[i]}_{\text{curr}^{[i]}} \neq \mathbf{p}^{[i]}, \\ 0, & \text{otherwise,} \end{cases} \tag{3.3}$$

Finally, at each communication round vehicle i executes the algorithm comm-rd displayed in Algorithm A.2 of Appendix A. The comm-rd algorithm operates as follows: Vehicle i, which is heading toward target curr$^{[i]}$, communicates with its neighbors to determine if any other vehicles are heading toward curr$^{[i]}$. If another vehicle is heading to curr$^{[i]}$, then the vehicle closer to curr$^{[i]}$ continues moving toward the target, while the farther vehicle selects a new target along the tour (ties are broken using uid's). The vehicles also exchange information on targets that are occupied using the prev, and next variables. The following is a more formal description that omits a few minor technicalities.

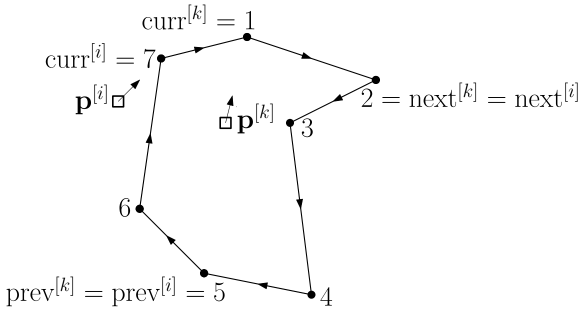
Fig. 3.3 gives an example of comm-rd resolving a conflict between vehicles i and k, over curr$^{[i]}$ = curr$^{[k]}$. In this figure, all other vehicles are omitted. In summary, the ETSP Assgmt algorithm is the triplet consisting of the initialization of each vehicle (see Algorithm A.1), the motion law in Eq. (3.3), and comm-rd (see Algorithm A.2), which is executed at each communication

Description of COMM-RD for vehicle i

```
1 Broadcast msg$^{[i]}$, consisting of UID$^{[i]}$, the target indices prev$^{[i]}$, curr$^{[i]}$, and
  next$^{[i]}$, and the distance to the current target, dist$^{[i]}$.
2 foreach message, msg$^{[k]}$, received do
3     Set status$^{[i]}(j)$ to assigned ('0') for each target $j$ from
      prev$^{[k]} + 1 \pmod n$ to next$^{[k]} - 1 \pmod n$ not equal to curr$^{[i]}$.
4     if curr$^{[i]}$ = curr$^{[k]}$ and dist$^{[i]}$ > dist$^{[k]}$ then
5         Set the status of curr$^{[i]}$ to assigned ('0').
6     else if curr$^{[i]}$ = curr$^{[k]}$ and dist$^{[i]}$ < dist$^{[k]}$ then
7         Leave curr$^{[i]}$ unchanged. However, vehicle $k$ will set curr$^{[k]}$ to a new
          target. This target will be at least as far along the tour as the
          farther of next$^{[i]}$ and next$^{[k]}$. So, set the status of next$^{[i]}$ and next$^{[k]}$
          to assigned ('0').
8 Update curr$^{[i]}$ to the next target in the tour with status available ('1'),
  next$^{[i]}$ to the next available target in the tour after curr$^{[i]}$, and prev$^{[i]}$ to
  the first available target in the tour before curr$^{[i]}$.
```


$\text{next}^{[k]} = \text{next}^{[i]} = 1$
$\text{curr}^{[k]} = \text{curr}^{[i]} = 7$
$\mathbf{p}^{[i]}$
$\mathbf{p}^{[k]}$
2
3
$\text{prev}^{[i]} = 6$
$\text{prev}^{[k]} = 5$
4

(a) Before conflict over target 7.


$\text{curr}^{[k]} = 1$
$\text{curr}^{[i]} = 7$
$\mathbf{p}^{[i]}$
$\mathbf{p}^{[k]}$
$2 = \text{next}^{[k]} = \text{next}^{[i]}$
3
6
$\text{prev}^{[k]} = \text{prev}^{[i]} = 5$
4

(b) After resolution of the conflict.

Figure 3.3: The resolution of a conflict between vehicles i and k over target 7. Since vehicle k is closer to target 7 than vehicle i, vehicle k wins the conflict.

round.

3.4.2 Performance of the ETSP Assgmt Algorithm

We now present our main result on the ETSP Assgmt algorithm. Section 3.4.3 contains its proof. Recall that the ETSP Assgmt algorithm requires the full target knowledge assumption.

Theorem 3.11 (Worst-case bound for ETSP Assgmt)**.** *For any initial positions of n vehicles and n targets in $\mathcal{E}(n)$,* ETSP Assgmt *solves the target assignment problem in $O(\sqrt{n|\mathcal{E}(n)|})$ time. In addition, if $\mathcal{E}(n)$ is sparse or critical, then* ETSP Assgmt *is within a constant factor of the optimal monotonic algorithm for worst-case initial positions.*

3.4.3 Proof s for Statements about the ETSP Assgmt Algorithm

To prove Theorem 3.11 we introduce a few definitions. We say that vehicle $i \in \mathcal{I}$ is *assigned* to target $\mathbf{q}_j^{[i]}$, $j \in \mathcal{I}$, when $\text{curr}^{[i]} = j$. In this case, we also say target j is assigned to vehicle i. We say that vehicle $i \in \mathcal{I}$ *enters a conflict* over the target $\text{curr}^{[i]}$, when vehicle i receives a message, $\text{msg}^{[k]}$, with $\text{curr}^{[i]} = \text{curr}^{[k]}$. Vehicle i *loses the conflict* if vehicle i is farther from $\text{curr}^{[i]}$ than vehicle k, and *wins the conflict* if vehicle i is closer to $\text{curr}^{[i]}$ than vehicle k, where ties are broken by comparing UIDs.

The following lemma is a direct result of the facts that the environment is bounded for each $n \in \mathbb{N}$, and that the vehicles move at constant speed $v_{\max} > 0$.

Lemma 3.12 (Conflict in finite time)**.** *Consider any communication range $r_{\text{comm}} > 0$, and any fixed number of vehicles $n \in \mathbb{N}$. If, for two vehicles i and k, $\text{curr}^{[i]} = \text{curr}^{[k]}$ at some time $t_1 \geq 0$, then vehicle i (and likewise, vehicle k) will enter a conflict over $\text{curr}^{[i]}$ in finite time.*

In order to prove correctness, we require a few properties of the ETSP Assgmt algorithm.

Lemma 3.13 (ETSP Assgmt properties)**.** *During an execution of the* ETSP Assgmt *algorithm, the following statements hold for vehicle $i \in \mathcal{I}$:*

(i) the current target $\text{curr}^{[i]}$ satisfies $\text{status}^{[i]}(\text{curr}^{[i]}) = 1$;

(ii) $\text{status}^{[i]}(j) = 0$ for each $j \in \{\text{prev}^{[i]} + 1, \text{prev}^{[i]} + 2, \dots, \text{next}^{[i]} - 1\} \setminus \{\text{curr}^{[i]}\} \pmod n$;

(iii) $\text{status}^{[i]}(j) = 0$ only if target j is assigned to some vehicle $k \neq i$;

(iv) *if* $\mathrm{status}^{[i]}(j) = 0$ *at some time* t_1*, then* $\mathrm{status}^{[i]}(j) = 0$ *for all* $t \geq t_1$*;*

(v) *if vehicle* i *receives* $\mathrm{msg}^{[k]}$ *during a communication round, then vehicle* i *will set* $\mathrm{status}^{[i]}(j) = 0$ *for each* $j \in \{\mathrm{prev}^{[k]} + 1, \ldots, \mathrm{next}^{[k]} - 1\} \setminus \{\mathrm{curr}^{[i]}\} \pmod n$*.*

Proof. Statements (i) and (iv) and (v) follow directly from the initialization and COMM-RD.

Statement (ii) is initially satisfied since $\mathrm{prev}^{[i]}+1 = \mathrm{curr}^{[i]} = \mathrm{next}^{[i]}-1$ implies that $\{\mathrm{prev}^{[i]} + 1, \ldots, \mathrm{next}^{[i]} - 1\} \setminus \{\mathrm{curr}^{[i]}\} = \emptyset$. Assume that statement (ii) is satisfied before the execution of COMM-RD. At the end of COMM-RD, $\mathrm{prev}^{[i]}$ is updated to the first target before $\mathrm{curr}^{[i]}$ in the tour with status available ('1'). If $\mathrm{status}^{[i]}(\mathrm{curr}^{[i]}) = 1$, then $\mathrm{curr}^{[i]}$ remains unchanged. If $\mathrm{status}^{[i]}(\mathrm{curr}^{[i]}) = 0$, then $\mathrm{curr}^{[i]}$ is increased to the first target with status available ('1'). Finally, $\mathrm{next}^{[i]}$ is set to the first target after $\mathrm{curr}^{[i]}$ that is available. Thus, at the end of COMM-RD the status of $\mathrm{prev}^{[i]}$, $\mathrm{curr}^{[i]}$ and $\mathrm{next}^{[i]}$ are available, and $\mathrm{status}^{[i]}(j) = 0$ for each target $j \in \{\mathrm{prev}^{[i]} + 1, \ldots, \mathrm{next}^{[i]} - 1\} \setminus \{\mathrm{curr}^{[i]}\} \pmod n$.

Statement (iii) is also initially satisfied since $\mathrm{status}^{[i]} = \mathbf{1}_n$ for each $i \in \mathcal{I}$. Assume Statement (iii) is satisfied before the execution of COMM-RD and that during this communication round vehicle i changes the status of a target j to assigned ('0'). We show that Statement (iii) is still satisfied upon completion of the execution of COMM-RD. In order for $\mathrm{status}^{[i]}(j)$ to be changed, vehicle i must have received a message, $\mathrm{msg}^{[k]}$, for which one of the following cases is satisfied: (1) Target $j \neq \mathrm{curr}^{[i]}$ lies between $\mathrm{prev}^{[k]}$ and $\mathrm{next}^{[k]}$ on the tour; (2) There is a conflict between vehicles i and k over target j that vehicle i loses; or, (3) There is a conflict between vehicles i and k that vehicle i wins and $\mathrm{next}^{[i]} = j$ or $\mathrm{next}^{[k]} = j$.

In Case (1) either $\mathrm{status}^{[k]}(j) = 0$ or $\mathrm{curr}^{[k]} = j$, and thus target j is assigned. In Case (2) vehicle k won the conflict implying $\mathrm{curr}^{[k]} = j$ entering the communication round. Thus after the communication round, $\mathrm{curr}^{[i]} \neq j$ and target j is assigned to another vehicle. In Case (3), $\mathrm{curr}^{[i]} = \mathrm{curr}^{[k]} \neq j$, and vehicle k loses the conflict. In this case, vehicle k will change $\mathrm{curr}^{[k]}$ to the next available target on its tour. All targets from $\mathrm{prev}^{[k]} + 1$ to $\mathrm{next}^{[k]} - 1$ have been assigned. Also, during the communication round, vehicle k will receive $\mathrm{msg}^{[i]}$ and determine that all targets from $\mathrm{prev}^{[i]} + 1$ to $\mathrm{next}^{[i]} - 1$ are assigned. Thus, the next available target is at least as far along the tour as the farther of $\mathrm{next}^{[i]}$ and $\mathrm{next}^{[k]}$. Thus, after the communication round, both $\mathrm{next}^{[i]}$ and $\mathrm{next}^{[k]}$ are assigned. □

We are now ready to prove Theorem 3.11.

Theorem 3.11. We begin by proving the correctness of the ETSP Assgmt algorithm. Assume by way of contradiction that at some time $t_1 \geq 0$ there are

$J \in \{1, \dots, n-1\}$ targets unassigned, and for all time $t \geq t_1$, J targets remain unassigned. Since the algorithm is monotonic, the same $n - J$ assigned targets remain assigned for all time, and thus it must be the same J targets that remain unassigned for all $t \geq t_1$. Let $\mathcal{J}$ denote the index set of the J unassigned targets. From our assumption, and by Lemma 3.13 (iii), for every $t \geq t_1$ and for every $i \in \mathcal{I}$, $\text{status}^{[i]}(j) = 1$ for each $j \in \mathcal{J}$. Now, among the $n - J$ assigned targets there is at least one target to which two or more vehicles are assigned. Consider one such target, call it j_1, and consider an vehicle i_1 with $\text{curr}^{[i_1]} = j_1$. By Lemma 3.12, vehicle i_1 will enter a conflict over j_1 in finite time. Let us follow the loser of this conflict. The losing vehicle, call it i_2, will set $\text{status}^{[i_2]}(j_1) = 0$ and will move to the next target in the tour it believes may be available, call it j_2. Now, we know j_2 is not in $\mathcal{J}$, for if it were $J - 1$ targets would be unassigned contradicting our assumption. Moreover, by Lemma 3.13 (i), $j_2 \neq j_1$. Thus, vehicle i_2 will enter a conflict over j_2 in finite time. After this conflict the losing vehicle, call it i_3, will set $\text{status}^{[i_3]}(j_2) = 0$ (because it lost the conflict), and from Lemma 3.13 (v), $\text{status}^{[i_3]}(j_1) = 0$. Again, vehicle i_3's next target j_3 must not be in $\mathcal{J}$, for if it were we would have a contradiction. Thus, repeating this argument $n - J$ times we have that vehicle i_{n-J} loses a conflict over j_{n-J}. After this conflict, we have $\text{status}^{[i_{n-J}]}(j_k) = 0$ for each $k \in \{1, \dots, n-J\}$, where $j_{k_1} = j_{k_2}$ if and only if $k_1 = k_2$. In other words, vehicle i_{n-J} knows that all $n - J$ assigned targets have indeed been assigned. Also, by our initial assumption, $\text{status}^{[i_{n-J}]}(j) = 1$ for each $j \in \mathcal{J}$. By Lemma 3.13 (i), vehicle i_{n-J}'s new current target must have status available ('1'). Therefore, it must be that vehicle i_{n-J} will set $\text{curr}^{[i_{n-J}]}$ to a target in $\mathcal{J}$. Thus, after a finite amount of time $J - 1$ targets are unassigned, a contradiction.

We now prove the upper bound on the performance of the ETSP Assgmt algorithm. First notice the following: Consider the optimal ETSP tour through all n targets. This provides an ordering in which the n targets are visited. Now, suppose k targets are removed from the tour, and the $n-k$ remaining targets are visited in the order they appeared in the n-target tour. In general, this is not the optimal tour through the $n - k$ points. However, by the triangle inequality, the length of the tour is no longer than that of the tour through all n points. Because of this, in the worst-case some vehicle must travel to its nearest target, and then around its entire ETSP tour, losing a conflict at each of the first $n - 1$ targets in the tour. For any initial vehicle and target positions, the distance to the nearest target is $O(\sqrt{|\mathcal{E}(n)|})$. Since the length of each vehicle's tour is a constant factor approximation of the optimal, the tour length is $O(\sqrt{n\mathcal{E}(n)})$ (see Theorem 2.2). The vehicle will not follow the ETSP tour exactly because it may enter conflicts before actually reaching the targets; however, by the triangle inequality, the resulting path cannot be longer than the ETSP tour. Hence, the total distance traveled is in $O(\sqrt{n\mathcal{E}(n)})$, and since the vehicles move at constant speed, the completion time is in $O(\sqrt{n\mathcal{E}(n)})$. Combining this with Theorem 3.7 we see that in critical or sparse environments the completion time is in $\Theta(\sqrt{n\mathcal{E}(n)})$. $\square$

3.5 A Constant Factor Monotonic Algorithm in Dense Environments

In the previous section we presented the ETSP Assgmt algorithm which operates only with full target knowledge but has provably good performance in sparse and critical environments. In this section we introduce a monotonic algorithm called the Grid Assgmt algorithm which operates under both full target knowledge and local target sensing with $r_{\text{sense}} \geq \sqrt{2/5}r_{\text{comm}}$. In this algorithm, the vehicles partition the environment into cells. Vehicles then determine local maximum assignments, and elect a leader in the cell which they occupy. Through communication between leaders of adjacent cells, each leader obtains estimates of the location of free targets, and uses this information to guide unassigned vehicles to free targets. We show that in critical or dense environments, the Grid Assgmt algorithm is within a constant factor of the optimal monotonic algorithm for worst-case initial conditions. In addition, we characterize the stochastic performance of the Grid Assgmt algorithm.

3.5.1 The Grid Assgmt Algorithm

In the Grid Assgmt algorithm we make either the full target knowledge assumption (i.e., $\mathcal{Q}^{[i]} := \mathcal{Q}$), or the local target sensing assumption with $r_{\text{sense}} \geq \sqrt{2/5}r_{\text{comm}}$. In addition we assume each vehicle knows the environment $\mathcal{E}(n)$. Each vehicle partitions the environment into b^2 equally sized square cells, where $b \in \mathbb{N}$. It then labels the cells like entries in a matrix, so cell $C(r,c)$ resides in the rth row and cth column, as shown in Fig. 3.4. Since the vehicles started with the same information, they all create the same partition. The quantity b is chosen so that an vehicle in cell $C(r,c)$ is within communication range of any vehicle in cells $C(r,c)$, $C(r-1,c)$, $C(r+1,c)$, $C(r,c-1)$, and $C(r,c+1)$. In light of Lemma 3.2, we see that this is satisfied when $b = \lceil\sqrt{5|\mathcal{E}(n)|}/r_{\text{comm}}\rceil$. Note that with $r_{\text{sense}} \geq \sqrt{2/5}r_{\text{comm}}$ an vehicle in cell $C(r,c)$ can sense the position of all targets in that cell. We now outline the Grid Assgmt algorithm.

Outline of the Grid Assgmt algorithm

***Initialization and role assignment*:** Each vehicle partitions the environ-

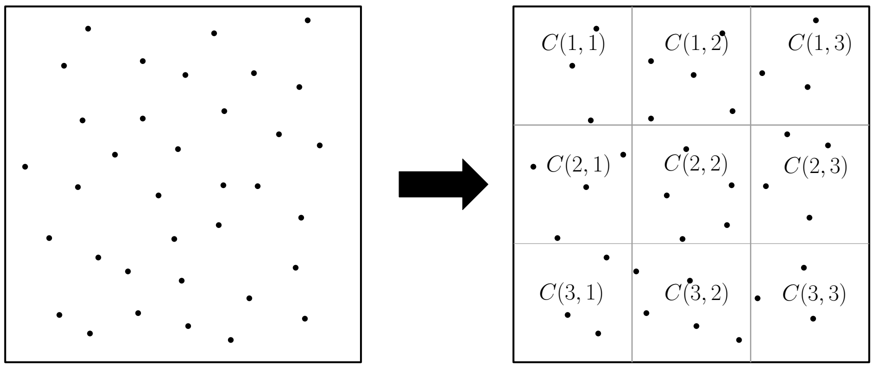


Figure 3.4: Partitioning $\mathcal{E}(n)$, containing 35 targets, into $b^2 = 9$ cells.

ment as described above. In each cell, vehicles find a maximum assignment between vehicles and targets occupying the cell, and assigned vehicles elect a leader among them. Accordingly, vehicles are labeled leader, unassigned, or assigned non-leader. According to their role, vehicles allocate certain variables describing their location and their knowledge about target assignments.

Assigned non-leader vehicles: Each assigned non-leader vehicle move to its assigned target and goes silent.

Cell leaders: Each cell leader estimates the number of available targets in all cells below it in its column. The leader i of cell $C(r,c)$ stores this estimate in the variable $\Delta^{[i]}_{\text{blw}}(r,c)$; to maintain the estimates, cell leaders communicate to the cell leader in the cell directly above it. Additionally, each cell leader in the top row communicates to the cell leader in the cell directly to the right, to obtain an estimate of the number of available targets in all columns to the right (denoted $\Delta^{[j]}_{\text{rght}}(1,c)$ for leader j of cell $C(1,c)$).

Unassigned vehicles: Each unassigned vehicle seeks a free target by entering cells and querying their respective leaders. The motion of unassigned vehicles is illustrated in Fig. 3.5. Assuming no communication with the leaders, the nominal order in which an unassigned vehicle visits all cells of the grid is shown in the left-hand figure. The way in which this path is shortened as the unassigned vehicle receives available target estimates from cell leaders is shown on the right-hand figure.

Remark 3.14 (Computations performed by cell leaders)**.** *If vehicle i is the leader of cell $C(r,c)$, it computes $\Delta^{[i]}(r,c)$, which is (# of targets) − (# of vehicles) in $C(r,c)$. In addition, leader i maintains $\Delta^{[i]}_{\text{blw}}(r,c)$, which is an estimate of (# of targets) − (# of vehicles) in cells $C(r+1,c)$ to $C(b,c)$. This quantity must*

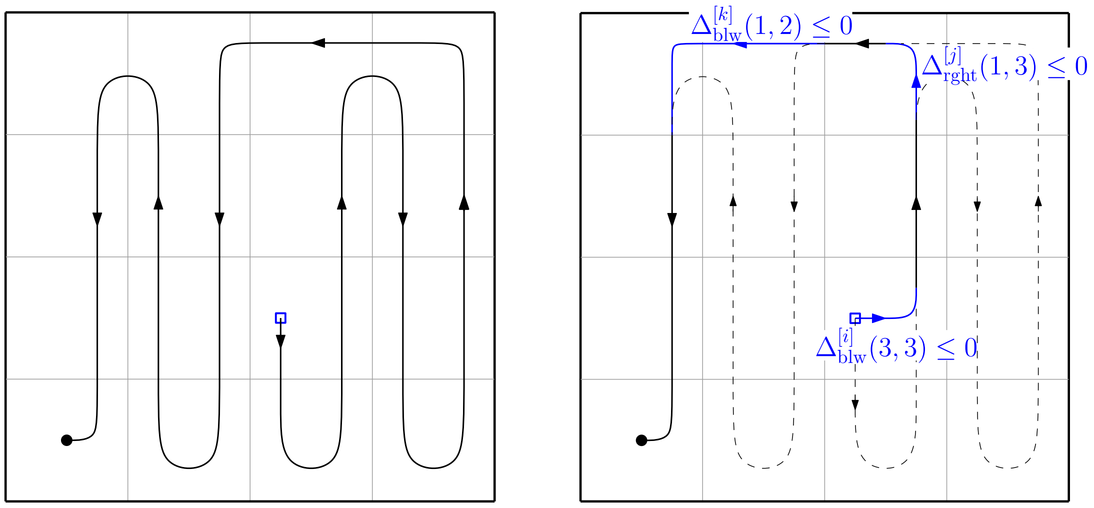

$\Delta^{[k]}_{\text{blw}}(1,2) \leq 0$
$\Delta^{[j]}_{\text{rght}}(1,3) \leq 0$
$\Delta^{[i]}_{\text{blw}}(3,3) \leq 0$

Figure 3.5: The left figure shows the nominal order in which an vehicle (blue square) searches the cells in the absence of communication. The blue lines on the right figure show how this path is shortened by the non-positive estimates from leader i of $C(3,3)$, leader j of $C(1,3)$ and leader k of $C(1,2)$.

be estimated because vehicle i does not initially know the number of vehicles in cells $C(r+1,c)$ to $C(b,c)$. The variable $\Delta^{[i]}_{\text{blw}}(r,c)$ is initialized to $+\infty$ (i.e., a very large positive number) for the leaders in rows 1 to $b-1$, and to 0 for the leaders in row b. Then, at each communication round vehicle i updates its estimate by communicating with the leaders in cells $C(r-1,c)$ and $C(r+1,c)$:

```
1  Send $\text{msg}^{[i]} := \Delta^{[i]}_{\text{blw}}(r,c) + \Delta^{[i]}(r,c)$ to leader in cell $C(r-1,c)$ and receive $\text{msg}^{[k]}$ from
     vehicle $k$, the leader of $C(r+1,c)$.
2  Set $\Delta^{[i]}_{\text{blw}}(r,c) := \text{msg}^{[k]} = \Delta^{[k]}_{\text{blw}}(r+1,c) + \Delta^{[k]}(r+1,c)$.
```

*The update procedure is depicted in Fig. 3.6. A leader j of cell $C(1,c)$ in the top row uses a similar method to maintain the estimate $\Delta^{[j]}_{\text{rght}}(1,c)$. It should be noted that as unassigned vehicles enter and exit cells, the actual values of Δ_{blw} and Δ_{rght} change. Thus, to maintain accurate estimates, there is a procedure whereby vehicles send **`enter`** and **`exit`** messages to cell leaders. This is detailed in Appendix B.* •

Remark 3.15 (Motion performed by unassigned vehicles)**.** *Let us describe the unassigned vehicles motion in more detail. First, each unassigned vehicle seeks a free target in its column as follows. It queries the leader of its current cell about free targets in its column, below its current cell. If the leaders estimate $\Delta^{[i]}_{\text{blw}}(r,c)$ is positive, then the vehicle moves down the column. Otherwise, the vehicle moves*

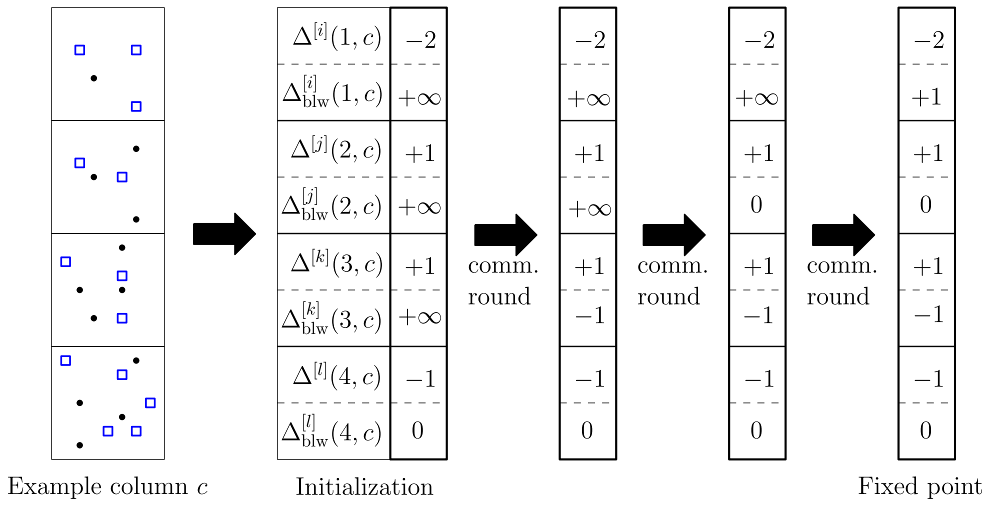

$\Delta^{[i]}(1,c)$
$\Delta^{[i]}_{\mathrm{blw}}(1,c)$
$\Delta^{[j]}(2,c)$
$\Delta^{[j]}_{\mathrm{blw}}(2,c)$
$\Delta^{[k]}(3,c)$
$\Delta^{[k]}_{\mathrm{blw}}(3,c)$
$\Delta^{[l]}(4,c)$
$\Delta^{[l]}_{\mathrm{blw}}(4,c)$
-2 $+\infty$ $+1$ $+\infty$ $+1$ $+\infty$ -1 0
comm. round
-2 $+\infty$ $+1$ $+\infty$ $+1$ -1 -1 0
comm. round
-2 $+\infty$ $+1$ 0 $+1$ -1 -1 0
comm. round
-2 $+1$ $+1$ 0 $+1$ -1 -1 0
Example column c
Initialization
Fixed point

Figure 3.6: In the column c, blue squares depict vehicles, and black disks depict targets. The figure shows how the estimates Δ and Δ_{blw} are initialized and updated by leader i of $C(1,c)$, j of $C(2,c)$, k of $C(3,c)$, and l of $C(4,c)$. The estimates converge to true values in three communication rounds.

up the column. While moving down, upon entering a new cell the vehicle first queries the cell leader on free targets in the cell, and then on free targets in cells below. If the vehicle starts moving up the column, then it only queries cell leaders on free targets in the cell (since it knows no targets are free in the cells below).

Second, if the vehicle reaches the top cell of its column, then the column contains no free targets. To transfer to a new column, the vehicle queries the leader of the top cell about free targets in all columns to the right. If the leader's estimate $\Delta^{[j]}_{\mathrm{blw}}(1,c)$ is positive, then the vehicle moves to the right; otherwise, the vehicle moves to the left. Upon reaching the next cell, the vehicle recommences the column procedure. •

A detailed description of the Grid Assgmt algorithm is given in Appendix B: all variables maintained by the vehicles are listed in Table B.1; the initialization and role assignment is performed by the Role Assgmt algorithm, see Algorithm B.1; the behavior of the cell leaders and of the unassigned vehicles are described by the Leader and Unassigned algorithms, see Algorithms B.2 and B.3, respectively.

Remark 3.16 (Using a single transfer row). *In our description of the* Grid Assgmt *algorithm, vehicles use the top row to transfer to a new column. This choice of "transfer row" is arbitrary and the top row was chosen for simplicity of presentation. Intuitively, it seems the middle row is a more efficient choice. The upcoming analysis shows that such a choice does not affect the algorithm's* *asymptotic performance. The reason we require unassigned vehicles to use a single transfer row is because it allows for cell leaders to easily maintain up-to-date estimates of unassigned vehicle and free target locations. To understand this, suppose that there were two transfer rows, row* 1 *and row* b*, and that two unassigned vehicles simultaneously transfer from column* $c-1$ *to column* c*, one using row* 1*, and the other using row* b*. Then, it would take* $b \in \Theta(\sqrt{|\mathcal{E}(n)|})$ *communication rounds for the leader in cell* $C(1,c)$ *to become aware that an unassigned vehicle transferred using row* b*, implying that leader estimates are not up-to-date. To overcome this, one would need to halt unassigned vehicle motion until leader estimates have been updated; a process which would require more leader communication. In addition, using more transfer rows does not appear to change the asymptotic performance (although the constant factor could be significantly reduced since the algorithm would rely more heavily on communication than vehicle motion). Thus, we have utilized a single transfer row to minimize excess communication, and avoid introducing more complexity in the algorithm. This also reduces wireless congestion, which can become significant for large numbers of vehicles (Gupta and Kumar, 2000).* •

Remark 3.17 (Details of the Grid Assgmt algorithm)**.** *(1) Vehicles move at speed* $v_{\max}$*, and to transfer between cells vehicles move toward the center of the new cell. (2) If an vehicle or target lies on the boundary between cells, a simple tie breaking scheme is used assign it to a cell. (3) In our presentation, we implicitly assumed that every cell initially contains at least one vehicle and one target. If a cell has no targets, then any vehicles initially in the cell leave, and the empty cell is then ignored. If a cell initially contains targets but no vehicles, then the first vehicles to enter the cell run the* Maximal Match *algorithm and a leader is elected.* •

3.5.2 Performance of the Grid Assgmt Algorithm

We now present our main results on the Grid Assgmt algorithm. Section 3.5.3 contains their proofs. Recall that the Grid Assgmt algorithm operates under full target knowledge, or local target sensing with $r_{\text{sense}} \geq \sqrt{2/5}r_{\text{comm}}$.

Theorem 3.18 (Worst-case bound for Grid Assgmt)**.** *For any initial positions of* n *vehicles and* n *targets in* $\mathcal{E}(n)$*, the* Grid Assgmt *algorithm solves the target assignment problem in* $O(|\mathcal{E}(n)|)$ *time. In addition, if* $\mathcal{E}(n)$ *is dense or critical, then the* Grid Assgmt *algorithm is within a constant factor of the optimal monotonic algorithm for worst-case initial conditions.*

Remark 3.19 (Grid Assgmt vs. ETSP Assgmt)**.** *The worst-case bound for the* ETSP Assgmt *algorithm in Theorem 3.11 was* $O(\sqrt{|\mathcal{E}(n)|n})$*. Thus, in* *sparse environments the* ETSP Assgmt *algorithm performs better, where as in dense environments the* Grid Assgmt *algorithm performs better. In critical environments, the bounds are equal. Thus, the two algorithms are complementary. In practice, a vehicle can determine which algorithm to run by comparing the area of the environment $|\mathcal{E}(n)|$ to the area of n disks of radius r_{comm}. That is, given n, $\mathcal{E}(n)$ and r_{comm}, a vehicle could use a rule such as the following: if $|\mathcal{E}(n)| > \pi r_{\mathrm{comm}}^2 n$, then execute the* ETSP Assgmt *algorithm, else if $|\mathcal{E}(n)| < \pi r_{\mathrm{comm}}^2 n$, then execute the* Grid Assgmt *algorithm.* •

The following theorem shows that for randomly placed targets and vehicles, the performance of the Grid Assgmt algorithm is considerably better than in the worst-case.

Theorem 3.20 (Stochastic time complexity). *Consider n vehicles and n targets, uniformly randomly distributed in $\mathcal{E}(n)$. Then, the* Grid Assgmt *algorithm solves the target assignment problem in $O(\sqrt{|\mathcal{E}(n)|})$ time with high probability if*

$$|\mathcal{E}(n)| \leq \frac{r_{\mathrm{comm}}^2}{5} \frac{n}{\log n + \gamma(n)},$$

where γ is any function such that $\gamma(n) \to +\infty$ as $n \to +\infty$.

Remark 3.21 (Generalization of Theorem 3.20). *The bound in Theorem 3.20 holds, more generally, for any initial positions such that every cell contains at least one target and at least one vehicle.*

Theorem 3.22 (Stochastic time complexity, cont'd). *Consider n vehicles and $n/\log n$ targets, uniformly randomly distributed in $\mathcal{E}(n)$. Then the* Grid Assgmt *algorithm solves the target assignment problem in $O(1)$ time with high probability if there exists $K > 1/\log(4/\mathrm{e})$, such that*

$$|\mathcal{E}(n)| \leq \frac{r_{\mathrm{comm}}^2}{5} \frac{n}{K \log n}.$$

3.5.3 Proofs for Statements about the Grid Assgmt Algorithm

In this section we prove the results presented in Section 3.5.2. The leaders of each cell maintain estimates of the difference between the number of targets and vehicles in various parts of the grid. In order to talk about the convergence of these estimates we introduce a few quantities. Let $\mathrm{tar}(r,c)$ denote the number of targets in $C(r,c)$. Let $\Delta(r,c)(t)$ denote the difference between $\mathrm{tar}(r,c)$ and the number of vehicles with $\mathrm{currcell}^{[i]} = C(r,c)$ at time $t > 0$. (Notice the lack of superscript on $\Delta(r,c)(t)$, when compared to vehicle i's estimate of the quantity, $\Delta^{[i]}(r,c)$.) Recall that in our model, communication round k occurs instantaneously at time t_k. Thus, we let t_k^- denote start of the round, and t_k^+, its completion, and so $\Delta^{[i]}(r,c)(t_k^+)$ denotes value of $\Delta^{[i]}(r,c)$ at the completion of communication round k.

Lemma 3.23 (Convergence of estimates)**.** *During an execution of the* GRID ASSGMT *algorithm, if vehicle $i \in \mathcal{I}$ is the leader of cell $C(r,c)$ then for each communication time t_k, $k \in \mathbb{N}$:*

(i) $\Delta^{[i]}(r,c)(t_k^+) = \Delta(r,c)(t_k)$;

(ii) $\Delta^{[i]}_{\mathrm{blw}}(r,c)(t_k^+) \geq \sum_{r^*=r+1}^{b} \Delta(r^*,c)(t_k)$;

(iii) if $k > b$ and each cell in column c contains a leader, then $\Delta^{[i]}_{\mathrm{blw}}(r,c)(t_k^+) = \sum_{r^*=r+1}^{b} \Delta(r^*,c)(t_k)$.

Proof. To see part (i) notice that each vehicle $j \in \mathcal{I}$ initially sets currcell$^{[j]}$ to the cell it occupies. The leader of cell $C(r,c)$, call it vehicle i, can communicate with all vehicles in its cell, and it knows the number of targets in $C(r,c)$. Thus, at t_1 vehicle i counts the vehicles in its cell, and correctly calculates $\Delta^{[i]}(r,c)(t_1^+) = \Delta(r,c)(t_1)$. Assume that $\Delta^{[i]}(r,c)$ is correct at t_{k-1}^+. We will show that it is correct at t_k^+. If at t_k^- vehicle j changes currcell$^{[j]}$ to $C(r,c)$, then it must either be in $C(r-1,c)$ or $C(r+1,c)$, or if $r = 1$, possibly $C(r,c-1)$ and $C(r,c+1)$. Upon changing currcell$^{[j]}$ to $C(r,c)$, vehicle j sends an `enter` message to the leader of $C(r,c)$, and by Lemma 3.2 the leader will receive it at t_k. Likewise, if an vehicle changes currcell$^{[j]}$ from $C(r,c)$ to another cell, the vehicle must be in cell $C(r,c)$. Thus, when this vehicle sends the `exit` message, the leader of $C(r,c)$ will receive it at t_k. Hence, after the leader updates $\Delta^{[i]}(r,c)$ (Step 8 of LEADER), it will have $\Delta^{[i]}(r,c)(t_k^+) = \Delta(r,c)(t_k)$.

The proof of (ii) is as follows. Notice that we can write $\sum_{r^*=r+1}^{b} \Delta(r^*,c)(t_k)$ as

$$\sum_{r^*=r+1}^{b} (\Delta(r^*,c)(t_{k-1})) + \mathrm{enter}(t_{k-1},t_k) - \mathrm{exit}(t_{k-1},t_k). \tag{3.4}$$

where enter(t_{k-1},t_k) is the number of vehicles that entered cells $C(r+1,c),\dots,C(b,c)$ between time t_{k-1} and time t_k, and exit(t_{k-1},t_k) is the number that exited.

Let vehicle i be the leader of cell $C(r,c)$. Vehicle i initializes $\Delta^{[i]}_{\mathrm{blw}}(r,c)$ to $+\infty$, so the inequality is satisfied initially. Assume (ii) is satisfied at t_{k-1}^+. We will show that it is satisfied at t_k^+. If there is no leader in $C(r+1,c)$, then vehicle i will not receive a message. In this case one of two updates occurs: 1) If an unassigned vehicle enters cell $C(r,c)$ from cell $C(r+1,c)$ then vehicle i sets

$\Delta^{[i]}_{\mathrm{blw}}(r,c) := 0$ (Step 10 of Leader). But, from Unassigned, an vehicle moves up a column only if there are no available targets below, and thus the inequality is satisfied at t_k^+. Alternatively, 2) vehicle i leaves $\Delta^{[i]}_{\mathrm{blw}}(r,c)$ unchanged, and thus the inequality will be satisfied at t_k^+.

The other case is that leader j is in cell $C(r+1,c)$, and vehicle i receives the message

$$\Delta^{[j]}_{\mathrm{blw}}(r+1,c)(t^+_{k-1}) + \Delta^{[j]}(r+1,c)(t^+_{k-1}), \tag{3.5}$$

But by assumption

$$\Delta^{[j]}_{\mathrm{blw}}(r+1,c)(t^+_{k-1}) \geq \sum_{r^*=r+2}^{b} (\Delta(r^*,c)(t_{k-1}))$$

and from (i), $\Delta^{[j]}(r+1,c)(t^+_{k-1}) = \Delta(r+1,c)(t_{k-1})$. Thus, Eq. (3.5) is no smaller than $\sum_{r^*=r+1}^{b} \Delta(r^*,c)(t_{k-1})$. But, when vehicle i receives the message in Eq. (3.5), it adds $\mathtt{enter}(t_{k-1},t_k)$ and subtracts $\mathtt{exit}(t_{k-1},t_k)$ (see Step 9 of Leader). Thus, from Eq. (3.4), the inequality is satisfied at t_k^+.

In light of the proof for (ii), we see that to prove (iii) we need only show that for all $k \geq b$, the message in Eq. (3.5) equals $\sum_{r^*=r+1}^{b} \Delta(r^*,c)(t_{k-1})$. We do this by induction. Notice that in cell $C(b,c)$, $\Delta^{[j]}_{\mathrm{blw}}(b+1,c)(t^+_{k-1}) = 0$, and so (iii) holds trivially for $k>0$. In cell $C(b-1,c)$, for $k>1$, the message in Eq. (3.5) becomes $\Delta^{[j]}(b,c)(t^+_{k-1})$, which by (i) equals $\Delta(b,c)(t_{k-1})$. Thus (iii) holds for cell $C(b-1,c)$ and $C(b,c)$ for all $k>1$. Assume that (iii) holds for $C(r+1,c),\ldots,C(b,c)$ at time t^+_{k-1}, where $k > b-r$. We will show it holds for $C(r,c)$ at time t_k^+. Since (iii) holds for cell $C(r+1,c)$ at t_{k-1}, the first term in Eq. (3.5) is $\sum_{r^*=r+2}^{b} \Delta(r^*,c)(t_{k-1})$, and from (i), the second term is $\Delta(r+1,c)(t_{k-1})$. Thus, the message is $\sum_{r^*=r+1}^{b} \Delta(r^*,c)(t_{k-1})$. □

We have an analogous result for the convergence of $\Delta^{[i]}_{\mathrm{rght}}(r,c)$. It follows directly from Lemma 3.23 (i) and (iii) and the fact that $\Delta^{[i]}_{\mathrm{rght}}(c)$ is initially overestimated.

Lemma 3.24 (Convergence of estimates, cont'd)**.** *If vehicle $i \in \mathcal{I}$ is the leader of cell $C(1,c)$, then for each communication time t_k, $k \in \mathbb{N}$,*

(i) $\Delta^{[i]}_{\mathrm{rght}}(c)(t_k^+) \geq \sum_{c^*=c+1}^{b} \sum_{r^*=1}^{b} \Delta(r^*,c^*)(t_k)$;

(ii) if each cell contains a leader and if $k > 2b$, then

$$\Delta^{[i]}_{\mathrm{rght}}(c)(t_k^+) = \sum_{c^*=c+1}^{b} \sum_{r^*=1}^{b} \Delta(r^*,c^*)(t_k).$$

We will now prove theorems 3.18, 3.20, and 3.22.

Theorem 3.18. We begin by proving the correctness of the GRID ASSGMT algorithm. Assume by way of contradiction that $J \in \{1, \dots, n-1\}$ targets remain unassigned for all time and thus $B \in \{1, \dots, J\}$ cells contain unassigned targets. By construction of the GRID ASSGMT algorithm, an assigned target never becomes unassigned. Thus, the same targets remain unassigned for all time. Let $\mathcal{C}$ denote the set of cells containing these unassigned targets.

Consider a cell $C(r, c) \in \mathcal{C}$. If $C(r, c)$ does not contain a leader, then it has never been entered by an vehicle. If it does contain a leader, then $\text{taravail}^{[i]}(r, c)$ contains the available targets. Thus if there is an unassigned vehicle in cell $C(r, c) \in \mathcal{C}$, then upon querying the leader (or if there is no leader, electing a leader), at least one of the targets in $C(r, c)$ will become assigned, contradicting our assumption. Likewise, for each cell $C(r, c) \notin \mathcal{C}$, either there is a leader and $\text{taravail}^{[i]}(r, c) = \emptyset$, or there are no targets in the cell.

Now, consider an unassigned vehicle i, in cell $C(r, c) \notin \mathcal{C}$. Vehicle i must never enter a cell in $\mathcal{C}$, for if it did an unassigned target would become assigned, a contradiction. We will show this is not possible. According to the UNASSIGNED algorithm, vehicle i travels down its current column, querying the leader of each cell for available targets in the cell and in cells below. By Lemma 3.23(ii) vehicle i will only travel back up the column if all targets in cells below have been assigned. After traveling back up the column, if there are no available targets in the top cell in the column, vehicle i will set $\text{colstatus}^{[i]}(c) = 0$ and will never enter column c again. By Lemmas 3.23 and 3.24, vehicle i will travel down each column that may possibly have a free target. Thus, at some point vehicle i necessarily will enter a column containing a cell in $\mathcal{C}$. Hence, either vehicle i, or another assigned vehicle will enter the cell in $\mathcal{C}$ at which point the number of assigned targets will increase by at least one, a contradiction.

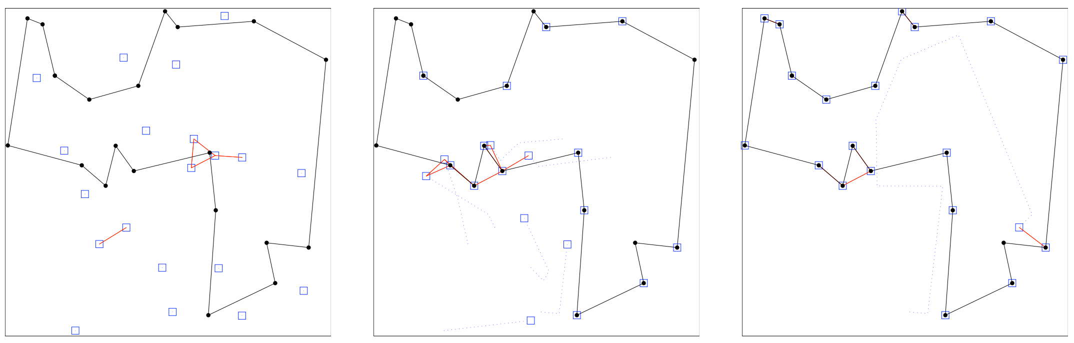
We now prove the upper bound on the performance of the GRID ASSGMT algorithm. In the worst case, the targets are positioned such that leaders cannot exchange any information about availability of targets. Then, in the worst case an vehicle, call it i, must visit all b^2 cells before reaching an unassigned target. In the worst case vehicle i will travel up and down once in every column in the grid, and back and forth once along the top of the grid. In each cell, vehicle i will query the leader for available targets. If there is no leader in the cell, then vehicle i will solve a maximum matching among vehicles that entered at the same time as it, and one of them will become the leader. In either case, the time spent in each cell is $O(1)$. The length of each column is $\sqrt{|\mathcal{E}(n)|}$, and thus the worst-case travel distance is bounded by $2\sqrt{|\mathcal{E}(n)|}(b+2) \in O(|\mathcal{E}(n)|)$. Since the vehicle moves at constant speed $v_{\max}$, the time for the last vehicle to reach its final target is in $O(|\mathcal{E}(n)|)$. □

Theorem 3.20. From Lemma 3.2, $b \leq \lceil\sqrt{n/(\log n + \gamma(n))}\rceil$, where $\gamma(n) \to +\infty$ as $n \to +\infty$. From Theorem 3.1 when we uniformly randomly distribute n targets and n vehicles into b^2 cells, w.h.p. each cell contains at least one vehicle, and one target. The maximum matching and leader election in the ROLE ASSGMT algorithm can be performed in $O(1)$ time. Thus in $O(1)$ time there will be a leader in every cell. By Lemma 3.23(iii), in $b \in O(\sqrt{|\mathcal{E}(n)|}$ communication rounds, every leader will know the difference between the number of vehicles and the number of targets in the cells below it. Thus after $O(\sqrt{|\mathcal{E}(n)|})$ time, the leader of each cell will only let an vehicle move further down the column if it knows the vehicle will find an assignment. Also, by Lemma 3.24(ii) after $O(\sqrt{|\mathcal{E}(n)|})$ time, each leader in the top row will only send vehicles right if there are available targets to the right. Thus, in the worst case, an vehicle may have to travel out of its own column, across the top column, and then down a new column in order to find its target. This distance is in $O(\sqrt{|\mathcal{E}(n)|})$, and since the vehicle spends $O(1)$ time in each cell, the time complexity is in $O(\sqrt{|\mathcal{E}(n)|})$. Thus the total time complexity is in $O(\sqrt{|\mathcal{E}(n)|}) + O(\sqrt{|\mathcal{E}(n)|}) \in O(\sqrt{|\mathcal{E}(n)|})$ time. □

Theorem 3.22. From Lemma 3.2, there are $b^2 \leq \lceil\sqrt{(n/K\log n)}\rceil^2$ cells, where K is a constant satisfying $K > 1/\log(4/\mathrm{e})$. Equivalently, we can write $b^2 = \frac{1}{C(n)}\lceil\sqrt{(n/K\log n)}\rceil^2$, where $C(n) \geq 1$ for all $n \in \mathbb{N}$. From Theorem 3.1(i), when we distribute $n/\log n$ targets into b^2 cells, w.h.p. there are at most $C(n)O\left(\frac{\log n}{\log\log n}\right)$ targets in any given cell. From Theorem 3.1(iv), w.h.p. there are at least $C(n)\Omega(\log n)$ vehicles in each cell. Thus, w.h.p, there are more vehicles than targets in every cell. Thus after running the ROLE ASSGMT algorithm, every target in each cell will be assigned. The maximum matching can be found in $O(1)$ time. Since each cells area is $\leq r_{\mathrm{comm}}^2/5$, and the vehicles move at constant speed, the assignment will be complete in $O(1)$ time, with high probability. □

3.6 Simulations

We have performed extensive simulations of the ETSP ASSGMT and GRID ASSGMT algorithms. The ETSP ASSGMT algorithm has been simulated in both two and three dimensional environments. To compute the ETSP tour we have used the `linkern` TSP solver (see Chapter 2 for a review of TSP computation).



(a) Initial positions. (b) Mid execution. (c) Final assignment.

Figure 3.7: Simulation of the ETSP Assgmt algorithm for 20 vehicles in a sparse environment. Targets are black dots and vehicles are blue squares. The ETSP tour is shown connecting the targets, and a red line is drawn between vehicles within communication range.

A representative simulation for 20 vehicles and targets uniformly randomly placed in a sparse environment is shown in Fig. 3.7. The ETSP tour is shown connecting the target positions. Dashed blue trails in Fig. 3.7(b) and Fig. 3.7(c), give the trajectories of vehicles that have yet to reach a target. A representative simulation of the Grid Assgmt algorithm for 65 vehicles and targets uniformly randomly distributed in a dense environment is shown in Fig. 3.8. In Fig. 3.8(c) the communication between the leaders of each cell is shown with red lines, and a dashed blue trail shows the trajectory for the final vehicle, as it is about to reach its target in cell $C(1,1)$.

Fig. 3.9 contains the numerical outcomes of Monte Carlo simulations for the ETSP Assgmt and Grid Assgmt algorithms with uniformly randomly generated target and vehicle positions. Both sets of simulations were performed for vehicles with $r_{\text{comm}} = 10$ and $v_{\max} = 1$. Each data point is the mean completion time of 30 trials, where each trial was performed at randomly generated vehicle and target positions. Error bars show plus/minus one standard deviation. The simulation for the ETSP Assgmt algorithm in Fig 3.9(a) was performed in a square environment with area $4r_{\text{comm}}^2 n$, and suggests that even for uniformly randomly generated positions, ETSP Assgmt solves the target assignment problem in time proportional to $\sqrt{n|\mathcal{E}(n)|}$. The Monte Carlo simulation for the Grid Assgmt algorithm is shown in Fig. 3.9(b). These simulations were performed in a square environment with area $r_{\text{comm}}^2 n/(6\log n)$, which satisfies the bound in Theorem 3.20. For simplicity of implementation we discard trials in which there exists a cell without targets. This is justified by the fact that w.h.p. every cell contains at least one target, and thus the number of discarded

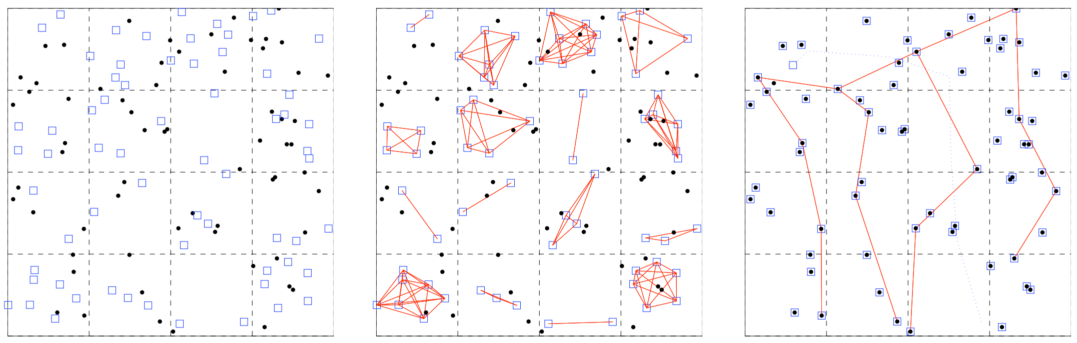

(a) Initial positions. (b) Role assignment. (c) Final assignment.

Figure 3.8: A simulation of 65 vehicles in a dense environment. Targets are black disks and vehicles are blue squares. The partition of $\mathcal{E}(n)$ is shown in dashed lines, and red lines are drawn between communicating vehicles.

trials tends to zero as n increases. The simulation suggests that asymptotically, the expected completion time is bounded below by $1.5\sqrt{|\mathcal{E}(n)|}$ and above by $2.5\sqrt{|\mathcal{E}(n)|}$. This agrees with the $O(\sqrt{|\mathcal{E}(n)|})$ bound in Theorem 3.20 and gives some idea as to the constant in front of this bound.

3.7 Extensions and Summary

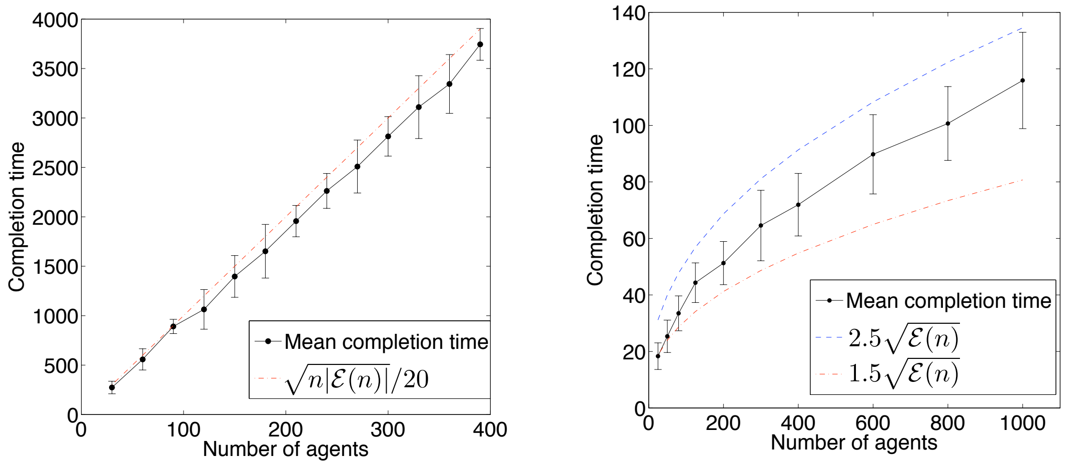
We have attempted to present the ETSP Assgmt and Grid Assgmt algorithms in their most basic forms. In this section we discuss some extensions to these algorithms.

3.7.1 Higher dimensional Spaces

We have presented our algorithms for the environment $\mathcal{E}(n) := [0, \ell(n)]^2 \subset \mathbb{R}^2$. However, these algorithms can be generalized to subsets of $\mathbb{R}^d$, $d \geq 1$. The ETSP Assgmt algorithm we have presented is valid for any environment $\mathcal{E}(n) \subset \mathbb{R}^d$, $d \geq 1$. In (Smith and Bullo, 2007*a*), we have presented time complexity bounds for environments in $\mathbb{R}^d$. In this case, the length of the ETSP tour is bounded by $O(n^{(d-1)/d}|\mathcal{E}(n)|^{1/d})$ and thus the ETSP Assgmt algorithm has time complexity in $O(n^{(d-1)/d}|\mathcal{E}(n)|^{1/d})$.[5]

The Grid Assgmt algorithm we have presented is only valid for environments in $\mathbb{R}^2$. This was done in an effort to simplify the presentation. However, the

[5]Here $|\mathcal{E}(n)|$ denotes the d-dimensional volume of $\mathcal{E}(n)$.



(a) ETSP Assgmt algorithm in a sparse environment.

(b) Grid Assgmt algorithm in a dense environment.

Figure 3.9: Monte Carlo simulations for uniformly randomly generated vehicle and target positions. Each point is the mean completion time of 30 independent trials. Error bars show plus-minus one standard deviation.

extension to $\mathbb{R}^d$ is straightforward. For example, in $\mathbb{R}^3$ the environment is partitioned into small cubes. Vehicles first try to find a free target in their own cube, then in their own column, then in their own plane, and then finally, they transfer into a new plane that has an available target. The worst-case bound is then given by $O(|\mathcal{E}(n)|)$, and for uniformly randomly generated target and vehicle positions, when the environment satisfies the bound

$$|\mathcal{E}(n)| \leq \frac{r_{\text{comm}}^2}{K(d+3)} \frac{n}{\log n},$$

where $K > 1$, is $O(|\mathcal{E}(n)|^{1/d})$, with high probability.

3.7.2 The Case of n Vehicles and m Targets

It should be noted that both the ETSP Assgmt and Grid Assgmt algorithms work, without any modification, when there are n vehicles and m targets. If $m \geq n$, at completion, then n targets are assigned and $m - n$ targets are not. When $m < n$, at completion, all m targets are assigned, and the $n - m$ unassigned vehicles come to a stop after losing a conflict at each of the m targets. By modifying the algorithms so that the $n - m$ unassigned vehicles revisit assigned targets to check for failed vehicles, the robustness of the algorithms can be increased. It is a straightforward exercise to alter the upper bounds when $m \neq n$. For example, the worst-case upper bound on the ETSP Assgmt algorithm becomes $O(\sqrt{|\mathcal{E}(n)|N})$, where $N := \min\{n, m\}$, and holds for any n and m. Similarly, the worst-case upper bound on the Grid Assgmt algorithm remains $O(|\mathcal{E}(n)|)$ and holds for any n and m. In addition, the lower bound on the monotonic class easily extends when $m \geq n$. However, the extension for $m < n$ appears to require a different construction of worst-case vehicle and target positions.

3.7.3 Alternate Scaling Laws

We have given complexity bounds for the case when r_{comm} and $v_{\max}$ are fixed constants, and $\mathcal{E}(n)$ grows with n. We allow the environment $\mathcal{E}(n)$ to grow with n so that, as more vehicles are involved in the task, their workspace is larger. An equivalent setup would be to consider a fixed size environment, and allow r_{comm} and $v_{\max}$ to decrease with increasing n. Scaling the communication radius inversely with the number of vehicles arises in the study of wireless networks (Gupta and Kumar, 2000). As the density of wireless nodes in a fixed area increase, the effects of wireless congestion and media access problems become more prevalent. To reduce these effects, the nodes reduce their transmission radius, thus reducing their interference footprint. The idea of scaling the vehicles' maximum speed inversely with n occurs due to physical congestion (Sharma et al., 2007). As the density of robots increases, it necessarily takes longer for the robots to travel across their environment.

Motivated by this discussion, we introduce a new set of parameters, $\tilde{\mathcal{E}}$, $\tilde{r}_{\text{comm}}(n)$, and $\tilde{v}_{\max}(n)$ satisfying $|\tilde{\mathcal{E}}| \in \mathbb{R}_{>0}$ and $\tilde{v}_{\max}(n) = \Theta(\tilde{r}_{\text{comm}}(n))$. Since $\tilde{v}_{\max}(n)$ and $\tilde{r}_{\text{comm}}(n)$ scale at the same rate, the amount of time required to travel a distance $\tilde{r}_{\text{comm}}(n)$ is independent of n. Then, analogous to the definition of environment size, we define the communication range to be: *sparse* if $\tilde{r}_{\text{comm}}(n)\sqrt{n} \to 0^+$, as $n \to +\infty$; *critical* if $\tilde{r}_{\text{comm}}(n)\sqrt{n} \to \text{const} \in \mathbb{R}_{>0}$ as $n \to +\infty$; *dense* if $\tilde{r}_{\text{comm}}(n)\sqrt{n} \to +\infty$, as $n \to +\infty$.

We now summarize the worst-case results as follows.

Corollary 3.25 (Scaling radius and speed). *Consider any initial positions of n vehicles, with communication range $\tilde{r}_{\text{comm}}(n)$ and maximum speed $\tilde{v}_{\max}(n) = \Theta(\tilde{r}_{\text{comm}}(n))$, and n targets in the fixed environment $\tilde{\mathcal{E}}$. Then:*

(i) the ETSP Assgmt *algorithm solves the target assignment problem in $O(\sqrt{n}/\tilde{r}_{\text{comm}}(n))$ time;*

(ii) if $\tilde{r}_{\text{comm}}(n)$ is sparse or critical, then ETSP Assgmt *is within a constant factor of the optimal monotonic algorithm for worst-case initial conditions;*

(iii) the Grid Assgmt *algorithm solves the target assignment problem in* $O(1/\tilde{r}_{\text{comm}}(n)^2)$ *time; and*

(iv) if $\tilde{r}_{\text{comm}}(n)$ *is dense or critical, then the* Grid Assgmt *algorithm is within a constant factor of the optimal monotonic algorithm for worst-case initial conditions.*

3.7.4 Summary

In this chapter we have studied a version of the target assignment problem in which each vehicle has a list of the target positions, but has only limited communication capabilities. We introduced the class of monotonic algorithms for approaching these problems and gave a lower bound on its asymptotic performance. We introduced two algorithms in this class, the ETSP Assgmt algorithm and the Grid Assgmt algorithm. We have shown that in sparse environments, where communication between vehicles is infrequent, the ETSP Assgmt algorithm is within a constant factor of the optimal monotonic algorithm for worst-case initial conditions. On the other hand, in dense environments, where communication is more prevalent, the Grid Assgmt algorithm is within a constant factor of the optimal monotonic algorithm for worst-case initial conditions. Both algorithms extend to higher dimensional spaces and to problems where the number of vehicles and targets differ, and the Grid Assgmt algorithm can be implemented in a sensor based version, where vehicles have no *a priori* target knowledge.

Chapter Four

Team Forming

In this chapter we consider a dynamic vehicle routing problem, where new task locations arrive in real-time. In particular, we consider a heterogeneous fleet of mobile robotic agents deployed in an environment $\mathcal{E} \subset \mathbb{R}^2$. Each robot is capable of providing one of κ services. Tasks (from now on we will call these *demands for service*, or simply *demands* to be consistent with the dynamic vehicle routing literature) appear in the environment sequentially over time, assume a random location in $\mathcal{E}$, and require some subset of the κ services. To service a demand, all required services must be present at the demand location. Thus, for each demand, a team of vehicles which can provide the required services must be formed, and must travel to the demand location. The goal is to minimize the expected delay between a demand arrival and its service completion. We refer to this problem as the *dynamic team forming problem* (DTFP). This problem arises, for example, in UAV surveillance (Chong et al., 2008) where the services represent waveforms for interrogation of a target/region, such as electro-optical, infra-red, synthetic aperture radar, foliage penetrating radar, and moving target indication radar.

In dynamic vehicle routing problems, such as the dynamic traveling repairperson and the dynamic pick-up and delivery problem (see Chapter 1) the expected delay increases with the demand arrival rate. This trade-off is well known in ad hoc wireless networks (Sharma et al., 2006; Gamal et al., 2006); If nodes increase the rate at which they send messages (i.e., the throughput), then this increases the expected delay a message will incur before arriving at its destination. Thus, our perspective in this chapter is to study the service delay in the DTFP as a function of the demand throughput.

The contributions of this chapter are the following. First, we introduce the novel *dynamic team forming problem.* Second, we propose three policies for dynamic team forming; the Complete Team policy where teams are formed that consist of all possible services; the Demand-Specific Team policy where teams are formed for each type of demand; and the Scheduled Demand-Specific Team policy where each type of demand is serviced by a demand-specific team, but only during certain intervals of time, as defined by a schedule. All three policies utilize Euclidean traveling salesperson tours to compute optimal routes through sets of demands. Third, by making some assumptions on the system parameters (e.g., on relative number of vehicles, demand-type frequency, etc.), we study the expected demand delay as a function of the throughput of the vehicle network (i.e., the rate at which demands are serviced). We derive a lower bound on the expected delay of the DTFP, upper bounds for the delay each of the three policies, and show that for certain classes of system parameters each policy performs within a constant factor of the optimal. In addition, we show that there is a canonical throughput delay profile which is common to the lower bound and all three policies.

The chapter is organized as follows. In Section 4.1 we review some background material. In Section 4.2 we introduce the dynamic team forming problem, and in Section 4.3 we present the three policies. In Section 4.4.1 we present simplifying assumptions for the purpose of analysis and in Section 4.4.2 we introduce the canonical throughput-delay profile. In Section 4.4.3 we present a lower bound on the delay, and in Section 4.4.4 we present upper bounds for each the policies. Finally, in Section 4.4.5 we present a comparison between the three policies.

4.1 Preliminary Material

Here we review results on partitioning an environment into "fairly" square regions, and some concepts of queueing theory.

4.1.1 Partitioning an environment

The following definition formalizes the idea of partitioning a square environment $\mathcal{E} \subset \mathbb{R}^2$ into n regions, such that each region is "approximately" a square of area $|\mathcal{E}|/n$. The result will be useful when bounding the length of an ETSP tour using Theorem 2.2.

Definition 4.1 (c-square partition)**.** *A partition of $\mathcal{E}$ into n regions is* c-square *if each region can be contained in a square of area $c|\mathcal{E}|/n$.*

One can easily create a 4-square partition by 1) gridding $\mathcal{E}$ into $\lceil\sqrt{n}\rceil^2$ squares, 2) selecting $\lceil\sqrt{n}\rceil^2 - n$ pairs of edge adjacent squares, such that no square appears in more than one pair, and 3) fusing each pair into a single region.

4.1.2 Queueing with bulk service

Consider a queueing system (see Chapter 2 for a review on queueing systems) with Poisson arrivals at rate λ, and a single server providing bulk service. As customers arrive they form a queue and are served in batches. Every t_{batch} seconds a batch is served containing either the first M customers in the queue, or the entire queue, whichever is smaller. In (Bailey, 1954) the following result is established.

Theorem 4.2 (Expected waiting time, (Bailey, 1954))**.** *If $M > \lambda t_{\mathrm{batch}}$, then the expected waiting time $\mathcal{W}$ satisfies*

$$\mathcal{W} \leq \frac{M-1}{2\lambda} + \frac{t_{\mathrm{batch}}}{2(M - \lambda t_{\mathrm{batch}})}. \tag{4.1}$$

4.2 The Dynamic Team Forming Problem (DTFP)

Vehicle model: Consider n vehicles (robotic agents) contained in a square environment $\mathcal{E} \subset \mathbb{R}^2$. Each vehicle has first order dynamics with speed bounded by $v_{\max} > 0$. Each vehicle is capable of providing one of κ services. We assume there are n_j vehicles capable of providing service j, for each $j \in \{1, \dots, \kappa\}$, and thus

$$n := \sum_{j=1}^{\kappa} n_j.$$

Demand model: There are $\mathcal{K}$ different types of demands. Demands of type $\alpha \in \{1, \dots, \mathcal{K}\}$ arrive in the environment over time according to a Poisson process with rate λ_α. Upon arrival each demand assumes an independent and identically distributed (i.i.d.) location uniformly in $\mathcal{E}$. Each demand-type $\alpha \in \{1, \dots, \mathcal{K}\}$ requires a subset of the κ services. It will be useful to record the required services in a zero-one vector $U_\alpha \in \{0, 1\}^\kappa$. The jth entry of U_α is 1 if service j is required for demand-type α, and 0 otherwise. The on-site service time for a demand of type α is an i.i.d. random variable with mean $\bar{s}_\alpha$ and finite variance. To complete the service of a demand of type α, a team of vehicles capable of providing the required services must travel to the demand location and remain there for the on-site service time. Note that the total demand arrival rate is

$$\lambda := \sum_{\alpha=1}^{\mathcal{K}} \lambda_\alpha.$$

Performance metric: A policy for a service vehicle is a map which assigns a velocity to each service vehicle as a function of the current state of the system (i.e., the outstanding demands and the current vehicle locations). For a given policy P by which vehicles service demands[1], let $D_\alpha(i)$ denote the difference between the service completion time and the arrival time of the ith demand of type α. Let $D_\alpha(P) := \lim_{i \to +\infty} \mathbb{E}[D_\alpha(i)]$ denote the limiting *expected delay*

[1]We assume that computations are centralized, and leave the problem of decentralizing our policies to future work.

of demand-type α under policy P. Then, the DTFP is to find policies which minimize some cost function of the delays. In particular we consider two cost functions; the worst-case delay, and the average delay:

$$\min_P \max_\alpha D_\alpha(P) \quad \text{or} \quad \min_P \sum_\alpha D_\alpha(P).$$

As in the classical queueing literature (Kleinrock, 1975), we can give a necessary condition for the existence of a stabilizing policy (i.e., a policy for which the limiting expected delays are all finite). First, we define the matrix

$$U := [U_1, \ldots, U_{\mathcal{K}}] \in \{0,1\}^{\kappa\times\mathcal{K}}. \tag{4.2}$$

Then, a necessary condition for stability is that

$$U[\lambda_1\bar{s}_1, \ldots, \lambda_{\mathcal{K}}\bar{s}_{\mathcal{K}}]^T < [n_1, \ldots, n_\kappa]^T \tag{4.3}$$

component-wise. The jth inequality in equation (4.3) states that the fraction of time a vehicle of type j is busy performing on-site service must be less than 1 for any stable policy.

Dynamic traveling repairperson problem (DTRP): The DTRP is a special case of the dynamic team forming problem in which there is only one service (i.e., $\kappa = 1$), and thus only one demand-type. When $\kappa = 1$ equation (4.3) becomes $\lambda\bar{s} < n$, or

$$\varrho := \frac{\lambda\bar{s}}{n} < 1.$$

The quantity ϱ is the load factor introduced in Chapter 2. In (Bertsimas and van Ryzin, 1993*a*), two lower bounds on the optimal expected delay D^* are presented which will be useful in the upcoming analysis. First,

$$D^* \geq \frac{1}{v_{\max}}\mathbb{E}\left[\min_{\mathbf{p}\in\{\mathbf{p}_1,\ldots,\mathbf{p}_n\}} \|\mathbf{q}-\mathbf{p}\|\right] + \bar{s}, \tag{4.4}$$

where $\mathbf{p}_1, \ldots, \mathbf{p}_n$ are the n locations which minimize the expected distance to a uniformly distributed location $\mathbf{q}$. Second, there exists a $\gamma > 0$ such that

$$D^* \geq \gamma^2 \frac{\lambda|\mathcal{E}|}{n^2 v_{\max}^2 (1-\varrho)^2} - \frac{\bar{s}(1-2\varrho)}{2\varrho} =: D_{\mathrm{DTRP}}(n,\lambda). \tag{4.5}$$

Several policies are developed in (Bertsimas and van Ryzin, 1993*a*). A policy which we will utilize in this chapter is the TSP Partitioning policy shown in Algorithm 4.1. We slightly alter the policy and use the c-square partition in Definition 4.1.

Algorithm 4.1: TSP Partitioning policy

```
  Optimize: over demand set-size $M$.
1 Partition $\mathcal{E}$ into $n$ approximately square regions and assign one vehicle to
  each region.
2 foreach region-vehicle pair do
3     As demands arrive in the region, form sets of size $M$.
4     As sets are formed, deposit them in a queue.
5     Service the queue first-come, first-served, following an optimal TSP
      tour on each set of $M$ demands.
```

From Bertsimas and van Ryzin (1993*a*), the optimal value of M is

$$\frac{C\lambda^2|\mathcal{E}|}{n^2 v_{\max}^2 (1-\varrho)^2},$$

for some $C > 0$. While we will utilize the TSP Partitioning policy in what follows, it should be noted that we could equivalently utilize the receding-horizon policy in Papastavrou (1996).

4.3 Dynamic Team Forming Policies

In what follows we will consider *demand-type unbiased* policies; policies P for which the delay of each demand is equal and thus

$$D_1(P) = D_2(P) = \cdots = D_{\mathcal{K}}(P).$$

For demand-type unbiased policies, the worst-case delay and average delay problems are equivalent. Policies of this type are amenable to analysis because the demand-type unbiased constraint collapses the feasible set of delays from a subset of $\mathbb{R}^{\mathcal{K}}$ to a subset of $\mathbb{R}$. Because of this, we can simply talk about the delay of a policy P as $D(P)$ (or D_P), and the least achievable delay as D^*.

We will now introduce three policies for the dynamic team forming problem. Their performance will be analyzed in Section 4.4.

4.3.1 Policy 1: Complete Team

We begin by proposing a policy which essentially turns the problem into a dynamic traveling repairperson problem.

Policy 1: COMPLETE TEAM

```
1 Form $N_{\text{CT}} := \min\{n_1, \dots, n_\kappa\}$ teams of $\kappa$ vehicles, where each team
  contains one vehicle of each type.
2 Have each team meet and move as a single entity.
3 As demands arrive, service them by one of the $N_{\text{CT}}$ teams according to the
  TSP PARTITIONING policy.
```

4.3.2 Policy 2: Demand-Specific Team

Recalling that $U := [U_1, \dots, U_{\mathcal{K}}] \in \mathbb{R}^{\kappa \times \mathcal{K}}$, the vector $U\mathbf{1}_{\mathcal{K}}$ records in its jth entry the number of demand-types that require service j, where $\mathbf{1}_{\mathcal{K}}$ is a $\mathcal{K} \times 1$ vector of ones. Thus, if

$$U\mathbf{1}_{\mathcal{K}} \leq [n_1, \dots, n_\kappa]^T \tag{4.6}$$

component-wise, then there are enough vehicles of each type to create a dedicated team for every demand-type. More specifically, we could create

$$N_{\text{TST}} := \left\lfloor \min\left\{ \frac{n_j}{e_j^T U \mathbf{1}_{\mathcal{K}}} \mid j \in \{1, \dots, \kappa\} \right\} \right\rfloor$$

teams for each demand-type, where e_j is the jth vector of the standard basis of $\mathbb{R}^\kappa$. Thus, when equation (4.6) is satisfied, we have the following policy.

Policy 2: DEMAND-SPECIFIC TEAM

```
  Assumes: Equation (4.6) is satisfied.
1 For each of the $\mathcal{K}$ demand-types, create $N_{\text{TST}}$ teams of vehicles, where there
  is one vehicle in the team for each service required by the demand-type.
2 Service each demand by one of its $N_{\text{TST}}$ corresponding teams, according to
  the TSP PARTITIONING policy.
```

4.3.3 Policy 3: Scheduled Demand-Specific Team

The DEMAND-SPECIFIC TEAM policy can be applied only when equation (4.6) is satisfied; that is, when there is a sufficient number of vehicles of each type. Here we propose a policy which requires only a single vehicle of each type. The policy partitions the demand-types into groups, where each group is chosen such that there is a sufficient number of vehicles to create a dedicated team for each demand-type in the group. The DEMAND-SPECIFIC TEAM policy is then run on each group sequentially. We begin by introducing a *service schedule* which defines the partition of demand-types into groups.

Definition 4.3 (Service schedule). *A service schedule is a partition of the $\mathcal{K}$ demand-types into L time slots, such that each demand-type appears in precisely one time slot, and the demand-types in each time slot are pairwise disjoint (i.e., In a given time slot, each service appears in at most one demand-type).*

With the definition of a service schedule we can present the third policy.

Policy 3: Scheduled Demand-Specific Team

```
Assumes: A service schedule with time slot duration $t_B$.
Optimize: over time slot duration $t_B$ and demand set-size $M$.
1 Partition $\mathcal{E}$ into $N_{CT} := \min\{n_1, \dots, n_\kappa\}$ approximately square regions and
  assign one vehicle of each type to each region.
2 foreach region do
3     Form a queue for each of the $\mathcal{K}$ demand-types.
4     foreach time slot in the schedule do
5         For each demand-type in the time slot, create a team containing one
          vehicle for each required service.
6         For each team, service the first $M$ demands in the corresponding
          queue, or as many as can be served in time $t_B$, by following an
          optimal TSP tour.
7 When the end of the service schedule is reached, repeat.
```

4.4 Analysis of the Dynamic Team Forming Problem

In this section we present simplifying assumptions, scaling laws, and the canonical throughput-delay profile which will allow us to study the asymptotic expected delay of the three team forming policies. We derive a lower bound on the achievable delay (independent of policy), and upper bounds for the delay of each policy.

4.4.1 Simplifying assumptions and asymptotic regime

To analyze the performance of the three policies we assume the following:

(A1) There are n/κ vehicles of each type (i.e., $n_j = n/\kappa$ for each $j \in \{1, \dots, \kappa\}$).

(A2) The arrival rate is $\lambda/\mathcal{K}$ for each demand-type, (i.e., $\lambda_\alpha = \lambda/\mathcal{K}$ for each demand $\alpha \in \{1, \dots, \mathcal{K}\}$).

(A3) The on-site service time has mean $\bar{s}$ and is upper bounded by $s_{\max}$ for all demand-types (i.e., $\bar{s}_\alpha = \bar{s}$ for each demand-type $\alpha \in \{1, \dots, \mathcal{K}\}$).

(A4) There exists $p \in [1/\kappa, 1]$ such that for each $j \in \{1, \dots, \kappa\}$, the service j appears in $p\mathcal{K}$ of the $\mathcal{K}$ demand-types. Thus, each demand will require service j with probability p.

With these assumptions, the stability condition in equation (4.3) simplifies to

$$\frac{\lambda}{n} < \frac{1}{p\kappa\bar{s}}. \tag{4.7}$$

For a stable policy P we say that λ is the *total throughput* of the system (i.e., the total number of demands served per unit time), and

$$\mathcal{T}(n) := \frac{\lambda}{n}$$

is the *per-vehicle throughput* (in what follows we write the per-vehicle throughput as $\mathcal{T}_n$ to simplify notation).

We are interested in studying the expected delay of each demand-type as a function of the per-vehicle throughput $\mathcal{T}_n$. In particular, in the next sections we study the performance as the number of vehicles n becomes large. As n increases, if the density of vehicles is to remain constant, then the environment must grow. In fact, the ratio $\sqrt{|\mathcal{E}|}/v_{\max}$ must scale as $\sqrt{n}$, (Sharma et al., 2007). In Chapter 3 we referred to this scaling as a *critical environment*. Thus we will study the performance in the following regime.

Definition 4.4 (Asymptotic regime). *In the asymptotic regime (i) the number of vehicles $n \to +\infty$; (ii) on-site service times are independent of n; (iii) $|\mathcal{E}(n)|/(nv_{\max}^2(n)) \to \text{const} > 0$.*

4.4.2 Canonical throughput-delay profile

In what follows we will characterize the way in which the delay varies with the per-vehicle throughput $\mathcal{T}_n$. We will see that there is a canonical throughput-delay profile $f_{D_{\min},D_{\text{ord}},\mathcal{T}_{\text{crit}}} : \mathbb{R}_{>0} \to \mathbb{R}_{>0} \cup \{+\infty\}$ which has the form

$$\mathcal{T} \mapsto \begin{cases} \max\left\{D_{\min}, \dfrac{D_{\text{ord}}(\mathcal{T}/\mathcal{T}_{\text{crit}})}{(1-\mathcal{T}/\mathcal{T}_{\text{crit}})^2}\right\}, & \text{if } \mathcal{T} < \mathcal{T}_{\text{crit}}, \\ +\infty, & \text{if } \mathcal{T} \geq \mathcal{T}_{\text{crit}}. \end{cases} \tag{4.8}$$

This profile is described by the three positive parameters $D_{\min}$, D_{ord} and $\mathcal{T}_{\text{crit}}$, where $D_{\text{ord}} \geq D_{\min}$. These parameters have the following interpretation:

- $D_{\min}$ is the minimum achievable delay for any positive throughput.

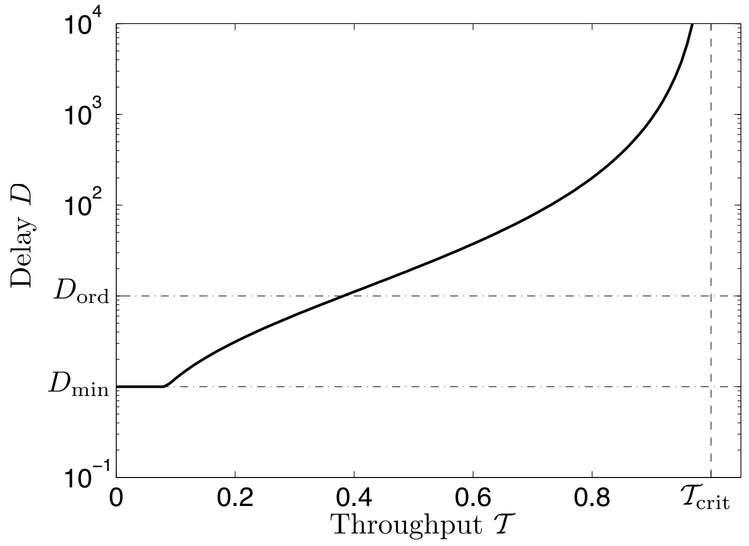


Figure 4.1: The canonical throughput-delay profile for the dynamic team forming problem. In this figure $D_{\text{min}} = 1$, $D_{\text{ord}} = 10$, and $\mathcal{T}_{\text{crit}} = 1$. If $\mathcal{T} \geq \mathcal{T}_{\text{crit}}$, then the delay is $+\infty$.

- $\mathcal{T}_{\text{crit}}$ is the maximum achievable throughput (or capacity).
- D_{ord} is the delay when operating at $(3 - \sqrt{5})/2 \approx 0.38$ of capacity $\mathcal{T}_{\text{crit}}$. Additionally, D_{ord} captures the order of the delay when operating at a constant fraction of capacity.

An example of the throughput-delay profile with parameters $D_{\text{min}} = 1$, $D_{\text{ord}} = 10$, and $\mathcal{T}_{\text{crit}} = 1$ is shown in Figure 4.1. In what follows we will use these three parameters to compare the performance of our policies.

4.4.3 Lower bound on the achievable delay

We now lower bound the achievable delay D^* for demand-type unbiased policies. Note that all parameters are potentially a function of n. However, to simplify the notation we omit the explicit dependence. For convenience, Table 4.1 contains all parameters and their definitions.

Theorem 4.5 (Optimal delay)**.** *In the asymptotic regime, the optimal delay of the DTFP as a function of the per-vehicle throughput $\mathcal{T}_n$ is in $\Omega\Big(f_{D_{\text{min}},D_{\text{ord}},\mathcal{T}_{\text{crit}}}(\mathcal{T}_n)\Big)$, where*

$$D_{\text{min}} = \sqrt{\kappa}, \quad D_{\text{ord}} = \kappa, \quad \mathcal{T}_{\text{crit}} = \frac{1}{p\kappa\bar{s}}.$$

Table 4.1: Parameters used in the dynamic team forming problem.

Parameter	Definition
κ	number of different services
$\mathcal{K}$	number of different demand-types
p	fraction of demands requiring an individual service
$\bar{s}$, $s_{\max}$	expected and maximum on-site service time
L	number of time slots in service schedule
$\kappa_{\max}$	maximum number of services required for a demand

Proof. By assumption (A4), service $j \in \{1, \dots, \kappa\}$ is required in $p\mathcal{K}$ of the $\mathcal{K}$ demand-types. By assumption (A2), the arrival rate of demands requiring service j is $p\lambda$. By assumption (A1), $n_j = n/\kappa$ vehicles can provide service j. Thus, we can use the results on the DTRP to lower bound the achievable delay of n/κ vehicles servicing demands arriving at rate $p\lambda$. That is, for every policy P we have

$$\sum_{\text{tasks } \alpha \text{ requiring service } j} \frac{\lambda_\alpha}{p\lambda} D_\alpha(P) \geq D_{\text{DTRP}}(n/\kappa, p\lambda). \tag{4.9}$$

By assumption (A2), $\lambda_\alpha = \lambda/\mathcal{K}$ for each $\alpha \in \{1, \dots, \mathcal{K}\}$, and by restricting our attention to demand-type unbiased policies, $D_\alpha(P) = D(P)$ for each $\alpha \in \{1, \dots, \mathcal{K}\}$. Applying the bound in equation (4.5), we can write equation (4.9) as

$$D^*(n) \geq D_{\text{DTRP}}(n/\kappa, p\lambda) \in \Omega\left(\frac{p\lambda|\mathcal{E}|}{(n/\kappa)^2 v_{\max}^2 (1 - p\kappa\lambda/n)^2}\right),$$

In the asymptotic regime the above equation becomes

$$D^*(n) \in \Omega\left(\frac{p\kappa^2 \mathcal{T}_n}{(1 - p\kappa\bar{s}\mathcal{T}_n)^2}\right)$$

In addition, in the asymptotic regime, equation (4.4) yields $D^*(n) \in \Omega(\sqrt{\kappa})$. Combining the two results we obtain a lower bound of $\Omega(f_{D_{\min},D_{\text{ord}},\mathcal{T}_{\text{crit}}}(\mathcal{T}_n))$, where $f_{D_{\min},D_{\text{ord}},\mathcal{T}_{\text{crit}}}$ is the canonical throughput-delay profile defined in equation (4.8), and the parameters are $D_{\min} = \sqrt{\kappa}$, $D_{\text{ord}} = \kappa$ and $\mathcal{T}_{\text{crit}} = 1/(p\kappa\bar{s})$. □

4.4.4 Upper bounds on the policy throughput-delay

In this section we characterize the performance of each policy in terms of the canonical throughput-delay profile of equation (4.8).

Policy 1: Complete Team

The Complete Team policy is simply the TSP Partitioning policy with n/κ vehicles and with arrival rate λ. In the limiting regime as $\varrho \to 1^-$, the performance of this policy (Bertsimas and van Ryzin, 1993*a*) is within a known constant factor of the lower bound in equation (4.5). The proof in (Bertsimas and van Ryzin, 1993*a*) utilizes the following facts: as $\varrho \to 1^-$, the number of unserviced demands M tends to $+\infty$, and for a set $\mathcal{Q}$ of M i.i.d. uniform points in a square environment of area $|\mathcal{E}|$

$$\lim_{M\to+\infty} \frac{\mathrm{ETSP}(\mathcal{Q})}{\sqrt{M}} = \beta_{\mathrm{TSP}}\sqrt{|\mathcal{E}|}, \tag{4.10}$$

for some constant $\beta_{\mathrm{TSP}} > 0$. Following the same proof as in (Bertsimas and van Ryzin, 1993*a*) but replacing equation (4.10) with the result in Theorem 2.2 (which is not as tight, but holds for all values of M), and using the fact that the partition is approximately square, one can prove that the delay of the TSP Partitioning policy with n vehicles and with arrival rate λ is in

$$O\left(\max\left\{\frac{1}{v_{\max}}\sqrt{\frac{|\mathcal{E}|}{n}}, \frac{\lambda|\mathcal{E}|}{n^2 v_{\max}^2(1-\lambda\bar{s}/n)^2}\right\}\right), \tag{4.11}$$

in the asymptotic regime. Combining equation (4.11) with the throughput-delay profile in equation (4.8) we obtain the following result.

Theorem 4.6 (Complete Team delay)**.** *In the asymptotic regime, the expected delay of the* Complete Team *policy as a function of the per-vehicle throughput* $\mathcal{T}_n$ *is in* $O\Big(f_{D_{\min},D_{\mathrm{ord}},\mathcal{T}_{\mathrm{crit}}}(\mathcal{T}_n)\Big)$*, where*

$$D_{\min} = \sqrt{\kappa}, \quad D_{\mathrm{ord}} = \kappa, \quad \mathcal{T}_{\mathrm{crit}} = \frac{1}{\kappa\bar{s}}.$$

Notice that if $p \sim 1$ (i.e., each service is required in a constant fraction of the demands), then the policy is within a constant factor of the optimal. However, in certain instances policy 1 may be inefficient as each vehicle visits every demand, not just the ones which require its service. This inefficiency appears as a limit on the per-vehicle throughput of $1/\kappa$, independent of p.

Remark 4.7 (Dynamic traveling repairperson delay)**.** *In the DTRP we have* $\kappa = p = \mathcal{K} = 1$*, and thus combining Theorems 4.5 and 4.6 we see that the expected delay in the asymptotic regime is in* $\Theta\Big(f_{D_{\min},D_{\mathrm{ord}},\mathcal{T}_{\mathrm{crit}}}(\mathcal{T}_n)\Big)$*, where* $D_{\min} = D_{\mathrm{ord}} = 1$*, and* $\mathcal{T}_{\mathrm{crit}} = 1/\bar{s}$*.* ●

Policy 2: Demand-Specific Team

With assumptions (A1)-(A4), the necessary condition on the number of vehicles required for this policy, given in equation (4.6), becomes $p\mathcal{K} \leq n/\kappa$, and thus $N_{\text{TST}} := \lfloor n/(\kappa p\mathcal{K}) \rfloor$. In the following theorem we characterize the delay of the demand-specific team policy.

Theorem 4.8 (Demand-Specific Team delay)**.** *In the asymptotic regime, if $p\mathcal{K} \leq n/\kappa$, then the expected delay of the* Demand-Specific Team *policy as a function of the per-vehicle throughput $\mathcal{T}_n$ is in $O\Big(f_{D_{\min},D_{\text{ord}},\mathcal{T}_{\text{crit}}}(\mathcal{T}_n)\Big)$ where*

$$D_{\min} = \sqrt{p\kappa\mathcal{K}}, \quad D_{\text{ord}} = p\kappa\mathcal{K}, \quad \mathcal{T}_{\text{crit}} = \frac{1}{C\bar{s}p\kappa},$$

and $C \in [1, 2[$ is defined as $C = n/(\kappa p\mathcal{K}N_{\text{TST}})$.

Proof. The arrival rate for each demand-type is $\bar{\lambda} = \lambda/\mathcal{K}$ (by assumption (A2)), and the number of teams that provide service to each demand type is N_{TST}. Since $N_{\text{TST}} \geq 1$, and $N_{\text{TST}} \leq n/(\kappa p\mathcal{K})$, we can define

$$C := \frac{n}{\kappa p\mathcal{K}N_{\text{TST}}} \in [1, 2[.$$

From the TSP Partitioning policy result in equation (4.11), the delay is

$$\begin{aligned} D_{\text{TST}}(n) &\in O\left(\max\left\{\frac{1}{v_{\max}}\sqrt{\frac{|\mathcal{E}|}{N_{\text{TST}}}}, \frac{\bar{\lambda}|\mathcal{E}|}{N_{\text{TST}}^2 v_{\max}^2 (1-\bar{\lambda}\bar{s}/N_{\text{TST}})^2}\right\}\right) \\ &= O\left(\sqrt{p\kappa\mathcal{K}}, \frac{p^2\kappa^2\mathcal{K}\mathcal{T}_n}{(1-C\bar{s}p\kappa\mathcal{T}_n)^2}\right). \end{aligned}$$

Letting $D_{\min} = \sqrt{p\kappa\mathcal{K}}$, $D_{\text{ord}} = p\kappa\mathcal{K}$ and $\mathcal{T}_{\text{crit}} = 1/(2\bar{s}p\kappa)$ we obtain the desired result. □

From this analysis we see that the demand-specific team policy can achieve near optimal throughput. However, it requires that there are a sufficient number of vehicles. The following policy requires only a single vehicle of each type.

Policy 3: Scheduled Demand-Specific Team

The following theorem bounds the delay of policy 3.

Theorem 4.9 (Scheduled Demand-Specific Team delay)**.** *In the asymptotic regime, the expected delay of the* Scheduled Demand-Specific Team *policy* *as a function of the per-vehicle throughput* $\mathcal{T}_n$ *is in* $O\Big(f_{D_{\min},D_{\mathrm{ord}},\mathcal{T}_{\mathrm{crit}}}(\mathcal{T}_n)\Big)$ *where*

$$D_{\min} = L\sqrt{\kappa}, \quad D_{\mathrm{ord}} = L\kappa, \quad \mathcal{T}_{\mathrm{crit}} = \frac{\mathcal{K}}{\nu s_{\max} L\kappa},$$

for any fixed $\nu > 1$.

Proof. Consider a service schedule with length L and time slot duration t_{B}. In each of the n/κ regions (assumption (A1)), each demand-type has arrival rate $\bar{\lambda} := \lambda\kappa/(\mathcal{K}n)$ (assumption (A2)), and the queue for that demand-type is serviced for t_{B} seconds every Lt_{B} seconds. (Notice that for stability we require that $t_{\mathrm{B}} \geq \bar{s}\bar{\lambda}Lt_{\mathrm{B}}$, which implies that the per-vehicle throughput must satisfy $\mathcal{T}_n < \mathcal{K}/(L\bar{s}\kappa)$.) Since each region can be contained in a square of area $c|\mathcal{E}|\kappa/n$, where $c \leq 4$, we can use Theorem 2.2 to upper bound the amount of time required to service M demands by $(2\beta/v_{\max})\sqrt{M|\mathcal{E}|\kappa/n} + s_{\max}M$. Using the fact that there exists $C \in \mathbb{R}_{>0}$ such that $\sqrt{|\mathcal{E}|}/v_{\max} \geq C\sqrt{n}$, and redefining $\beta := 2C\beta$, the upper bound becomes

$$\beta\sqrt{M\kappa} + s_{\max}M. \tag{4.12}$$

Now, fix $\epsilon > 0$, and let us set $M := \eta\bar{\lambda}(Lt_{\mathrm{B}})$, where $\eta \geq 1+\epsilon$ is the smallest number such that $M \in \mathbb{N}$. With this value of M we are guaranteed to service more demands in time slot t_{B} than are expected to arrive in time Lt_{B}. Let us now consider two cases: $M = 1$, and $M > 1$. If $M = 1$, then in order to service one demand in time t_{B}, we require from equation (4.12) that

$$t_{\mathrm{B}} \geq \beta\sqrt{\kappa} + s_{\max}. \tag{4.13}$$

In the other case, when $M = \eta\bar{\lambda}(Lt_{\mathrm{B}}) \in \mathbb{N}\setminus\{1\}$, in order to service M demands in time t_{B}, we require from equation (4.12) that

$$t_{\mathrm{B}} \geq \sqrt{\eta}\beta\sqrt{\bar{\lambda}Lt_{\mathrm{B}}\kappa} + \eta s_{\max}\bar{\lambda}Lt_{\mathrm{B}}.$$

If $\eta s_{\max}\bar{\lambda}L < 1$, or equivalently $M \leq t_{\mathrm{B}}/s_{\max}$, then the previous condition can be rewritten as

$$t_{\mathrm{B}} \geq \frac{\eta\beta^2\bar{\lambda}L\kappa}{(1-\eta s_{\max}\bar{\lambda}L)^2}. \tag{4.14}$$

The condition for t_{B} to be finite (i.e., $\eta s_{\max}\bar{\lambda}L < 1$) depends on $\eta \geq (1+\epsilon)$, which is not desirable since the exact value of η is implicitly defined. However, notice that as $\eta s_{\max}\bar{\lambda}L \to 1^-$, we have $t_{\mathrm{B}} \to +\infty$, and thus $M = \eta\bar{\lambda}(Lt_{\mathrm{B}}) \to +\infty$. This implies that as $\eta s_{\max}\bar{\lambda}L \to 1^-$, we have $(1+\epsilon)/\eta \to 1^-$. Thus, we can replace denominator of equation (4.14) by a constant times $(1-(1+\epsilon)s_{\max}\bar{\lambda}L)^2$. Making this replacement and substituting $\bar{\lambda} = \kappa\mathcal{T}_n/\mathcal{K}$, we obtain

$$t_{\mathrm{B}} \geq \frac{\beta L \kappa^2 \mathcal{T}_n}{\mathcal{K}(1-(1+\epsilon)s_{\max} L\kappa\mathcal{T}_n/\mathcal{K})^2}, \tag{4.15}$$

where the constant β has been redefined.

Let us now examine the queue for a particular demand-type and compute the expected delay. In this queue, demands arrive at a rate $\bar{\lambda}$, and every Lt_{B} seconds, M are served (i.e., $t_{\mathrm{batch}} = Lt_{\mathrm{B}}$). Thus, from equation (4.1), the expected time $\mathcal{W}$ that a demand spends waiting in the queue is

$$\mathcal{W} \leq \frac{M-1}{2\bar{\lambda}} + \frac{Lt_{\mathrm{B}}}{2(M-\bar{\lambda}Lt_{\mathrm{B}})}.$$

If $M = 1$, then we easily obtain that $W \in O(Lt_{\mathrm{B}})$.

On the other hand, if $M > 1$, then

$$\begin{aligned}\mathcal{W} &\leq \frac{\eta\bar{\lambda}Lt_{\mathrm{B}}-1}{2\bar{\lambda}} + \frac{Lt_{\mathrm{B}}}{2(\eta\bar{\lambda}Lt_{\mathrm{B}}-\bar{\lambda}Lt_{\mathrm{B}})}\\ &\leq \frac{\eta Lt_{\mathrm{B}}}{2} + \frac{1}{2(\eta-1)\bar{\lambda}}.\end{aligned}$$

Noticing that $M > 1$ implies that $1/\bar{\lambda} \leq \eta Lt_{\mathrm{B}}$, we again obtain that $\mathcal{W} \in O(Lt_{\mathrm{B}})$. The expected delay for a demand to be serviced is $D_{\mathrm{STST}}(n) \leq \mathcal{W} + t_{\mathrm{B}} \in O(Lt_{\mathrm{B}})$. Choosing t_{B} to be the smallest value that satisfies both equations (4.13) and (4.15), we can upper bound $D_{\mathrm{STST}}(n)$ by the canonical throughput-delay profile, where $D_{\min} = L\sqrt{k}$, $D_{\mathrm{ord}} = L\mathcal{K}$, and $\mathcal{T}_{\mathrm{crit}} = \mathcal{K}/\big((1+\epsilon)s_{\max}L\kappa\big)$, for any positive constant ϵ. □

Next, we will describe a method for creating a service schedule, and bound the schedule length L. The following lemma, lower bounding L, follows from assumption (A4).

Lemma 4.10 (Schedule length I)**.** *Every service schedule contains at least $p\mathcal{K}$ time slots. (i.e., $L \geq p\mathcal{K}$).*

From Lemma 4.10, every service schedule must contain at least $p\mathcal{K}$ slots. We now give a method for creating a schedule. Consider the graph consisting of $\mathcal{K}$ vertices, one for each demand-type, and edges connecting any two vertices that contain a common service. This is known as an intersection graph (McKee and McMorris, 1999). A service schedule is then simply a vertex coloring of this graph. From Section 4.1, the problem of determining the optimal (minimal) coloring is *NP*-hard. However, we can color the graph using the greedy heuristic

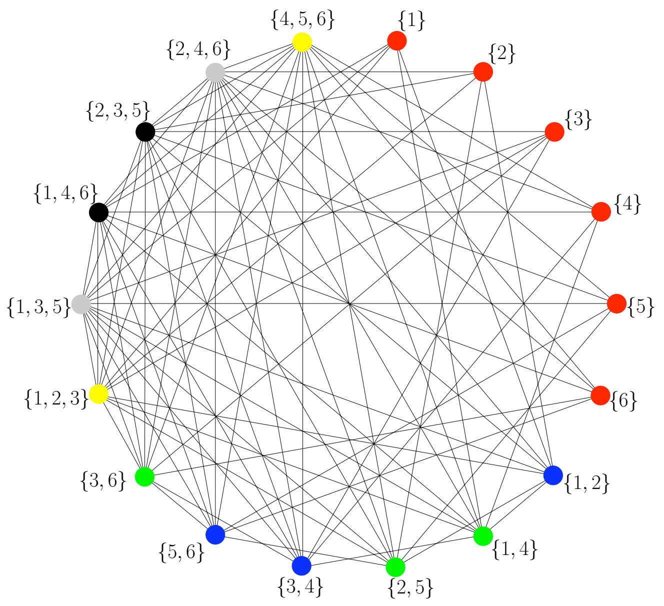


Figure 4.2: Creating a service schedule using the greedy vertex coloring heuristic. In this figure, $\kappa = 6$, $\mathcal{K} = 18$, $p = 1/3$, and the resulting schedule has length $L = 6$.

in Section 4.1. An example is shown in Figure 4.2. Using Theorem 2.5 we arrive at the following result.

Lemma 4.11 (Schedule length II)**.** *If each demand requires no more than $\kappa_{\max} \leq \kappa$ services, then a service schedule with $L \leq \mathcal{K} \min\{\kappa_{\max} p, 1\}$ can be found in $O(\mathcal{K})$ computation time.*

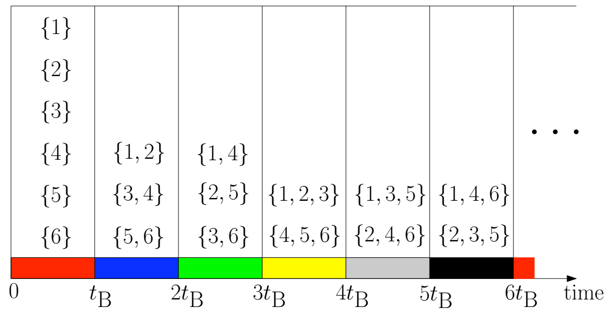


Figure 4.3: Service schedule created by the coloring in Figure 4.2. The demand-types serviced during each time slot are shown (e.g., in time slot $[t_B, 2t_B[$, vehicles of type 1 and 2 meet to service demands with demand-type $\{1, 2\}$).

Table 4.2: A comparison the canonical throughput-delay parameters for the three policies. Two entries for the scheduled demand-specific policy are shown depending on the value of $p \in [1/k, 1]$. Only the order of the capacity is shown, with the constant omitted.

	D_{min}	D_{ord}	$\mathcal{T}_{\text{crit}}$
Lower bound	$\sqrt{\kappa}$	κ	$1/(p\kappa)$
Policy 1: Complete Team	$\sqrt{\kappa}$	κ	$1/k$
Policy 2: Demand-Specific Team	$\sqrt{p\kappa\mathcal{K}}$	$p\kappa\mathcal{K}$	$1/(p\kappa)$
Policy 3: Scheduled Demand-Specific ($p \sim \frac{1}{k}$)	$p\sqrt{\kappa}\mathcal{K}$	$p\kappa\mathcal{K}$	$1/(p\kappa)$
Policy 3: Scheduled Demand-Specific ($p \sim 1$)	$\sqrt{\kappa}\mathcal{K}$	$\kappa\mathcal{K}$	$1/(\kappa)$

4.4.5 Policy comparison

We have shown that the lower bound and the three policies all have delay profiles of the form

$$D(n) \sim \max\left\{D_{\text{min}}, \frac{D_{\text{ord}}(\mathcal{T}_n/\mathcal{T}_{\text{crit}})}{(1-\mathcal{T}_n/\mathcal{T}_{\text{crit}})^2}\right\}.$$

The parameters D_{min}, D_{ord}, and $\mathcal{T}_{\text{crit}}$ are summarized for the lower bound and each of the three policies in Table 4.2. From these results, we can make several conclusions. First, if the throughput is very low, then Policy 1 has an expected delay of $\Theta(\sqrt{k})$, which is within a constant factor of the optimal. In addition, if $p \sim 1$ and each demand requires nearly every service, then Policy 1 is within a constant factor of the optimal in terms of capacity and delay. Second, if $p \sim 1/k$ and each demand requires few services, then the capacity of Policy 1 is sub-optimal, and the capacity of both Policies 2 and 3 are within a constant factor of optimal. However, the delay of Policies 2 and 3 may be much higher than the lower bound when the number of demand-types $\mathcal{K}$ is very large. Third, Policy 2 performs at least as well as Policy 3, both in terms of capacity and delay. Thus, one should use Policy 2 if there are a sufficient number of vehicles of each type. However, if $p \sim 1/k$ and if resources are limited such that Policy 2 cannot be used, then Policy 3 should be used to maximize capacity.

From this discussion we see that the policies are complementary, and have large parameter regimes for which their performance, either in terms of capacity or delay, is within a constant factor of the optimal.

4.5 Summary

In this chapter we introduced the novel dynamic team forming problem for vehicle (robotic) networks. We proposed three policies for team forming and characterized their performance in certain asymptotic regimes. For each policy we showed that there is a large regime in the problem parameters for which the performance is within a constant factor of the optimal.

Chapter Five

Priority Classes of Stochastic Demands

In the previous chapter we looked introduced a dynamic vehicle routing problem called the dynamic team forming problem. However, we restricted ourselves to unbiased policies where the expected delay is the same for all demands. In this chapter we explore the concept of biasing or prioritizing among tasks.

A classic problem in queueing theory is that of priority queueing (Kleinrock, 1976). In the simplest setup, customers arrive at a single server sequentially over time. Each customer is a member of either the high-priority or the low-priority class. High priority customers and low priority customers form separate queues. The goal is to provide the highest possible quality of service to the high priority queue (Q_1) while maintaining stability of the low priority queue (Q_2). That is, the goal is to minimize the expected delay for high-priority customers while keeping the length of the low-priority queue finite. If the customer inter-arrival times and the customer service times are distributed exponentially, then the preemptive priority policy is known to be optimal (Kleinrock, 1976):

> When Q_1 is nonempty, serve high priority customers; when Q_1 is empty, serve low-priority customers. If a high priority customer arrives while serving Q_2, then preempt service and immediately begin serving the high-priority customer.

A more general two-class queueing problem is to minimize a convex combination of the service delays for high- and low-priority customers

$$cD_1 + (1-c)D_2, \quad \text{where } c \in (0,1).$$

In this case an optimal policy can be created by using a mixed policy that spends fraction c of the time serving Q_1 as the high-priority queue, and fraction $(1-c)$ serving Q_2 as though it is the high-priority queue (Coffman Jr. and Mitrani, 1980). The set of achievable delays has also been studied in the more general setting of queueing networks (Bertsimas et al., 1994).

In this chapter we introduce an m-class, n-service-vehicle spatial queueing problem, called *dynamic vehicle routing with priority classes.* Demands for service arrive sequentially over time in a compact environment $\mathcal{E}$ in the plane. Each demand is a member of one of m priority classes. Upon arrival, each demand assumes a location in $\mathcal{E}$, and requires a class-dependent amount of on-site service time. To service a demand, one of the n vehicles must travel to the demand location and perform the on-site service. If we specify a policy by which the vehicles serve demands, then the expected delay for demands of class α, denoted D_α, is the expected amount of time between a demands arrival and its service completion. Then, given convex combination coefficients $c_1, \dots, c_m > 0$, the goal is to find the vehicle routing policy that minimizes $c_1 D_1 + \cdots + c_m D_m$. By increasing the coefficients for certain classes, a higher priority level can be given to their demands. This problem has important applications in areas such as UAV surveillance, where targets are given different priority levels based on their urgency or potential importance (Beard et al., 2002).

The main contribution of this chapter is to introduce dynamic vehicle routing with priority classes. We derive a lower bound on the achievable values of the convex combination of delays, and propose a novel policy in which each class of demands is served separately from the others. We show that in heavy load, the policy performs within a constant factor $2m^2$ of the lower bound. Thus, the constant factor is independent of the number of vehicles, the arrival rates of demands, the on-site service times, and the convex combination coefficients. To establish the constant factor, we proceed in a similar manner as (Papastavrou, 1996; Pavone, Bisnik, Frazzoli and Isler, 2008) and develop a system of nonlinear inequality-based recursive equations on the expected number of outstanding demands. We then utilize a novel proof technique to upper bound the limiting number of outstanding demands, which relies on constructing a set of linear equality-based recursive equations to bound trajectories. We present an improvement on the policy in which classes of similar priority are merged together. We also perform extensive simulations and introduce an effective heuristic improvement called the tube heuristic.

This chapter is organized as follows. In Section 5.1 we formalize the problem and in Section 5.2 we derive a lower bound on the achievable delay. In Section 5.3 we introduce and analyze the Separate Queues policy, and present the improvements given by *queue merging* and the *tube heuristic*. Finally, in Section 5.4 we present simulation results.

The work in this chapter was performed in collaboration with Marco Pavone of the Massachusetts Institute of Technology.

5.1 Problem Statement

Consider a compact environment $\mathcal{E}$ in the plane with area $|\mathcal{E}|$. The environment contains n vehicles, each with maximum speed v. Demands of type $\alpha \in \{1, \dots, m\}$

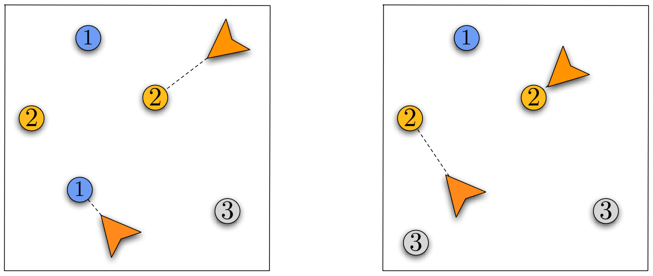


Figure 5.1: A depiction of the problem for two vehicles and three priority classes. Left figure: One vehicle is moving to a class 1 demand, and the other to a class 2 demand. Right figure: The bottom vehicle has serviced the class 1 demand and is moving to a class 2 demand. A new class 3 demand has arrived.

(also called α-demands) arrive in the environment according to a Poisson process with rate λ_α. Upon arrival, demands assume an independently and uniformly distributed location in $\mathcal{E}$. An α-demand is serviced when the vehicle spends an on-site service time at the demand location, which is generally distributed with finite mean $\bar{s}_\alpha$.

Consider the arrival of the ith α-demand. The service delay for the ith demand, $D_\alpha(i)$, is the time elapsed between its arrival and its service completion. The wait-time is $\mathcal{W}_\alpha(i) = D_\alpha(i) - s_\alpha(i)$, where $s_\alpha(i)$ is the on-site service time required by demand i. Recall that a policy for routing the vehicles is said to be *stable* if the expected number of demands in the system for each class is bounded uniformly at all times. A necessary condition for the existence of a stable policy is

$$\varrho := \frac{1}{n}\sum_{\alpha=1}^{m} \lambda_\alpha \bar{s}_\alpha < 1. \tag{5.1}$$

We define this as the *load factor* for this spatial queueing system. As noted in the review on queueing theory in Chapter 2, in general, it is difficult to study a queueing system for all values of $\varrho \in [0,1)$, and a common technique is to focus on the limiting regimes of $\varrho \to 1^-$, referred to as the *heavy load* regime, and $\varrho \to 0^+$, referred to as the *light load* regime.

Given a stable policy P the steady-state service delay is defined as $D_\alpha(P) := \lim_{i\to+\infty} \mathbb{E}\left[D_\alpha(i)\right]$, and the steady-state wait time is $\mathcal{W}_\alpha(P) = D_\alpha(P) - \bar{s}_\alpha$. Thus, for a stable policy P, the *average delay per demand* is

$$D(P) = \frac{1}{\Lambda}\sum_{\alpha=1}^{m} \lambda_\alpha D_\alpha(P),$$

where $\Lambda := \sum_{\alpha=1}^{m} \lambda_\alpha$. The average delay per demand is the standard cost functional for queueing systems with multiple classes of demands. Notice that we can write $D(P) = \sum_{\alpha=1}^{m} c_\alpha D_\alpha(P)$ with $c_\alpha = \lambda_\alpha/\Lambda$. Thus, we can model priority among classes by allowing any convex combination of $D_1, \dots, D_m$. If $c_\alpha > \lambda_\alpha/\Lambda$, then the delay of α-demands is being weighted more heavily than in the average case. Thus, the quantity $c_\alpha \Lambda/\lambda_\alpha$ gives the priority of α demands compared to that given in the average delay case. Without loss of generality we can assume that priority classes are labeled so that

$$\frac{c_1}{\lambda_1} \geq \frac{c_2}{\lambda_2} \geq \cdots \geq \frac{c_m}{\lambda_m}, \tag{5.2}$$

implying that if $\alpha < \beta$ for some $\alpha, \beta \in \{1, \dots, m\}$, then the priority of α-demands is at least as high as that of β-demands. With these definitions, we are now ready to state our problem.

> **Problem Statement:** Let Π be the set of all causal, stable and stationary policies for dynamic vehicle routing with priority classes. Given the coefficients $c_\alpha > 0$, $\alpha \in \{1, \dots, m\}$, with $\sum_{\alpha=1}^{m} c_\alpha = 1$, and satisfying equation (5.2), let $D(P) := \sum_{\alpha=1}^{m} c_\alpha D_\alpha(P)$ be the cost of policy $P \in \Pi$. Then, the problem is to determine a vehicle routing policy P^*, if one exists, such that
>
> $$D(P^*) = \inf_{P \in \Pi} D(P). \tag{5.3}$$

We let D^* denote the right-hand side of equation (5.3). A policy P for which $D(P)/D^*$ is bounded has a *constant-factor guarantee*. If $\limsup_{\varrho \to 1^-} D(P)/D^* = \xi < +\infty$, then the policy P has a *heavy-load constant-factor guarantee* of ξ. In this paper we focus on the heavy-load regime, and look for policies with a heavy-load constant-factor guarantee that is *independent* of the number of vehicles, the arrival rates of demands, the on-site service times, and the convex combination coefficients. In the light-load regime, existing policies for the dynamic traveling repairperson can be used, as is summarized in the following remark.

Remark 5.1 (Light-load regime). *In light load, $\varrho \to 0^+$, optimal policies have been developed for the dynamic traveling repairperson problem (i.e., the single-class dynamic vehicle routing problem). These policies rely on the computation of a set of n-median locations for the environment $\mathcal{E}$; that is, a set of n positions $\mathcal{Q}^* \subset \mathcal{E}$, that minimize*

$$\mathbb{E}\left[\min_{\mathbf{q} \in \mathcal{Q}^*} \|\mathbf{q} - \mathbf{q}_0\|\right],$$

where $\mathbf{q}_0$ is a uniformly distributed location in $\mathcal{E}$. In particular, the n Stochastic Queue Median (nSQM) policy, first introduced in (Bertsimas and van Ryzin, 1993a), can be described as follows:

> *Place one vehicle at each of the n-median locations of $\mathcal{E}$. When a demand arrives, assign it to the closest median location, and to the corresponding vehicle. Have each vehicle service its demands in the first-come-first-served order, returning to its median location after each service is completed.*

*In fact, by following the proof in (Bertsimas and van Ryzin, 1993*a*), one can show that the nSQM policy is an optimal policy for dynamic vehicle routing with priority classes. The proof of this statement is omitted in the interest of brevity, and we refer interested readers to (Bertsimas and van Ryzin, 1993*a*) for details.*●

5.2 Lower Bound in Heavy Load

In this section we present two lower bounds on the delay in equation (5.3). The first holds only in heavy load (i.e., as $\varrho \to 1^-$), while the second (less tight) bound holds for all ϱ.

Theorem 5.2 (Heavy-load lower bound)**.** *For every routing policy P,*

$$D(P) \geq \frac{\beta_{\mathrm{TSP}}^2 |\mathcal{E}|}{2n^2 v^2 (1-\varrho)^2} \sum_{\alpha=1}^{m} \left(c_\alpha + 2 \sum_{j=\alpha+1}^{m} c_j \right) \lambda_\alpha \quad \text{as } \varrho \to 1^-, \tag{5.4}$$

where $c_1, \ldots, c_m$ satisfy equation (5.2).

Before proving Theorem 5.2 let us quickly comment on the form of equation (5.4). The right-hand side of equation (5.4) approaches $+\infty$ as $\varrho \to 1^-$. Thus, one should more formally write the inequality with $D(P)(1-\varrho)^2$ on the left-hand side, so that the right-hand side is finite. However, this makes the presentation less readable, and thus, henceforth we adhere to the less formal but more transparent style of equation (5.4).

Proof. Consider a tagged demand i of type α, and let us quantify its total service requirement. The demand requires on-site service time $s_\alpha(i)$. Let us denote by $d_\alpha(i)$ the distance from the location of the demand served prior to i, to i's location. In order to compute a lower bound on the wait-time, we will allow "remote" servicing of some of the demands. For an α-demand i that can be serviced remotely, the travel distance $d_\alpha(i)$ is zero (i.e., a service vehicle can service the ith α-demand from any location by simply stopping for the on-site service time $s_\alpha(i)$). Thus, the wait-time for the modified remote servicing problem provides a lower bound on the wait-time for the problem of interest. To formalize this idea, we introduce the variables $\mathrm{rem}_\alpha \in \{0, 1\}$ for each $\alpha \in \{1, \ldots, m\}$.

If $\text{rem}_\alpha = 0$, then α-demands can be serviced remotely. If $\text{rem}_\alpha = 1$, then α-demands must be serviced on location. We assume that $\text{rem}_\alpha = 1$ for at least one $\alpha \in \{1, \ldots, m\}$. Thus, the total service requirement of α-demand i is $\text{rem}_\alpha d_\alpha(i) + s_\alpha(i)$. The steady-state expected service requirement is $\text{rem}_\alpha \bar{d}_\alpha + s_\alpha$, where $\bar{d}_\alpha := \lim_{i \to +\infty} \mathbb{E}\left[d_\alpha(i)\right]$. In order to maintain stability of the system we must require

$$\frac{1}{n} \sum_{\alpha=1}^{m} \lambda_\alpha \left(\frac{\text{rem}_\alpha \bar{d}_\alpha}{v} + \bar{s}_\alpha \right) < 1. \tag{5.5}$$

Applying the definition of ϱ in equation (5.1), we write equation (5.5) as

$$\sum_{\alpha=1}^{m} \text{rem}_\alpha \lambda_\alpha \bar{d}_\alpha < (1 - \varrho) n v. \tag{5.6}$$

For a stable policy P, let $\bar{N}_\alpha$ represent the steady-state expected number of unserviced α-demands. Then, the expected total number of outstanding demands that require on-site service (i.e., cannot be serviced remotely) is given by $\sum_{j=1}^{m} \text{rem}_j \bar{N}_j$. We now apply a result from the dynamic traveling repairperson problem (see (Xu, 1995), page 23) which states that in heavy load ($\varrho \to 1^-$), if the steady-state number of outstanding demands is N, then a lower bound on expected travel distance between demands is $(\beta_{\text{TSP}}/\sqrt{2})\sqrt{|\mathcal{E}|/N}$. Applying this result we have that

$$\bar{d}_\alpha \geq \frac{\beta_{\text{TSP}}}{\sqrt{2}} \sqrt{\frac{|\mathcal{E}|}{\sum_j \text{rem}_j \bar{N}_j}} =: \bar{d}, \tag{5.7}$$

for each $\alpha \in \{1, \ldots, m\}$. Combining equations (5.6) and (5.7),

$$\frac{\sum_\alpha \text{rem}_\alpha \lambda_\alpha}{nv(1-\varrho)} < \frac{1}{\bar{d}}.$$

Applying the definition of $\bar{d}$, squaring both sides, and rearranging we obtain

$$\frac{\beta_{\text{TSP}}^2}{2} \frac{|\mathcal{E}| (\sum_\alpha \text{rem}_\alpha \lambda_\alpha)^2}{n^2 v^2 (1-\varrho)^2} < \sum_\alpha \text{rem}_\alpha \bar{N}_\alpha.$$

From Little's law, $\bar{N}_\alpha = \lambda_\alpha \mathcal{W}_\alpha$ for each $\alpha \in \{1, \ldots, m\}$, and thus

$$\sum_\alpha \text{rem}_\alpha \lambda_\alpha \mathcal{W}_\alpha > \frac{\beta_{\text{TSP}}^2}{2} \frac{|\mathcal{E}|}{n^2 v^2 (1-\varrho)^2} \left(\sum_\alpha \text{rem}_\alpha \lambda_\alpha \right)^2. \tag{5.8}$$

Recalling that $\mathcal{W}_\alpha = D_\alpha - \bar{s}_\alpha$ and $\text{rem}_\alpha \in \{0, 1\}$ for each $\alpha \in \{1, \ldots, m\}$, we see that equation (5.8) gives us $2^m - 1$ constraints on the feasible values of $D_1(P), \ldots, D_m(P)$. Hence, a lower bound on D^* can be found by minimizing

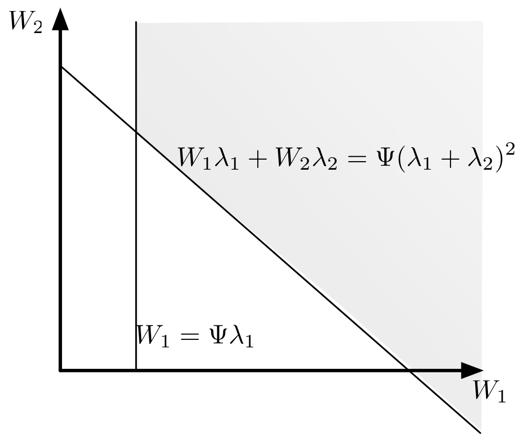


Figure 5.2: The feasible region of the linear program for 2 queues. When class 1 is of higher priority, the solution is given by the corner. Otherwise, the solution is $-\infty$.

$\sum_{\alpha=1}^{m} \mathcal{W}_\alpha$ subject to the constraints in equation (5.8). We can lower bound the solution to the optimization problem by minimizing the cost function subject to only a subset of the $2^m - 1$ constraints. In particular, consider the following linear program

$$\begin{aligned} &\textbf{minimize} && \sum_{\alpha=1}^{m} c_\alpha \mathcal{W}_\alpha, \\ &\textbf{subject to} && \begin{bmatrix} \lambda_1 & 0 & 0 & \cdots & 0 \\ \lambda_1 & \lambda_2 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & & 0 \\ \lambda_1 & \lambda_2 & \lambda_3 & \cdots & \lambda_m \end{bmatrix} \begin{bmatrix} \mathcal{W}_1 \\ \mathcal{W}_2 \\ \vdots \\ \mathcal{W}_m \end{bmatrix} \geq \Psi \begin{bmatrix} \lambda_1^2 \\ (\lambda_1+\lambda_2)^2 \\ \vdots \\ (\lambda_1+\cdots+\lambda_m)^2 \end{bmatrix}, \end{aligned}$$

where

$$\Psi := \frac{\beta_{\mathrm{TSP}}^2}{2} \frac{|\mathcal{E}|}{n^2 v^2 (1-\varrho)^2}.$$

The above problem is feasible (see Fig. 5.2), it has only one basic feasible solution, and it is of the form: minimize $\mathbf{c}^T\mathbf{W}$ subject to $A\mathbf{W} \geq \mathbf{b}$, where $\mathbf{W} = [\mathcal{W}_1, \ldots, \mathcal{W}_m]^T$. Thus, either the problem is unbounded, or the solution $\mathbf{W}^*$ is given by the basic feasible solution. To establish boundedness we consider the dual problem: maximize $\mathbf{b}^T\mathbf{y}$ subject to $A^T\mathbf{y} = \mathbf{c}$ and $\mathbf{y} \geq 0$. By the Duality Theorem of Linear Programming (Luenberger, 1984), if the dual is feasible, then the minimization problem is bounded. To check feasibility of the dual, we solve for $A^T\mathbf{y} = \mathbf{c}$, with $\mathbf{y} \geq 0$, to obtain

$$\begin{aligned} y_\alpha &= \frac{c_\alpha}{\lambda_\alpha} - \frac{c_{\alpha+1}}{\lambda_{\alpha+1}} \geq 0 \quad \text{for all } \alpha \in \{1, \dots, m-1\}, \\ y_m &= \frac{c_m}{\lambda_m} \geq 0. \end{aligned}$$

Thus, the dual is feasible if and only if the priority classes are labeled as in equation (5.2). When equation (5.2) is satisfied, the minimization problem is bounded, and its solution $(\mathcal{W}_1^*, \dots, \mathcal{W}_m^*)$ is given by

$$\mathcal{W}_\alpha^* = \frac{\Psi}{\lambda_\alpha}\left((\lambda_1 + \cdots + \lambda_\alpha)^2 - (\lambda_1 + \cdots + \lambda_{\alpha-1})^2\right) = \Psi\left(\lambda_\alpha + 2\sum_{j=1}^{\alpha-1}\lambda_j\right).$$

(In fact, this is the solution of the full optimization problem with $2^m - 1$ constraints. This fact can be verified, somewhat tediously, by writing the dual of the full problem and directly computing its solution. To shorten the presentation we omit the direct computation and use the above technique.) The optimal value of the cost function, and thus the lower bound on D^*, is given by

$$\sum_{\alpha=1}^m c_\alpha \mathcal{W}_\alpha^* = \Psi \sum_{\alpha=1}^m c_\alpha \left(\lambda_\alpha + 2\sum_{j=1}^{\alpha-1}\lambda_j\right) = \Psi\sum_{\alpha=1}^m \left(c_\alpha + 2\sum_{j=\alpha+1}^m c_j\right)\lambda_\alpha.$$

Applying the definition of Ψ we obtain the desired result. $\square$

Remark 5.3 (Lower bound for all $\varrho \in [0, 1)$). *With slight modifications, it is possible to obtain a less tight lower bound valid for all values of ϱ. In the above derivation, the assumption that $\varrho \to 1^-$ is used only in equation (5.7). It is possible to use, instead, a lower bound valid for all $\varrho \in [0, 1)$ (see (Bertsimas and van Ryzin, 1993a)):*

$$\bar{d}_\alpha \geq \gamma\sqrt{\frac{|\mathcal{E}|}{\sum_\alpha \mathrm{rem}_\alpha N_\alpha + n/2}},$$

where $\gamma = 2/(3\sqrt{2\pi}) \approx 0.266$. Using this bound we obtain the same linear program as in the proof of Theorem 5.2, with the difference that Ψ is now a function given by

$$\Psi(x) := \frac{\gamma^2|\mathcal{E}|}{n^2 v^2 (1-\varrho)^2}x - \frac{n}{2}.$$

Following the procedure in the proof of Theorem 5.2

$$\mathcal{W}_1^* = \frac{\gamma^2|\mathcal{E}|}{n^2v^2(1-\varrho)^2}\lambda_1 - \frac{n}{2\lambda_1}$$
$$\mathcal{W}_\alpha^* = \frac{\gamma^2|\mathcal{E}|}{n^2v^2(1-\varrho)^2}\left(\lambda_\alpha + 2\sum_{j=1}^{\alpha-1}\lambda_j\right),$$

for each $\alpha \in \{2,\dots,m\}$. Finally, for every policy P, $D_\alpha(P) \geq \mathcal{W}_\alpha^* + \bar{s}_\alpha$, and thus

$$D(P) \geq \frac{\gamma^2|\mathcal{E}|}{n^2v^2(1-\varrho)^2}\sum_{\alpha=1}^{m}\left(\left(c_\alpha + 2\sum_{j=\alpha+1}^{m}c_j\right)\lambda_\alpha\right) - \frac{nc}{2\lambda_1} + \sum_{\alpha=1}^{m}c_\alpha\bar{s}_\alpha, \tag{5.9}$$

for all $\varrho \in [0,1)$ under the labeling in equation (5.2). •

In the remainder of the chapter we design a policy and establish a constant-factor guarantee relative to the heavy load lower bound.

5.3 Separate Queues Policy

In this section we introduce and analyze the Separate Queues (SQ) policy. We show that this policy is within a factor $2m^2$ of the lower bound in heavy load.

To present the SQ policy we need some notation. We assume vehicle $k \in \{1,\dots,n\}$ has a service region $\mathcal{R}^{[k]} \subseteq \mathcal{E}$, such that $\{\mathcal{R}^{[1]},\dots,\mathcal{R}^{[n]}\}$ forms a partition of the environment $\mathcal{E}$. In general the partition could be time varying, but for the description of the SQ policy this will not be required. We assume that information on outstanding demands of type $\alpha \in \{1,\dots,m\}$ in region $\mathcal{R}^{[k]}$ at time t is summarized as a finite set of demand positions $Q_\alpha^{[k]}(t)$ with $N_\alpha^{[k]}(t) := \text{card}(Q_\alpha^{[k]}(t))$. Demands of type α with location in $\mathcal{R}^{[k]}$ are inserted in the set $Q_\alpha^{[k]}$ as soon as they are generated. Removal from the set $Q_\alpha^{[k]}$ requires that service vehicle k moves to the demand location, and provides the on-site service. The SQ policy is described in Algorithm 5.1. Fig. 5.3 shows an illustrative example of the SQ policy. In the first two frames the vehicle is servicing only class 1 (circle shaped) demands, whereas in the third frame, the vehicle is servicing class 2 (diamond shaped) demands.

5.3.1 Stability Analysis of the SQ Policy in Heavy Load

In this section we analyze the SQ policy in heavy load, i.e., as $\varrho \to 1^-$. In the SQ policy each region $\mathcal{R}^{[k]}$ has equal area, and contains a single vehicle. Thus,

Algorithm 5.1: Separate Queues (SQ) policy

```
   Assumes: A probability distribution $[p_1, \dots, p_m]$.
1  Partition $\mathcal{E}$ into $n$ equal area regions and assign one vehicle to each region.
2  foreach vehicle-region pair $k$ do
3      if the set $\cup_\alpha Q_\alpha^{[k]}$ is empty then
4          Move vehicle toward the median of its own region until a demand arrives.
5      else
6          Select $Q \in \{Q_1^{[k]}, \dots, Q_m^{[k]}\}$ according to $[p_1, \dots, p_m]$.
7          if $Q$ is empty then
8              Reselect until $Q$ is nonempty.
9          Compute TSP tour through all demands in $Q$.
10         Service $Q$ following the TSP tour, starting at the demand closest to the vehicle's current position.
11     Repeat.
12 Optimize over $[p_1, \dots, p_m]$.
```

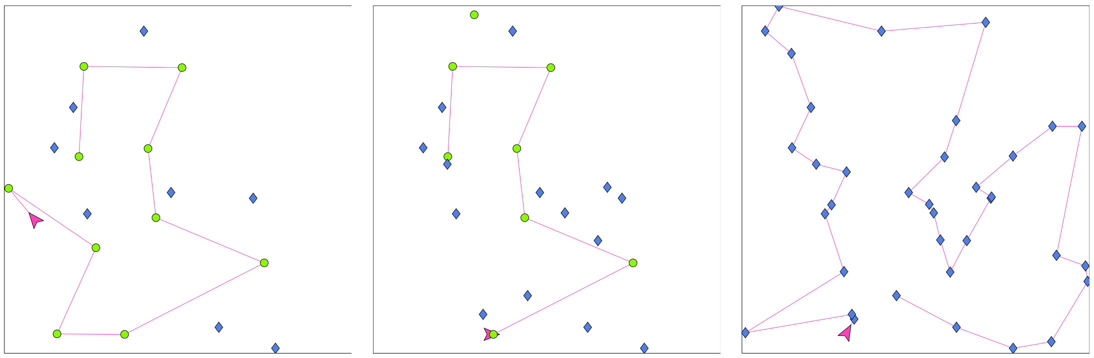


Figure 5.3: A representative simulation of the SQ policy for one vehicle and two priority classes. Circle shaped demands are high priority, and diamond shaped are low priority. The vehicle is marked by a chevron shaped object and TSP tour is shown in a solid line.

the n vehicle problem in a region of area $|\mathcal{E}|$ has been turned into n independent single-vehicle problems, each in a region of area $|\mathcal{E}|/n$, with arrival rates λ_α/n. To determine the performance of the policy we need only study the performance in a single region k. For simplicity of notation we omit the label k. We refer to the time instant t_i in which the vehicle computes a new TSP tour as the epoch i of the policy; we refer to the time interval between epoch i and epoch $i+1$ as the ith iteration and we will refer to its length as Δt_i. Finally, let $N_\alpha(t_i) := N_{\alpha,i}$, $\alpha \in \{1,\dots,m\}$, be the number of outstanding α-demands at beginning of iteration i.

The following straightforward lemma, similar to Lemma 1 in (Papastavrou, 1996), will be essential in deriving our main results.

Lemma 5.4 (Number of outstanding demands)**.** *In heavy load (i.e., $\varrho \to 1^-$), after a transient, the number of demands serviced in a single tour of the vehicle in the SQ policy is very large with high probability (i.e., the number of demands tends to $+\infty$ with probability that tends to 1, as ϱ approaches 1^-).*

Proof. Consider the case where the vehicle moves with infinite velocity (i.e., $v \to +\infty$); then the system is reduced to the usual M/G/1 queue. The infinite-velocity system has fewer demands (for every $\alpha \in \{1,\dots,m\}$) waiting in queue. A known result on M/G/1 queues (Wolff, 1989) states that, after transients, the total number of demands, as $\varrho \to 1^-$, is very large with high probability. Thus, in the SQ policy, the number of demands in all m classes is very large with high probability. In particular, this implies that, after a transient, the number of demands is very large with high probability at the instances when the vehicle starts a new tour. □

Let TS_j be the event that Q_j is selected for service at iteration i of the SQ policy. By the total probability law

$$\mathbb{E}\left[N_{\alpha,i+1}\right] = \sum_{j=1}^{m} p_j \mathbb{E}\left[N_{\alpha,i+1}|\mathrm{TS}_j\right], \quad \alpha \in \{1,\dots,m\},$$

where the conditioning is with respect to the task being performed during iteration i. During iteration i of the policy, demands arrive according to independent Poisson processes. Call $N^{\mathrm{new}}_{\alpha,i}$ the α-demands ($\alpha \in \{1,\dots,m\}$) newly arrived during iteration i; then, by definition of the SQ policy

$$\mathbb{E}\left[N_{\alpha,i+1}|\mathrm{TS}_j\right] = \begin{cases} \mathbb{E}\left[N^{\mathrm{new}}_{\alpha,i}|\mathrm{TS}_j\right], & \text{if } \alpha = j \\ \mathbb{E}\left[N_{\alpha,i}|\mathrm{TS}_j\right] + \mathbb{E}\left[N^{\mathrm{new}}_{\alpha,i}|\mathrm{TS}_j\right], & \text{otherwise.} \end{cases}$$

By the law of iterated expectation, we have $\mathbb{E}\left[N^{\mathrm{new}}_{\alpha,i}|\mathrm{TS}_j\right] = (\lambda_\alpha/n)\mathbb{E}\left[\Delta t_i|\mathrm{TS}_j\right]$.

Moreover, since the number of demands outstanding at the beginning of iteration i is independent of the task that will be chosen, we have $\mathbb{E}[N_{\alpha,i}|\mathrm{TS}_j] = \mathbb{E}[N_{\alpha,i}]$. Thus we obtain

$$\mathbb{E}[N_{\alpha,i+1}|\mathrm{TS}_j] = \begin{cases} \frac{\lambda_\alpha}{n}\mathbb{E}[\Delta t_i|\mathrm{TS}_j], & \text{if } \alpha = j \\ \mathbb{E}[N_{\alpha,i}] + \frac{\lambda_\alpha}{n}\mathbb{E}[\Delta t_i|\mathrm{TS}_j], & \text{otherwise.} \end{cases}$$

Therefore, we are left with computing the conditional expected values of Δt_i. The length of Δt_i is given by the time needed by the vehicle to travel along the TSP tour plus the time spent to service demands. Assuming i large enough, Lemma (5.4) holds, and we can apply Theorem 2.3 to estimate from the quantities $N_{\alpha,i}$, $\alpha \in \{1, \ldots, m\}$, the length of the TSP tour at iteration i. Conditioning on TS_j (when only demands of type j are serviced), we have

$$\begin{aligned} \mathbb{E}[\Delta t_i|\mathrm{TS}_j] &= \frac{\beta_{\mathrm{TSP}}\sqrt{|\mathcal{E}|/n}}{v}\mathbb{E}\left[\sqrt{N_{j,i}}|\mathrm{TS}_j\right] + \mathbb{E}\left[\textstyle\sum_{k=1}^{N_{j,i}} s_{j,k}|\mathrm{TS}_j\right] \\ &\le \frac{\beta_{\mathrm{TSP}}\sqrt{|\mathcal{E}|/n}}{v}\sqrt{\mathbb{E}[N_{j,i}]} + \mathbb{E}[N_{j,i}]\bar{s}_j, \end{aligned}$$

where we have

(i) applied equation (2.1);

(ii) applied Jensen's inequality for concave functions, in the form $\mathbb{E}\left[\sqrt{X}\right] \le \sqrt{\mathbb{E}[X]}$;

(iii) removed the conditioning on TS_j, since the random variables $N_{\alpha,i}$ are independent from *future* events, and in particular from the choice of the task at iteration i; and

(iv) used the crucial fact that the on-site service times are independent from the number of outstanding demands.

Collecting the above results (and using the shorthand $\bar{X}$ to indicate $\mathbb{E}[X]$, where X is any random variable), we have

$$\bar{N}_{\alpha,i+1} \le (1 - p_\alpha)\bar{N}_{\alpha,i} + \sum_{j=1}^{m} p_j \frac{\lambda_\alpha}{n}\left[\frac{\beta_{\mathrm{TSP}}\sqrt{|\mathcal{E}|}}{\sqrt{n}v}\sqrt{\bar{N}_{j,i}} + \bar{N}_{j,i}\bar{s}_j\right], \tag{5.10}$$

for each $\alpha \in \{1, \ldots, m\}$. The m inequalities above describe a system of recursive relations that describe an upper bound on $\bar{N}_{\alpha,i}$, $\alpha \in \{1, \ldots, m\}$. The following theorem bounds the values to which they converge.

Theorem 5.5 (Steady-state queue length). *For every set of initial conditions $\{\bar{N}_{\alpha,0}\}_{\alpha\in\{1,\dots,m\}}$, the trajectories $i \mapsto \bar{N}_{\alpha,i}$, $\alpha \in \{1,\dots,m\}$, resulting from equations (5.10), satisfy*

$$\limsup_{i\to+\infty} \bar{N}_{\alpha,i} \le \frac{\beta_{\mathrm{TSP}}^2 |\mathcal{E}|}{n^3 v^2 (1-\varrho)^2} \frac{\lambda_\alpha}{p_\alpha} \left(\sum_{j=1}^m \sqrt{\lambda_j p_j} \right)^2, \quad \text{as } \varrho \to 1^-.$$

Proof. Define $q_j := 1 - p_j$ and let $\hat{\lambda}_\alpha$ denote the arrival rate in region $\mathcal{R}^{[k]}$. Thus $\hat{\lambda}_\alpha := \lambda_\alpha / n$ for each $\alpha \in \{1,\dots,m\}$. Let $x(i) := (\bar{N}_{1,i}, \bar{N}_{2,i}, \dots, \bar{N}_{m,i}) \in \mathbb{R}^m$ and define two matrices

$$A := \begin{bmatrix} \hat{\lambda}_1 p_1 \bar{s}_1 + q_1 & \hat{\lambda}_1 p_2 \bar{s}_2 & \dots & \hat{\lambda}_1 p_m \bar{s}_m \\ \hat{\lambda}_2 p_1 \bar{s}_1 & \hat{\lambda}_2 p_2 \bar{s}_2 + q_2 & \dots & \hat{\lambda}_2 p_m \bar{s}_m \\ \vdots & & \ddots & \vdots \\ \hat{\lambda}_m p_1 \bar{s}_1 & \hat{\lambda}_m p_2 \bar{s}_2 & \dots & \hat{\lambda}_m p_m \bar{s}_m + q_m \end{bmatrix},$$

and

$$B := \frac{\beta_{\mathrm{TSP}} \sqrt{|\mathcal{E}|}}{\sqrt{n} v} \begin{bmatrix} \hat{\lambda}_1 p_1 & \hat{\lambda}_1 p_2 & \dots & \hat{\lambda}_1 p_m \\ \hat{\lambda}_2 p_1 & \hat{\lambda}_2 p_2 & \dots & \hat{\lambda}_2 p_m \\ \vdots & & \ddots & \vdots \\ \hat{\lambda}_m p_1 & \hat{\lambda}_m p_2 & \dots & \hat{\lambda}_m p_m \end{bmatrix}.$$

Then, letting the relation "$\le$" in $\mathbb{R}^m$ denote the product order of m copies of $\mathbb{R}$ (in other words, for v, $w \in \mathbb{R}^m$, the relation $v \le w$ is interpreted component-wise), equations (5.10) can be written as

$$x(i+1) \le A x(i) + B \begin{bmatrix} \sqrt{x_1(i)} \\ \sqrt{x_2(i)} \\ \vdots \\ \sqrt{x_m(i)} \end{bmatrix} =: f(x(i)), \tag{5.11}$$

where $f : \mathbb{R}_{\ge 0} \to \mathbb{R}_{\ge 0}$, and $x_j(i)$, $j \in \{1,\dots,m\}$, are the components of vector $x(i)$. We refer to the discrete system in equation (5.11) as System-X. Next we define two auxiliary systems, System-Y and System-Z. We define System-Y as

$$y(i+1) = f(y(i)). \tag{5.12}$$

System-Y is, therefore, equal to System-X, with the exception that we replaced the inequality with an equality.

Pick, now, any $\varepsilon > 0$. From Young's inequality

$$\sqrt{a} \leq \frac{1}{4\varepsilon} + \varepsilon a, \quad \text{for all } a \in \mathbb{R}_{\geq 0}. \tag{5.13}$$

Hence, for $i \mapsto y(i) \in \mathbb{R}^m_{\geq 0}$, the equation (5.12) becomes

$$\begin{aligned} y(i+1) &\leq Ay(i) + B\Big(\frac{1}{4\varepsilon}\mathbf{1}_m + \varepsilon\, y(i)\Big) \\ &= \Big(A + \varepsilon B\Big)y(i) + \frac{1}{4\varepsilon}B\mathbf{1}_m. \end{aligned}$$

where $\mathbf{1}_m$ is the vector $(1, 1, \ldots, 1)^{\mathrm{T}} \in \mathbb{R}^m$. Next, define System-Z as

$$z(i+1) = \Big(A + \varepsilon B\Big)z(i) + \frac{1}{4\varepsilon}B\mathbf{1}_m =: g(z(i)). \tag{5.14}$$

The proof now proceeds as follows. First, we show that if $x(0) = y(0) = z(0)$, then

$$x(i) \leq y(i) \leq z(i), \quad \text{for all } i \geq 0. \tag{5.15}$$

Second, we show that the trajectories of System-Z are bounded; this fact, together with equation (5.15), implies that also trajectories of System-Y and System-X are bounded. Third, and last, we will compute $\limsup_{i\to+\infty} y(i)$; this quantity, together with equation (5.15), will yield the desired result.

Let us consider the first issue. We have $y(1) = f(y(0))$ and $z(1) = g(z(0))$. Since, by assumption $z(0) = y(0)$, we have that $g(z(0)) = g(y(0)) \geq f(y(0))$, where the last inequality follows from equation (5.13) and by definition of f and g . Therefore, we get $y(1) \leq z(1)$. Then, we have $y(2) = f(y(1))$ and $z(2) = g(z(1))$. Since $z(1), y(1) \in \mathbb{R}^m_{\geq 0}$, and the elements in matrices A and B are all non-negative, then $y(1) \leq z(1)$ implies $g(y(1)) \leq g(z(1))$. Using similar arguments, we can write $z(2) \geq g(y(1)) \geq f(y(1)) = x(2)$; therefore, we get $y(2) \leq z(2)$. Then, it is immediate by induction that $y(i) \leq z(i)$ for all $i \geq 0$.

Similarly, we have $x(1) \leq f(x(0)) = f(y(0)) = y(1)$, where we have used the assumption $x(0) = y(0)$. Then, we get $x(1) \leq y(1)$. Since $x(1), y(1) \in \mathbb{R}^m_{\geq 0}$, the elements in matrices A and B are nonnegative, and by the monotonicity of $\sqrt{\cdot}$, then $x(1) \leq y(1)$ implies $f(x(1)) \leq f(y(1))$. Therefore, we can write $x(2) \leq f(x(1)) \leq f(y(1)) = y(2)$; thus, we get $x(2) \leq y(2)$. Then, it is immediate to show by induction that $x(i) \leq y(i)$ for all $i \geq 0$, and equation (5.15) holds.

We now turn our attention to the second issue, namely boundedness of trajectories for System-Z (in equation (5.14)). Notice that System-Z is a discrete-time linear system. The eigenvalues of A are characterized in the following lemma.

Lemma 5.6. *The eigenvalues of A are real and have magnitude strictly less than* 1 *(i.e., A is a stable matrix).*

Proof. Let $w \in \mathbb{C}^m$ be an eigenvector of A, and $\mu \in \mathbb{C}$ be the corresponding eigenvalue. Then we have $Aw = \mu w$. Define $r := (p_1\bar{s}_1, p_2\bar{s}_2, \ldots, p_m\bar{s}_m)$. Then the m eigenvalue equations are

$$\hat{\lambda}_j\, w \cdot r + q_j w_j = \mu\, w_j, \quad j \in \{1, \ldots, m\}, \tag{5.16}$$

where $w \cdot r$ is the scalar product of vectors w and r, and w_j is the jth component of w.

There are two possible cases. If $w \cdot r = 0$, then equation (5.16) becomes $q_j\, w_j = \mu\, w_j$, for all j. Since $w \neq 0$, there exists j^* such that $w_j^* \neq 0$; thus, we have $\mu = q_{j^*}$. Since $q_{j^*} \in \mathbb{R}$ and $0 < q_{j^*} < 1$, we have that μ is real and $|\mu| < 1$.

Assume, now, that $w \cdot r \neq 0$. This implies that $\mu \neq q_j$ and $w_j \neq 0$ for all j, thus we can write, for all j,

$$w_j = \frac{\hat{\lambda}_j}{\mu - q_j}\, w \cdot r, \tag{5.17}$$

and hence

$$w_j = \frac{\hat{\lambda}_j}{\hat{\lambda}_1}\frac{\mu - q_1}{\mu - q_j} w_1.$$

Therefore, (5.17) can be rewritten as

$$\sum_{j=1}^{m} \frac{\mathrm{rem}_j \hat{\lambda}_j}{\mu - q_j} = 1. \tag{5.18}$$

Equation (5.18) implies that the eigenvalues are real. To see this, write $\mu = a + ib$, where i is the imaginary unit: then

$$\sum_{j=1}^{m} \frac{\mathrm{rem}_j \hat{\lambda}_j}{a + ib - q_j} = \sum_{j=1}^{m} \frac{\mathrm{rem}_j \hat{\lambda}_j [(a - q_j) - ib]}{(a - q_j)^2 + b^2}.$$

Thus equation (5.18) implies

$$b \underbrace{\sum_{j=1}^{m} \frac{\mathrm{rem}_j \hat{\lambda}_j}{(a - q_j)^2 + b^2}}_{>0} = 0,$$

that is, $b = 0$. Equation (5.18) also implies that the eigenvalues (that are real) have magnitude strictly less than 1. Indeed, assume, by contradiction, that $\mu \geq 1$. Then we have $\mu - q_j \geq 1 - q_j > 0$ (recall that the eigenvalues are real and $0 < q_j < 1$) and we can write

$$\sum_{j=1}^{m} \frac{\mathrm{rem}_j \hat{\lambda}_j}{\mu - q_j} \leq \sum_{j=1}^{m} \frac{\mathrm{rem}_j \hat{\lambda}_j}{1 - q_j} = \sum_{j=1}^{m} \bar{s}_j \hat{\lambda}_j = \varrho < 1,$$

which is a contradiction. Assume, again by contradiction, that $\mu \leq -1$. In this case we trivially get another contradiction $\sum_{j=1}^{m} \mathrm{rem}_j \hat{\lambda}_j / (\mu - q_j) < 0$, since $\mu - q_j < 0$. □

Hence, $A \in \mathbb{R}^{m \times m}$ has eigenvalues strictly inside the unit disk, and since the eigenvalues of a matrix depend continuously on the matrix entries, there exists a sufficiently small $\varepsilon > 0$ such that the matrix $A + \varepsilon B$ has eigenvalues strictly inside the unit disk. Accordingly, each solution $i \mapsto z(i) \in \mathbb{R}^m_{\geq 0}$ of System-Z converges exponentially fast to the unique equilibrium point

$$z^* = \Big(I_m - A - \varepsilon B \Big)^{-1} \frac{1}{4\varepsilon} B \mathbf{1}_m. \tag{5.19}$$

Combining equation (5.15) with the previous statement, we see that the solutions $i \mapsto x(i)$ and $i \mapsto y(i)$ are bounded. Thus

$$\limsup_{i \to +\infty} x(i) \leq \limsup_{i \to +\infty} y(i) < +\infty. \tag{5.20}$$

Finally, we turn our attention to the third issue, namely the computation of $y := \limsup_{i \to +\infty} y(i)$. Taking the lim sup of the left- and right-hand sides of equation (5.12), and noting that

$$\limsup_{i \to +\infty} \sqrt{y_\alpha(i)} = \sqrt{\limsup_{i \to +\infty} y_\alpha(i)} \quad \text{for } \alpha \in \{1, 2, \dots, m\},$$

since $x \mapsto \sqrt{x}$ is continuous and strictly monotone increasing on $\mathbb{R}_{>0}$, we obtain that

$$y_\alpha = (1 - p_\alpha) y_\alpha + \hat{\lambda}_\alpha \sum_{j=1}^{m} p_j \left(\frac{\beta_{\mathrm{TSP}} \sqrt{|\mathcal{E}|}}{\sqrt{nv}} \sqrt{y_j} + \bar{s}_j y_j \right).$$

Rearranging we obtain

$$p_\alpha y_\alpha = \hat{\lambda}_\alpha \sum_{j=1}^{m} p_j \left(\frac{\beta_{\mathrm{TSP}} \sqrt{|\mathcal{E}|}}{\sqrt{nv}} \sqrt{y_j} + \bar{s}_j y_j \right). \tag{5.21}$$

Dividing $p_\alpha y_\alpha$ by $p_1 y_1$ we obtain

$$y_\alpha = \frac{\hat{\lambda}_\alpha p_1}{\hat{\lambda}_1 p_\alpha} y_1. \tag{5.22}$$

Combining equations (5.21) and (5.22), we obtain

$$p_1 y_1 = \varrho\, p_1 y_1 + \frac{\beta_{\mathrm{TSP}} \sqrt{|\mathcal{E}|}}{\sqrt{nv}} \sqrt{p_1 \hat{\lambda}_1 y_1} \sum_{j=1}^{m} \sqrt{\hat{\lambda}_j p_j}$$

Thus, recalling that $\hat{\lambda}_\alpha = \lambda_\alpha / n$, we obtain

$$y_\alpha = \frac{\beta_{\mathrm{TSP}}^2 |\mathcal{E}|}{n^3 v^2 (1-\varrho^2)} \frac{\lambda_\alpha}{p_\alpha} \left(\sum_{j=1}^{m} \sqrt{\lambda_j p_j} \right)^2 .$$

Noting that from equation (5.20), $\limsup_{i \to +\infty} N_{\alpha,i} \leq y_\alpha$, we obtain the desired result. □

5.3.2 Delay of the SQ Policy in Heavy Load

From Theorem 5.5, and using Little's law, the delay of α-demands satisfies

$$\begin{aligned} D_\alpha(\mathrm{SQ}) &\leq \frac{n}{\lambda_\alpha} \limsup_{i \to +\infty} \bar{N}_{\alpha,i} + \bar{s}_\alpha \\ &= \frac{\beta_{\mathrm{TSP}}^2 |\mathcal{E}|}{n^2 v^2 (1-\varrho)^2} \frac{1}{p_\alpha} \left(\sum_{j=1}^{m} \sqrt{\lambda_j p_j} \right)^2 . \end{aligned}$$

Thus, the delay (as defined in equation (5.3)) of the SQ policy, satisfies

$$D(\mathrm{SQ}) \leq \frac{\beta_{\mathrm{TSP}}^2 |\mathcal{E}|}{n^2 v^2 (1-\varrho)^2} \sum_{\alpha=1}^{m} \frac{c_\alpha}{p_\alpha} \left(\sum_{i=1}^{m} \sqrt{\lambda_i p_i} \right)^2, \quad \text{as } \varrho \to 1^-. \tag{5.23}$$

With this expression we prove our main result on the performance of the SQ policy.

Theorem 5.7 (SQ policy performance)**.** *As $\varrho \to 1^-$, the delay of the SQ policy is within a factor $2m^2$ of the optimal delay. This factor is independent of the arrival rates $\lambda_1, \dots, \lambda_m$, coefficients $c_1, \dots, c_m$, service times $\bar{s}_1, \dots, \bar{s}_m$, and the number of vehicles n.*

Proof. We would like to compare the performance of this policy with the lower bound. To do this, consider setting

$$p_\alpha := c_\alpha \quad \text{for each } \alpha \in \{1, \dots, m\}.$$

Defining $\Psi := \beta_{\mathrm{TSP}}^2 |\mathcal{E}|/(n^2 v^2 (1-\varrho)^2)$, equation (5.23) can be written as

$$D(\mathrm{SQ}) \leq \Psi m \left(\sum_{i=1}^m \sqrt{c_i \lambda_i} \right)^2 .$$

Next, the lower bound in equation (5.4) is

$$D^* \geq \frac{\Psi}{2} \sum_{i=1}^m \left(c_i + 2 \sum_{j=i+1}^m c_j \right) \lambda_i \geq \frac{\Psi}{2} \sum_{i=1}^m (c_i \lambda_i) .$$

Thus, comparing the upper and lower bounds

$$\frac{D(\mathrm{SQ})}{D^*} \leq 2m \frac{\left(\sum_{i=1}^m \sqrt{c_i \lambda_i} \right)^2}{\sum_{i=1}^m (c_i \lambda_i)} . \tag{5.24}$$

Letting $x_i := \sqrt{c_i \lambda_i}$, and $\mathbf{x} := [x_1, \dots, x_m]$, the numerator of the fraction in equation (5.24) is $\|\mathbf{x}\|_1^2$, and the denominator is $\|\mathbf{x}\|_2^2$. But the one- and two-norms of a vector $\mathbf{x} \in \mathbb{R}^m$ satisfy $\|\mathbf{x}\|_1 \leq \sqrt{m} \|\mathbf{x}\|_2$. Thus,

$$\frac{D(\mathrm{SQ})}{D^*} \leq 2m \left(\frac{\|\mathbf{x}\|_1}{\|\mathbf{x}\|_2} \right)^2 \leq 2m^2, \quad \text{as } \varrho \to 1^-,$$

and the policy is a $2m^2$-factor approximation. □

Remark 5.8 (Relation to Randomized Priority policy in (Smith et al., 2008))**.** *For $m = 2$ the SQ policy is within a factor of 8 of the optimal. This improves on the factor of 12 obtained for the* Randomized Priority *policy in (Smith et al., 2008). However, it appears that the* Randomized Priority *policy bound is not tight, since for two classes, simulations indicate it performs no worse than the SQ policy.* •

5.3.3 Separate Queues Policy with Queue Merging

In this section we propose a modification the SQ policy based on *queue merging*. Queue merging is guaranteed to never increase the upper bound on the expected delay, and in certain instances it significantly decreases the upper bound. The modification can be used when we have a modest number of classes (fewer than, say, 20), which encompasses most applications of interest.

To motivate the modification, consider the case when all classes have equal priority (i.e., $c_1/\lambda_1 = \cdots = c_m/\lambda_m$), and we use the probability assignment $p_\alpha = c_\alpha$ for each class α. Then, the upper bound for the Separate Queues policy in equation (5.23) becomes

$$\Psi m(\lambda_1 + \cdots + \lambda_m),$$

where $\Psi := \beta_{\mathrm{TSP}}^2 |\mathcal{E}|/(n^2 v^2 (1-\varrho)^2)$.

On the other hand, if we ignore priorities, merge the m classes into a single class, and run the SQ policy on the merged class (i.e., at each iteration, service all outstanding demands in $\mathcal{E}$ via the TSP tour), then the upper bound becomes

$$\Psi(\lambda_1 + \cdots + \lambda_m).$$

Thus, there is a factor of m separating the two upper bounds. This is due to the fact that the basic SQ policy services each of the m classes separately, even when they have the same priority.

The above discussion motivates the addition of queue merging to the SQ policy. We define a *merge configuration* to be a partition of m classes $\{1, \dots, m\}$ into ℓ sets $C_1, \dots, C_\ell$, where $\ell \in \{1, \dots, m\}$. The upper bound for a merge configuration $\{C_1, \dots, C_\ell\}$ is

$$\Psi \ell \left(\sum_{i}^{\ell} \sqrt{\sum_{\alpha \in C_i} c_\alpha \sum_{\beta \in C_i} \lambda_\beta} \right)^2. \tag{5.25}$$

The SQ-policy with merging can be summarized as follows:

```
Algorithm 5.2: SEPARATE QUEUES WITH MERGING policy
1  Find the merge configuration $\{C_1, \dots, C_\ell\}$ which minimizes equation (5.25).
2  Run the SEPARATE QUEUES policy on $\ell$ classes, where class $i$ has arrival
   rate $\sum_{\alpha \in C_i} \lambda_\alpha$ and convex combination coefficient $\sum_{\alpha \in C_i} c_\alpha$.
```

Now, to minimize equation (5.25) in step 1 of the Separate Queues with Merging policy, one must search over all possible partitions of a set of m elements. The number of partitions is given by the Bell Number B_m which is defined recursively as $B_m = \sum_{k=0}^{m-1} B_k \binom{m-1}{k}$. Thus, the search becomes infeasible for more than 10 classes.

If the search space is too large, then one can limit the search to all partitions such that if $i < j$, then each class in C_i has higher priority than all classes in C_j. This is the set of partitions in which only adjacent classes are merged. For m

classes, there are 2^{m-1} such merge configurations, which is significantly less than the Bell number B_m, but is still infeasible more than, say, 20 classes.

5.3.4 The "Tube" Heuristic for Improving Performance

We now introduce a simple heuristic improvement for the SQ policy that can be used for implementation. The heuristic improvement is as follows:

> **Tube Heuristic:** When following the tour in step 10 of the SQ policy, service all newly arrived demands that lie within distance $\epsilon > 0$ of the tour.

The idea behind the heuristic is to utilize the fact that some newly arrived demands will be "close" to the demands in the current service batch, and thus can be serviced with minimal additional travel cost. Analysis of the tube heuristic is complicated by the fact that it introduces correlation between demand locations. A similar difficulty arises when attempting to analyze the nearest neighbor policy (Bertsimas and van Ryzin, 1991). However, we can demonstrate the effectiveness of this heuristic through simulations.

The parameter ϵ should be chosen such that the total tour length is not increased by more than, say, 10%. A rough calculation shows that the area of the "tube" of width 2ϵ centered around a tour that passes through the $\text{card}(Q)$ demands in Q, has area upper bounded by $2\epsilon\beta_{\text{TSP}}\sqrt{\text{card}(Q)|\mathcal{E}|}$. While following the tour, a vehicle will deviate to service no more than $2\epsilon\beta_{\text{TSP}}\sqrt{\text{card}(Q)/|\mathcal{E}|}(\bar{N}_1 + \cdots + \bar{N}_m)$ demands. Finally, since the vehicle will have to travel no more than 2ϵ to service each demand in the "tube," we see that ϵ should scale as

$$\epsilon \sim \sqrt{\frac{f|\mathcal{E}|}{\bar{N}_1 + \cdots + \bar{N}_m}},$$

where $\bar{N}_\alpha$ is expected number of α-demands in the environment, and f is the fractional increase in tour length (e.g., 10%).

Fig. 5.4 shows numerical results for the Tube Heuristic for a single unit speed vehicle in a square environment with side length 50. The simulation is performed for two classes of demands with $c = 0.8$, $\lambda_2 = 6\lambda_1$, and several different load factors ϱ. Each experimental data point represents the average of the steady state delay of ten runs, where each run consists of 200 iterations of the SQ policy. To ensure convergence to steady state and avoid effects due to the transient response, only the last 50 iterations in each run are used to calculate the delay. The basic policy is shown in left-most data points (i.e., $\epsilon = 0$). Fig. 5.4 demonstrates that as the load factor increases, the value of ϵ should be chosen smaller in order to

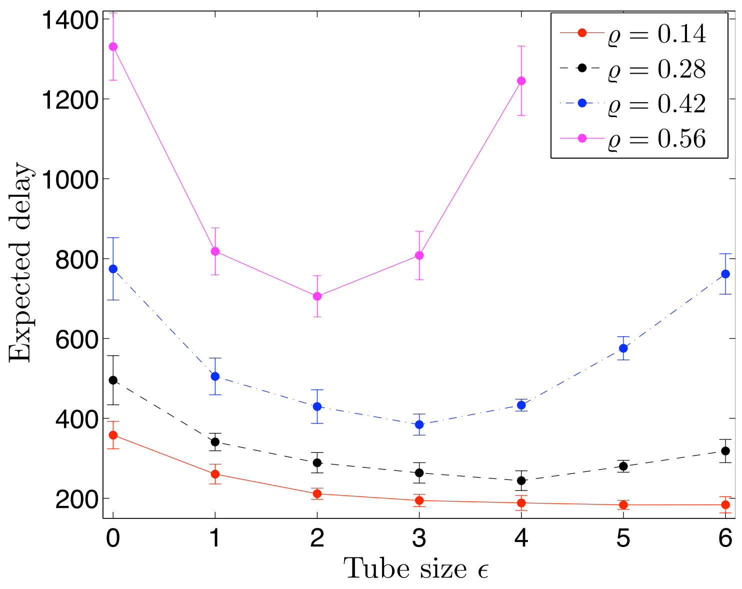


Figure 5.4: The tube heuristic for two classes of demands with $c = 0.8$, $\lambda_2 = 6\lambda_1$, and several different load factors ϱ. The delay at $\epsilon = 0$ corresponds to the basic SQ policy.

achieve the best performance. Table 5.1 shows the improvement in expected delay when using the tube heuristic. The heuristic appears to decrease the delay by a factor of 2. One should note that the heuristic is difficult to accurately simulate for high load factors. This is due to the additional computations required to determine if a newly arrived demand lies within a distance ϵ of the current tour. A more sophisticated implementation of tube heuristic is to define an ϵ_α for each

Load factor ϱ	Delay	Best ϵ	Delay with best ϵ	Heuristic improvement
0.14	358 (34)	5	183 (11)	0.51 (0.16)
0.28	496 (61)	4	244 (25)	0.49 (0.23)
0.42	774 (78)	3	384 (26)	0.50 (0.17)
0.56	1330 (84)	2	706 (52)	0.53 (0.14)
0.70	3380 (357)	1	1770 (121)	0.52 (0.17)

Table 5.1: A comparison between the expected delay of the basic SQ policy, and the SQ policy with the tube heuristic. The values in brackets give the standard deviation of the corresponding table entry.

$\alpha \in \{1, \dots, m\}$, where the magnitude of ϵ_α is proportional to its priority, and thus proportional to the probability p_α.

5.4 Simulations and Discussion

In this section we discuss, through the use of simulations, the performance of the SQ policy with the probability assignment $p_\alpha := c_\alpha$, for each $\alpha \in \{1, \dots, m\}$. In particular, we study (i) the tightness of the upper bound in equation (5.23), (ii) conditions for which the gap between the lower bound in equation (5.4) and the upper bound in equation (5.23) is maximized, (iii) the suboptimality of the probability assignment $p_\alpha = c_\alpha$, and (iv) the difference in performance between the SQ policy and a policy that merges all classes together irrespective of priorities. Simulations of the SQ policy were performed using `linkern` as a solver to generate approximations to the optimal TSP tour (see Chapter 2 for more information on computing TSP tour approximations).

5.4.1 Tightness of the Upper Bound

We consider one vehicle, four classes of demands, and several values of the load factor ϱ. For each value of ϱ we perform 100 runs. In each run we uniformly randomly generate arrival rates $\lambda_1, \dots, \lambda_m$, convex combination coefficients $c_1, \dots, c_m$, and on-site service times $\bar{s}_1, \dots, \bar{s}_m$, and normalize the values such that the constraints $\sum_{\alpha=1}^{m} \lambda_\alpha \bar{s}_\alpha = \varrho$ and $\sum_{\alpha=1}^{m} c_\alpha = 1$ are satisfied. In each run we iterate the SQ policy 4000 times, and compute the steady-state expected delay by considering the number of demands in the last 1000 iterations. For each value of ϱ we compute the ratio χ between the expected delay and the theoretical upper bound in equation (5.23). Table 5.2 reports the ratio, its standard deviation, and its minimum and maximum values for each ϱ value. One can see that the upper bound provides a reasonable approximation for load factors as low as $\varrho = 0.75$.

Load factor (ϱ)	$\mathbb{E}[\chi]$	σ_χ	$\max \chi$	$\min \chi$
0.75	0.803	0.092	1.093	0.354
0.8	0.778	0.108	0.943	0.256
0.85	0.773	0.111	1.150	0.417
0.9	0.733	0.159	1.162	0.203
0.95	0.716	0.131	0.890	0.257

Table 5.2: Ratio χ between experimental results and upper bound for various values of ϱ.

5.4.2 Maximum deviation from lower bound

In Theorem 5.7 we showed that the SQ policy performs within a factor of $2m^2$ of the lower bound for all initial conditions. The difference between the upper bound equation (5.23) and the lower bound in equation (5.4) can be made arbitrarily

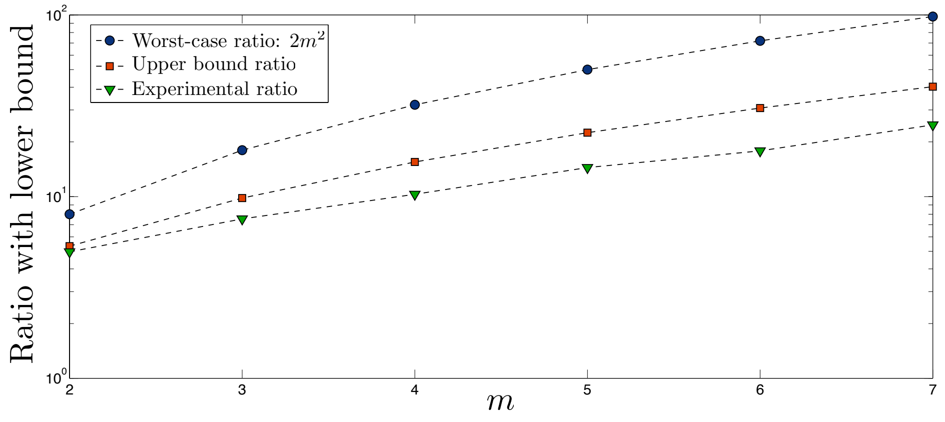


Figure 5.5: Experimental results for the SQ policy in worst-case conditions; $\varrho = 0.85$ and $\lambda_1 = 1$.

close to $2m^2$ by choosing $\lambda_1 \ll \lambda_2 \ll \cdots \ll \lambda_m$ and $c_1 \gg c_2 \gg \cdots \gg c_m$, with $\lambda_\alpha c_\alpha = a$, for each $\alpha \in \{1, \ldots, m\}$ and for some positive constant a. In this case, the upper bound is equal to $Bm^3 a$ and the lower bound is approximately $Bma/2$. To test the deviation experimentally we simulated the SQ policy for several values of ϱ with initial conditions of $\lambda_m = b\lambda_{m-1} = b^2\lambda_{m-1} = \cdots = b^{m-1}\lambda_1$ and $c_1 = bc_2 = \cdots = b^{m-1}c_m$, where $b = 2$. Fig. 5.5 shows that the experimentally determined ratio of delays (averaged over 10 simulation runs) indeed increases proportionally to m^2.

5.4.3 Suboptimality of the Approximate Probability Assignment

To prove Theorem 5.7 we used the probability assignment

$$p_\alpha := c_\alpha \quad \text{for each } \alpha \in \{1, \ldots, m\}. \tag{5.26}$$

However, one would like to select $[p_1, \ldots, p_m]$ that minimizes the right-hand side of equation (5.23). The minimization of the right-hand side of equation (5.23) is a constrained multi-variable nonlinear optimization problem over $[p_1, \ldots, p_m]$, that is, in m dimensions. Thus, for a general m class problem, solving the optimization problem is difficult. However, for two classes of demands the optimization is over a single variable p_1 (with the constraint that $p_2 = 1 - p_1$), and it can be readily solved. A comparison of optimized upper bound, denoted upbd_{opt}, with the upper bound obtained using the probability assignment in equation (5.26), denoted upbd_c, is shown in Fig. 5.6. In this figure the ratio of upper bounds is bounded by two.

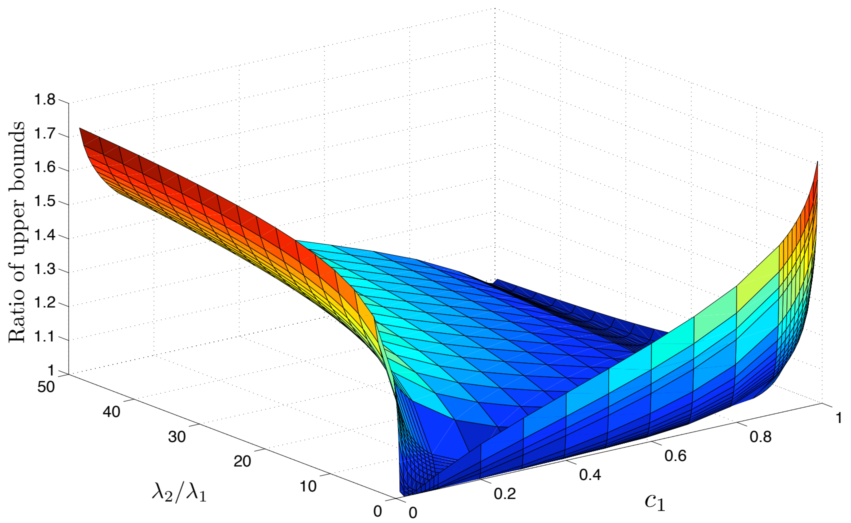


Figure 5.6: The ratios $\text{upbd}_c/\text{upbd}_{opt}$ for 2 classes of demands.

Number of classes (m)	$\text{upbd}_c/\text{upbd}_{\text{local opt}}$	Max. % variation in ratio
3	1.60	0.12
4	1.51	0.04
5	1.51	0.08
6	1.74	0.02
7	1.88	0.08
8	1.63	0.15

Table 5.3: Ratio of upper bound with $p_\alpha = c_\alpha$ for each $\alpha \in \{1, \dots, m\}$ and the upper bound with a locally optimized probability assignment.

For $m > 2$ we approximate the solution of the optimization problem as follows. For each value of m we perform 1000 runs. In each run we randomly generate $\lambda_1, \dots, \lambda_m$, $c_1, \dots, c_m$, and five sets of initial probability assignments. From each initial probability assignment we use a line search to locally optimize the probability assignment. We take the ratio between upbd_c and the least upper bound $\text{upbd}_{\text{local opt}}$ obtained from the five locally optimized probability assignments. We also record the maximum variation in the five locally optimized upper bounds. This is summarized in Table 5.3. The second column shows the largest ratio obtained over the 1000 runs. The third column shows the largest % variation in the 1000 runs. The assignment in equation (5.26) seems to perform within a factor of two of the optimized assignment, and the optimization appears to converge to values close to a global optimum since all five random conditions converge to values that are within $\sim 0.1\%$ of each other on every run.

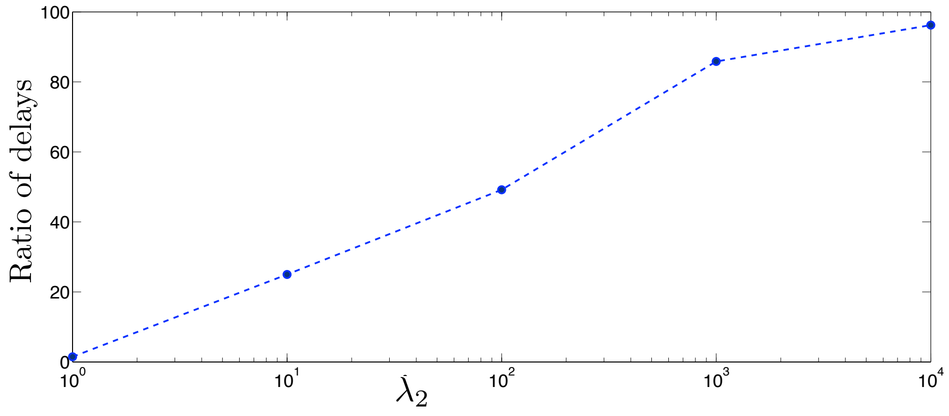


Figure 5.7: Ratio of experimental delays between Complete Merge policy and SQ policy as a function of λ_2, with $m = 2$, $\lambda_1 = 1$, c $= 0.995$ and $\varrho = 0.9$.

5.4.4 The Complete Merge Policy

As described in Section 5.3.3, a naive policy for our problem is to ignore priorities, merge all classes into a single class, and repeatedly form TSP tours through all outstanding demands. We call this policy the Complete Merge (CM) policy. In this section we briefly verify that the performance of the Complete Merge policy can be arbitrarily bad when compared to the SQ policy. In addition, the conditions for which the Complete Merge policy performs poorly are precisely the conditions of interest for most applications; when low priority demands arrive much more frequently than high priority demands. To see this, define the total arrival rate $\Lambda := \sum_{\alpha=1}^{m} \lambda_\alpha$ and total mean on-site service $\bar{S} := \sum_{\alpha=1}^{m} \bar{s}_\alpha$. Using the upper bounds in (Bertsimas and van Ryzin, 1991), we immediately obtain that $D(\text{CM}) \leq \frac{\beta_{\text{TSP}}^2 |\mathcal{E}| \Lambda}{n^2 v^2 (1-\varrho)^2}$. Thus, the ratio $D(\text{CM})/D(\text{SQ})$ can be made arbitrarily large by choosing $\lambda_1 \ll \lambda_2 \ll \cdots \ll \lambda_m$ and $c_1 \gg c_2 \gg \cdots \gg c_m$. Fig. 5.7 shows the experimentally obtained ratio between the delay of the Complete Merge policy and that of the SQ policy (averaged over 10 simulation runs), and verifies the poor performance of the CM policy.

5.5 Summary

In this chapter we introduced a dynamic vehicle routing problem with priority classes. We captured the priority levels of classes by writing the system delay as a convex combination of the delay of each class. We determined a lower bound on the achievable values of the convex combination of the class delays. We then presented the Separate Queues (SQ) policy and showed that it performs within a constant factor of the lower bound, which depends only on the number of the classes. We believe that it may be possible to improve the lower bound and remove, or reduce, the constant factor's dependence on the number of classes.

Chapter Six

Translating Demands

In this chapter, we introduce a dynamic vehicle routing problem in which the demands move with a specified velocity upon arrival, and we design a novel receding horizon control policy for a single vehicle to service them. This problem has applications in areas such as surveillance and perimeter defense, wherein the demands could be visualized as moving targets trying to cross a region under surveillance by an Unmanned Air Vehicle. Another application is in the automation industry where the demands are objects that arrive continuously on a conveyor belt and a robotic arm performs a pick-and-place operation on them (Chalasani and Motwani, 1999). Our problem also has similarities to the RoboFlag game (D'Andrea and Murray, 2003), which consists of two teams of robots, with each team attempting to capture the other team's flag while defending its own.

In addition to the dynamic vehicle routing literature reviewed in Chapter 1, the work in this chapter is related to geometric location problems such as (Megiddo and Supowit, 1984), and (Zemel, 1984), where given a set of static demands, the goal is to find supply locations that minimize a cost function of the distance from each demand to its nearest supply location. In our problem, in the asymptotic regime of low arrival, when the arrival rate λ tends to zero for a fixed demand speed v, the problem becomes one of providing optimal coverage. In this regime, the demands are served in a first-come-first-served order; such policies are common in classical queuing theory (see Chapter 2). Another related work is by Martínez et al. (2007), who study the problem of deploying robots into a region so as to provide optimal coverage. Other relevant literature include (Tang and Özgüner, 2005) in which multiple mobile targets are to be kept under surveillance by multiple mobile sensor agents. Kingston and Schumacher (2005) propose mixed-integer-linear-program approach to assigning multiple agents to time-dependent cooperative tasks such as tracking mobile targets.

The contributions of this chapter are as follows. We introduce a dynamic vehicle routing problem in which demands arrive via a stochastic process on a line segment of fixed length, and upon arrival, translate with a fixed velocity perpendicular to the segment. A service vehicle, modeled as a first-order integrator having speed greater than that of the demands, seeks to serve these mobile demands. We analyze the problem when the demands are uniformly distributed along the segment and the demand arrival process is Poisson with rate λ. For a fixed length W of the segment and the vehicle speed normalized to unity, the problem is governed by two parameters; the demand speed v and the arrival rate λ. Our results are as follows. First, we derive a necessary condition on λ in terms of v for the existence of a stable service policy. Second, we analyze our novel TMHP-BASED policy and derive a sufficient condition for λ in terms of v that ensures stability of the policy. With respect to stability of the problem, we identify two asymptotic regimes: (a) *High speed regime*: when the demands move as fast as the vehicle, i.e., $v \to 1^-$ (and therefore for stability, $\lambda \to 0^+$); and (b) *Low speed regime*: when demand speed tends to zero, i.e., $v \to 0^+$ (and so a sufficiently high λ may still ensure stability). In the high speed regime, we show that: (i) for existence of a stabilizing policy, λ must converge to zero as $1/\sqrt{-\log(1-v)}$, (ii) every stabilizing policy must service the demands in the first-come-first-served (FCFS) order, and (iii) of all such policies, the TMHP-BASED policy minimizes the expected time to service a demand. In the low speed regime, we show that the sufficient condition on λ for the stability of the TMHP-BASED policy is within a constant factor of the necessary condition on λ for stability of any policy. Third, we identify another asymptotic regime, termed as the *low arrival regime*, in which the arrival rate $\lambda \to 0^+$ for a fixed demand speed. In this low arrival regime, we establish that the TMHP-BASED policy is optimal in terms of minimizing the expected time to service a demand. Fourth, for the analysis of the TMHP-BASED policy, we study the classic FCFS policy in which demands are served in the order in which they arrive. We determine necessary and sufficient conditions on λ for the stability of the FCFS policy. Fifth and finally, we validate our analysis with extensive simulations and provide an empirically accurate characterization of the region in the parameter space of demand speed and arrival rate for which the TMHP-BASED policy is stable.

A plot of the theoretically established necessary and sufficient conditions for stability in the v-λ parameter space is shown in Figure 6.1. The bottom figures are for the asymptotic regimes of $v \to 0^+$, and $v \to 1^-$, respectively.

This chapter is organized as follows. The problem formulation, the TMHP-BASED policy, and the main results are presented in Section 6.1. The FCFS policy is presented and analyzed in Section 6.2. Utilizing the results of Section 6.2, the main results are proven in Section 6.3. Finally, simulation results are presented in Section 6.4.

The work in this chapter was performed in collaboration with Shaunak D. Bopardikar. For consistency, this thesis contains all results. However, the TMHP-based policy was primarily the work of S. D. Bopardikar in collaboration with S. L. Smith, while the necessary stability conditions, and the analysis of the

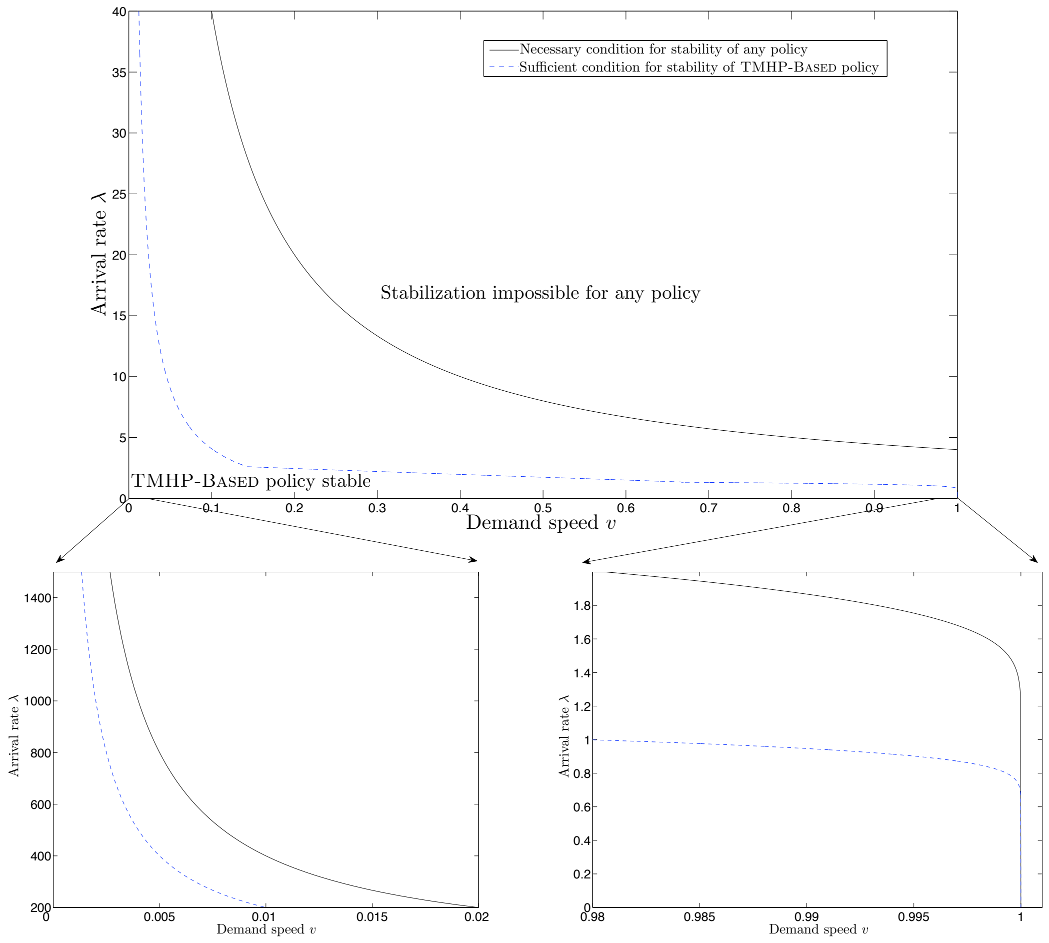


Figure 6.1: A summary of stability regions for the TMHP-Based policy. Stable service policies exist only under the solid black curve. In the top figure, the solid black curve is due to part (i) of Theorem 6.1 and the dashed blue curve is due to part (i) of Theorem 6.2. In the asymptotic regime shown in the bottom left, the dashed blue curve is described in part (ii) of Theorem 6.2, and is different than the one in the top figure. In the asymptotic regime shown in the bottom right, the solid black curve is due to part (ii) of Theorem 6.1, and is different from the solid black curve in the top figure.

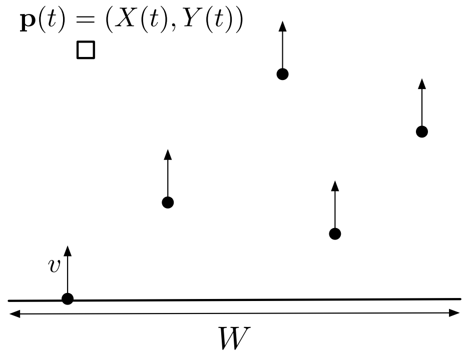


Figure 6.2: The problem setup. The thick line segment is the generator of mobile demands. The dark circle denotes a demand and the square denotes the service vehicle.

FCFS policy was primarily the work of S. L. Smith in collaboration with S. D. Bopardikar.

6.1 Problem Formulation and the TMHP-Based policy

In this section, we pose the dynamic vehicle routing problem with translating demands and present the TMHP-Based policy along with the main results.

6.1.1 Problem Statement

We consider a single service vehicle that seeks to service mobile demands that arrive via a spatio-temporal process on a line segment with length W along the x-axis, termed the *generator*. The vehicle is modeled as a first-order integrator with speed upper bounded by one. The demands arrive uniformly distributed on the generator via a temporal Poisson process with intensity $\lambda > 0$, and move with constant speed $v \in]0, 1[$ along the positive y-axis, as shown in Figure 6.2. We assume that once the vehicle reaches a demand, the demand is served instantaneously. The vehicle is assumed to have unlimited fuel and demand servicing capacity.

We define the environment as $\mathcal{E} := [0, W] \times \mathbb{R}_{\geq 0} \subset \mathbb{R}^2$, and let $\mathbf{p}(t) = [X(t), Y(t)]^T \in \mathcal{E}$ denote the position of the service vehicle at time t. Let $\mathcal{Q}(t) \subset \mathcal{E}$ denote the set of all demand locations at time t, and $N(t)$ the cardinality of $\mathcal{Q}(t)$. Servicing of a demand $\mathbf{q}_i \in \mathcal{Q}$ and removing it from the set $\mathcal{Q}$ occurs when the service vehicle reaches the location of the demand. Recall that a policy for the vehicle is a map $P : \mathcal{E} \times \text{FIN}(\mathcal{E}) \to \mathbb{R}^2$, where $\text{FIN}(\mathcal{E})$ is the set of finite subsets of $\mathcal{E}$, assigning a velocity to the service vehicle as a function of the current state of the system: $\dot{\mathbf{p}}(t) = P(\mathbf{p}(t), \mathcal{Q}(t))$. Let $D(i)$ denote the delay between the generation of the ith demand and the time it is serviced. Recall that a policy P is *stable* if under its action, the steady-state expected delay $D(P)$, or equivalently the steady-state $\bar{N}(P)$ is finite. That is, if the vehicle is able to service demands at a rate that is—on average—at least as fast as the rate at which new demands arrive. In what follows, our goal is to *design stable control policies* for the system.

To obtain further intuition into stability of a policy, consider the v-λ parameter space. In the asymptotic regime of high speed, where $v \to 1^-$, the arrival rate λ *must* tend to zero for stability, otherwise the service vehicle would have to move successively further away from the generator in expected value, thus making the system unstable. In the asymptotic region of low demand speed, where $v \to 0^+$, if $\lambda \to +\infty$, then we expect the system to be unstable; while for a sufficiently low λ, we expect to be able to stabilize the system. Thus, our goal is to characterize regions in the v-λ parameter space in which one can never design *any* stable policy, as well as those in which our policies are stable, with additional emphasis in the above two asymptotic regimes. In addition, for the asymptotic regime of low arrival, where for a fixed speed $v < 1$, the arrival rate $\lambda \to 0^+$, stability is intuitive as demands arrive very rarely. Hence, in this regime, we seek to minimize the steady state expected delay for a demand.

6.1.2 The TMHP-Based policy

We now present a novel receding horizon service policy for the vehicle that is based on the repeated computation of a translational minimum Hamiltonian path through successive groups of outstanding demands. For a given arrival rate λ and demand speed $v \in]0, 1[$, let (X^*, Y^*) denote the vehicle location in the environment that minimizes the expected time to service a demand once it appears on the generator. The expression for the optimal location (X^*, Y^*) is postponed to Section 6.2.1. The TMHP-based policy is summarized in Algorithm 6.1, and an iteration of the policy is illustrated in Figure 6.3.

Algorithm 6.1: TMHP-Based policy

```
  Assumes: The optimal location $(X^*, Y^*) \in \mathcal{E}$ is given.
1 if no outstanding demands are present in $\mathcal{E}$ then
2     Move to the optimal position $(X^*, Y^*)$.
3 else
4     Service all outstanding demands by following a translational minimum
      Hamiltonian path starting from the vehicle's current location, and
      terminating at the demand with the lowest $y$-coordinate.
5 Repeat.
```

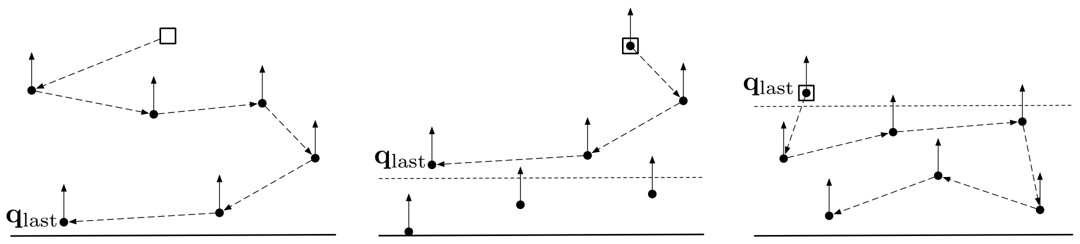

$\mathbf{q}_{\text{last}}$
$\mathbf{q}_{\text{last}}$
$\mathbf{q}_{\text{last}}$

Figure 6.3: An iteration of the TMHP-Based policy. The vehicle (square) serves all outstanding demands (black dots) as per the TMHP beginning at (X, Y) and terminating at $\mathbf{q}_{\text{last}}$. The left-figure shows a TMHP at the beginning of an iteration. The middle-figure shows the vehicle midway through servicing the demands along the TMHP; new demands have arrived in the environment. The right-figure shows the vehicle reaching $\mathbf{q}_{\text{last}}$ and commencing the next iteration by computing a TMHP through all outstanding demands.

6.1.3 Main Results

The following is a summary of our main results and the locations of their proofs within the chapter. We begin with a necessary condition for stability, the proof of which is presented in Section 6.3.1.

Theorem 6.1 (Necessary condition for stability). *The following are necessary conditions for the existence of a stabilizing policy:*

(i) For $v \in \,]0,1[$,

$$\lambda \leq \frac{4}{vW}.$$

(ii) For the asymptotic regime of high speed, where $v \to 1^-$, every stabilizing policy must serve the demands in the order in which they arrive and hence,

$$\lambda \leq \frac{3\sqrt{2}}{W\sqrt{-\log(1-v)}}.$$

Then, we derive a sufficient condition for stability of the TMHP-Based policy, the proof of which is presented in Section 6.3.2. We introduce the following notation. Let

$$\lambda_{\text{FCFS}}(v, W) := \begin{cases} \dfrac{3}{W}\sqrt{\dfrac{1-v}{1+v}}, & \text{for } v \leq v^*_{\text{suf}}, \\ \dfrac{\sqrt{12v}}{W\sqrt{(1+v)\left(C_{\text{suf}} - \log\left(\frac{1-v}{v}\right)\right)}}, & \text{otherwise,} \end{cases}$$

where $C_{\text{suf}} = \pi/2 - \log(0.5 \cdot \sqrt{3}/\sqrt{2})$, and v^*_{suf} is the solution to

$$\sqrt{12v^*} - 3\sqrt{(1-v^*)(C_{\text{suf}} - \log(1-v^*) + \log v^*)} = 0,$$

and is approximately equal to $2/3$.

Theorem 6.2 (Sufficient condition for stability)**.** *The following are sufficient conditions for stability of the* TMHP-BASED *policy.*

(i) For $v \in]0,1[$,

$$\lambda < \max\left\{\frac{(1-v^2)^{3/2}}{2vW(1+v)^2}, \lambda_{\text{FCFS}}(v,W)\right\}.$$

(ii) For the asymptotic regime of low speed where $v \to 0^+$,

$$\lambda < \frac{1}{\beta^2_{\text{TSP}} vW}, \qquad \textit{where } \beta_{\text{TSP}} \approx 0.7120.$$

A plot of the necessary and sufficient conditions is shown in Fig. 6.1. In the asymptotic regime of high speed, the sufficient condition from part (i) of Theorem 6.2 simplifies to

$$\lambda < \frac{\sqrt{6}}{W\sqrt{-\log(1-v)}} =: \lambda^{1^-}_{\text{suf}},$$

and the necessary condition established in part (ii) of Theorem 6.1 simplifies to

$$\lambda \le \frac{3\sqrt{2}}{W\sqrt{-\log(1-v)}} =: \lambda^{1^-}_{\text{nec}}.$$

In the asymptotic regime of low speed, the sufficient condition from part (ii) of Theorem 6.2 is $\lambda < 1/(\beta^2_{\text{TSP}} vW) =: \lambda^{0^+}_{\text{suf}}$, and the necessary condition established in part (i) of Theorem 6.1 is $\lambda \le 4/(vW) =: \lambda^{0^+}_{\text{nec}}$.

Theorems 6.1 and 6.2 immediately lead to the following corollary.

Corollary 6.3 (Constant factor sufficient condition)**.** *In the asymptotic regime of*

(i) high speed, which is the limit as $v \to 1^-$, the ratio $\lambda^{1^-}_{\text{nec}}/\lambda^{1^-}_{\text{suf}} \to \sqrt{3}$.

(ii) low speed, which is the limit as $v \to 0^+$, the ratio $\lambda^{0^+}_{\text{nec}}/\lambda^{0^+}_{\text{suf}} \to 4\beta^2_{\text{TSP}} \approx 2.027$.

Finally, we state the following optimality property of the TMHP-BASED policy, the proof of which is presented in Section 6.3.2.

Theorem 6.4 (Optimality of TMHP-BASED policy)**.** *The* TMHP-BASED *policy minimizes the expected time to service a demand in*

(i) the low arrival asymptotic regime, where $\lambda \to 0^+$ for a fixed $v \in]0, 1[$, and

(ii) the high speed asymptotic regime, where $v \to 1^-$ (and therefore $\lambda \to 0^+$).

In order to study the stability of the TMHP-BASED policy, we introduce and analyze a first-come-first-served (FCFS) policy in the next section.

6.2 The First-Come-First-Served (FCFS) Policy and its Analysis

In this section, we present the FCFS policy and establish some of its properties. To state the FCFS policy we review the constant bearing control motion from Section 2.2.3.

Definition 6.5 (Constant bearing control)**.** *Given initial locations* $\mathbf{p} := (X, Y) \in \mathbb{R}^2$ *and* $\mathbf{q} := (x, y) \in \mathbb{R}^2$ *of the service vehicle and a demand, respectively, with the demand moving in the positive y-direction with constant speed $v \in]0, 1[$, the motion of the service vehicle towards the point $(x, y + vT)$, where*

$$T(\mathbf{p}, \mathbf{q}) := \frac{\sqrt{(1 - v^2)(X - x)^2 + (Y - y)^2}}{1 - v^2} - \frac{v(Y - y)}{1 - v^2}, \tag{6.1}$$

with unit speed is defined as the constant bearing control.

Constant bearing control is illustrated in Figure 6.4 and characterized in the following proposition.

Proposition 6.6 (Minimum time control, Isaacs (1965))**.** *The constant bearing control is the minimum time control for the service vehicle to reach the moving demand.*

In the FCFS policy, the service vehicle uses constant bearing control and services the demands in the order in which they arrive. If the environment contains no demands, the vehicle moves to the location (X^*, Y^*) which minimizes the expected time to catch the next demand to arrive. This policy is summarized in Algorithm 6.2. Figure 6.5 illustrates an instance of the FCFS policy. The following lemma establishes the relationship between the FCFS policy and the TMHP-BASED policy.

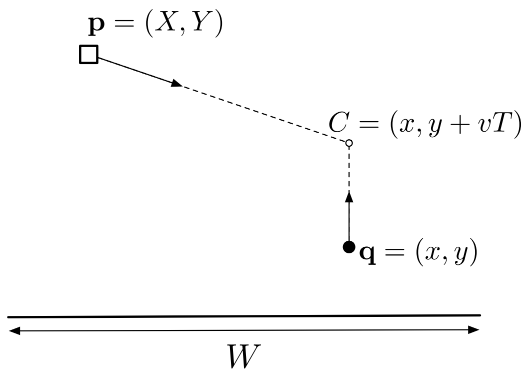


Figure 6.4: Constant bearing control. The vehicle moves towards the point $C := (x, y + vT)$, where x, y, v and T are as per Definition 6.5, to reach the demand.

Algorithm 6.2: FCFS policy

```
  Assumes: The optimal location $(X^*, Y^*) \in \mathcal{E}$ is given.
1 if no outstanding demands are present in $\mathcal{E}$ then
2     Move toward $(X^*, Y^*)$ until the next demand arrives.
3 else
4     Move using the constant bearing control to service the furthest demand
      from the generator.
5 Repeat.
```

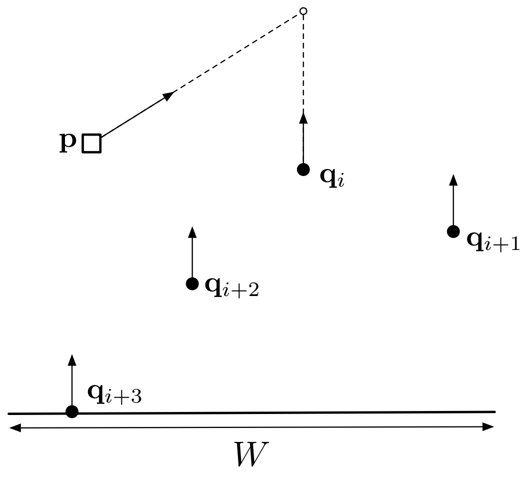


Figure 6.5: The FCFS policy. The vehicle services the demands in the order of their arrival in the environment, using the constant bearing control.

Lemma 6.7 (Relationship between TMHP-Based policy and FCFS policy). *Given an arrival rate λ and a demand speed v, if the FCFS policy is stable, then the* TMHP-Based *policy is stable.*

Proof. Consider an initial vehicle position and a set of outstanding demands, all of which have lower y-coordinates than the vehicle. Let us compare the amount of time required to service the outstanding demands using the TMHP-Based policy with the amount of time required for the FCFS policy. Both policies generate paths through all outstanding demands, starting at the initial vehicle location, and terminating at the outstanding demand with the lowest y-coordinate. By definition, the TMHP-Based policy generates the shortest such path. Thus, the TMHP-Based policy will require no more time to service all outstanding demands than the FCFS policy. Since this holds at every iteration of the policy, the region of stability of TMHP-Based policy contains the region of stability for the FCFS policy. $\square$

In the following subsections, we analyze the FCFS policy. We then combine these results with the above lemma to establish analogous results for the TMHP-Based policy.

The first question is, how do we compute the optimal position (X^*, Y^*)? This will be answered in the following subsection.

6.2.1 Optimal Vehicle Placement

In this subsection, we study the FCFS policy when $v \in \,]0,1[$ is fixed and $\lambda \to 0^+$. In this regime, stability is not an issue as demands arrive very rarely, and the problem becomes one of optimally placing the service vehicle (i.e., determining (X^*, Y^*) in the statement of the FCFS policy).

We seek to place the vehicle at location that minimizes the expected time to service a demand once it appears on the generator. Demands appear at uniformly random positions on the generator and the vehicle uses the constant bearing control to reach the demand. Thus, the expected time to reach a demand generated at position $\mathbf{q} = (x, 0)$ from vehicle position $\mathbf{p} = (X, Y)$ is given by

$$\mathbb{E}\left[T(\mathbf{p}, \mathbf{q})\right] = \frac{1}{W(1-v^2)} \int_0^W \left(\sqrt{(1-v^2)(X-x)^2 + Y^2} - vY\right) dx. \tag{6.2}$$

The following lemma characterizes the way in which this expectation varies with the position $\mathbf{p}$.

Lemma 6.8 (Properties of the expected time). *The expected time* $\mathbf{p} \mapsto \mathbb{E}\left[T(\mathbf{p}, \mathbf{q})\right]$ *is convex in* $\mathbf{p}$*, for all* $\mathbf{p} \in [0, W] \times \mathbb{R}_{>0}$*. Additionally, there exists a unique point*

$\mathbf{p}^* := (W/2, Y^*) \in \mathbb{R}^2$ *that minimizes* $\mathbf{p} \mapsto \mathbb{E}\left[T(\mathbf{p}, \mathbf{q})\right]$.

Proof. Regarding the first statement, it suffices to show that the integrand in equation (6.2), $T(\mathbf{p}, (x, 0))$ is convex for all x. To do this we compute the Hessian of $T((X, Y), (x, 0))$ with respect to X and Y. Thus, for $Y > 0$,

$$\begin{bmatrix} \frac{\partial^2 T}{\partial X^2} & \frac{\partial^2 T}{\partial X \partial Y} \\ \frac{\partial^2 T}{\partial Y \partial X} & \frac{\partial^2 T}{\partial Y^2} \end{bmatrix} = \frac{1}{\Big((1 - v^2)(X - x)^2 + Y^2\Big)^{3/2}} \begin{bmatrix} Y^2 & Y(X - x) \\ Y(X - x) & (X - x)^2 \end{bmatrix}.$$

The Hessian is positive semi-definite because its determinant is zero and its trace is non-negative. This implies that $T(\mathbf{p}, \mathbf{q})$ is convex in $\mathbf{p}$ for each $\mathbf{q} = (x, 0)$, from which the first statement follows.

Regarding the second statement, since demands are uniformly randomly generated on the interval $[0, W]$, the optimal horizontal position is $X^* = W/2$. Thus, it suffices to show that $Y \mapsto \mathbb{E}\left[T((W/2, Y), \mathbf{q})\right]$ is strictly convex. From the $\partial^2 T/\partial Y^2$ term of the Hessian we see that $T(\mathbf{p}, \mathbf{q})$ is strictly convex for all $x \neq W/2$. But, letting $\mathbf{p} = (W/2, Y)$ and $\mathbf{q} = (x, 0)$ we can write

$$\mathbb{E}\left[T(\mathbf{p}, \mathbf{q})\right] = \frac{1}{W(1 - v^2)} \int_{x \in [0, W] \setminus \{W/2\}} T(\mathbf{p}, \mathbf{q}) dx.$$

The integrand is strictly convex for all $x \in [0, W] \setminus \{W/2\}$, implying that $\mathbb{E}\left[T(\mathbf{p}, \mathbf{q})\right]$ is strictly convex on the line $X = W/2$, and that the point $(W/2, Y^*)$ is the unique minimizer. □

Lemma 6.8 tells us that there exists a unique point $\mathbf{p}^* := (X^*, Y^*)$ which minimizes the expected travel time. In addition, we know that $X^* = W/2$. Obtaining a closed form expression for Y^* does not appear to be possible. Computing the integral in equation (6.2), with $X = W/2$, one can obtain

$$\mathbb{E}\left[T(\mathbf{p}, \mathbf{q})\right] = \frac{Y}{a}\left(\frac{1}{2}\sqrt{1 + \frac{aW^2}{4Y^2}} - \frac{Y}{\sqrt{a}W}\log\left(\sqrt{1 + \frac{aW^2}{4Y^2}} - \sqrt{\frac{aW^2}{4Y^2}}\right) - v\right),$$

where $a = 1 - v^2$. For each value of v and W, this convex expression can be easily numerically minimized over Y, to obtain Y^*. A plot of Y^* as a function of v for $W = 1$ is shown in Figure 6.6.

For the optimal position $\mathbf{p}^*$, the expected delay between a demand's arrival and its service completion is

$$D^* := \mathbb{E}\left[T(\mathbf{p}^*, (x, 0))\right].$$

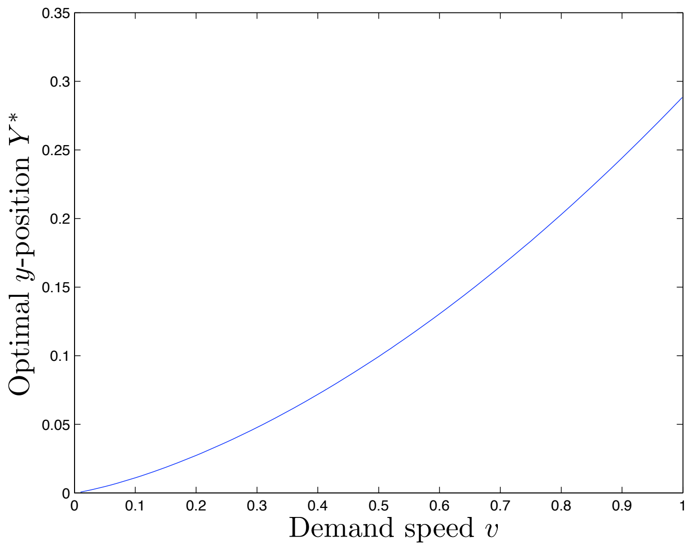


Figure 6.6: The optimal position Y^* of the service vehicle which minimizes the expected distance to a demand, as a function of v. In this plot, the generator has length $W = 1$.

Thus, a lower bound on the steady-state expected delay of any policy is D^*. We now characterize the steady-state expected delay of the FCFS policy D_{FCFS}, as λ tends to zero.

Lemma 6.9 (FCFS optimality)**.** *Fix any $v \in]0, 1[$. Then in the limit as $\lambda \to 0^+$, the FCFS policy minimizes the expected time to service a demand, i.e., $D_{\text{FCFS}} \to D^*$.*

Proof. We have shown how to compute the position $\mathbf{p}^* := (X^*, Y^*)$ which minimizes equation (6.2). Thus, if the vehicle is located at $\mathbf{p}^*$, then the expected time to service the demand is minimized. But, as $\lambda \to 0^+$, the probability that demand $i+1$ arrives before the vehicle completes service of demand i and returns to $\mathbf{p}^*$ tends to zero. Thus, the FCFS policy is optimal as $\lambda \to 0^+$. $\square$

Remark 6.10 (Minimizing the worst-case time)**.** *Another metric that can be used to determine the optimal placement (X^*, Y^*) is the worst-case time to service a demand. Using an argument identical to that in the proof of Lemma 6.9, it is possible to show that for fixed $v \in]0, 1[$, and as $\lambda \to 0^+$, the FCFS policy, with $(X^*, Y^*) = (W/2, vW/2)$, minimizes the worst-case time to service a demand.* •

6.2.2 A Necessary Condition for FCFS Stability

In the previous subsection, we studied the case of fixed v and $\lambda \to 0^+$. In this subsection, we analyze the problem when $\lambda > 0$, and determine necessary conditions on the magnitude of λ that ensure the FCFS policy remains stable. To establish these conditions we utilize a standard result in queueing theory (see Chapter 2) which states that a necessary condition for the existence of a stabilizing policy is that $\lambda \mathbb{E}[T] \leq 1$, where $\mathbb{E}[T]$ is the expected time to service a demand (i.e., the travel time between demands). We begin with the following result.

Proposition 6.11 (Special case of equal speeds)**.** *For $v = 1$, there does not exist a stabilizing policy.*

Proof. When $v = 1$, each demand and the service vehicle move at the same speed. If a demand has a higher vertical position than the service vehicle, then clearly the service vehicle cannot reach it. The same impossibility result holds if the demand has the same vertical position and a distinct horizontal position as the service vehicle. In summary, a demand can be reached only if the service vehicle is above the demand. Next, note that the only policy that ensures that a demand's y-coordinate never exceeds that of the service vehicle (i.e., that all demands remain below the service vehicle at all time) is the FCFS policy. In what follows, we prove the proposition statement by computing the expected time to travel between demands using the FCFS policy. First, consider a vehicle location $\mathbf{p} := (X, Y)$ and a demand location with initial location $\mathbf{q} := (x, y)$, the minimum time T in which the vehicle can reach the demand is given by

$$T(\mathbf{p}, \mathbf{q}) = \frac{(X - x)^2 + (Y - y)^2}{2(Y - y)}, \quad \text{if } Y > y, \tag{6.3}$$

and is undefined if $Y \leq y$. Second, assume there are many outstanding demands below the service vehicle, and none above. Suppose the service vehicle completed the service of demand i at time t_i and position $(x_i(t_i), y_i(t_i))$. Let us compute the expected time to reach demand $i + 1$, with location $(x_{i+1}(t_i), y_{i+1}(t_i))$. Since arrivals are Poisson it follows that $y_i(t_i) > y_{i+1}(t_i)$. To simplify notation we define $\Delta x = |x_i(t_i) - x_{i+1}(t_i)|$ and $\Delta y = y_i(t_i) - y_{i+1}(t_i)$. Then, from equation (6.3)

$$T(\mathbf{q}_i, \mathbf{q}_{i+1}) = \frac{\Delta x^2 + \Delta y^2}{2\Delta y} = \frac{1}{2}\left(\frac{\Delta x^2}{\Delta y} + \Delta y\right).$$

Taking expectation and noting that Δx and Δy are independent,

$$\mathbb{E}[T(\mathbf{q}_i, \mathbf{q}_{i+1})] = \frac{1}{2}\left(\mathbb{E}[\Delta x^2]\mathbb{E}\left[\tfrac{1}{\Delta y}\right] + \mathbb{E}[\Delta y]\right).$$

Now, we note that $\mathbb{E}[\Delta y] = 1/\lambda$, that $\mathbb{E}[\Delta x^2]$ is a positive constant independent of λ, and that

$$\mathbb{E}\left[\tfrac{1}{\Delta y}\right] = \int_{y=0}^{+\infty} \frac{1}{y} \lambda e^{-\lambda y} dy = +\infty.$$

Thus $\mathbb{E}[T(\mathbf{q}_i, \mathbf{q}_{i+1})] = +\infty$, and for every $\lambda > 0$,

$$\lambda \mathbb{E}[T(\mathbf{q}_i, \mathbf{q}_{i+1})] = +\infty.$$

This means that the necessary condition for stability, i.e., $\lambda \mathbb{E}[T(\mathbf{q}_i, \mathbf{q}_{i+1})] \leq 1$, is violated. Thus, there does not exist a stabilizing policy. □

Next we look at the FCFS policy and give a necessary condition for its stability.

Lemma 6.12 (Necessary stability condition for FCFS)**.** *A necessary condition for the stability of the FCFS policy is*

$$\lambda \leq \begin{cases} \dfrac{3}{W}, & \text{for } v \leq v^*_{\text{nec}}, \\ \dfrac{3\sqrt{2v}}{W\sqrt{(1+v)\left(C_{\text{nec}} - \log\left(\frac{\sqrt{1-v^2}}{v}\right)\right)}}, & \text{otherwise,} \end{cases}$$

where $C_{\text{nec}} = 0.5 + \log(2) - \gamma$, where γ is the Euler constant; and v^*_{nec} is the solution to

$$2v - (1+v)(C_{\text{nec}} - 0.5 \cdot \log(1-v^2) + \log v) = 0,$$

and is approximately equal to $4/5$.

Proof. Suppose the service vehicle completed the service of demand i at time t_i at position $(x_i(t_i), y_i(t_i))$, and demand $i+1$ is located at $(x_{i+1}(t_i), y_{i+1}(t_i))$. Define $\Delta x := |x_i(t_i) - x_{i+1}(t_i)|$ and $\Delta y := y_i(t_i) - y_{i+1}(t_i)$. For $v \in]0, 1[$, the travel time between demands is given by

$$T = \frac{1}{1-v^2}\left(\sqrt{(1-v^2)\Delta x^2 + \Delta y^2} - v\Delta y\right). \tag{6.4}$$

Observe that the function T is convex in Δx and Δy. Jensen's inequality leads to

$$\mathbb{E}[T] \geq \frac{1}{1-v^2}\left(\sqrt{(1-v^2)(\mathbb{E}[\Delta x])^2 + (\mathbb{E}[\Delta y])^2} - v\mathbb{E}[\Delta y]\right).$$

Substituting the expressions for the expected values, we obtain

$$\mathbb{E}\left[T\right] \geq \frac{1}{1-v^2}\left(\sqrt{(1-v^2)\frac{W^2}{9}+\frac{v^2}{\lambda^2}}-\frac{v^2}{\lambda}\right).$$

From the necessary condition for stability, we must have

$$\lambda\mathbb{E}\left[T\right] \leq 1 \qquad \Longleftrightarrow \qquad \lambda\frac{1}{1-v^2}\left(\sqrt{\frac{(1-v^2)W^2}{9}+\frac{v^2}{\lambda^2}}-\frac{v^2}{\lambda}\right) \leq 1.$$

By simplifying, we obtain

$$\lambda \leq \frac{3}{W}. \tag{6.5}$$

This provides a good necessary condition for low v. Next, we obtain a much better necessary condition for large v.

Since T is convex in Δx, we apply Jensen's inequality to equation (6.4) to obtain

$$\mathbb{E}\left[T|\Delta y\right] \geq \frac{1}{1-v^2}\left(\sqrt{(1-v^2)W^2/9+\Delta y^2}-v\Delta y\right), \tag{6.6}$$

where $\mathbb{E}\left[\Delta x\right] = W/3$. Now, the random variable Δy is distributed exponentially with parameter λ/v and probability density function

$$f(y) = \frac{\lambda}{v}\mathrm{e}^{-\lambda y/v}.$$

Un-conditioning equation (6.6) on Δy, we obtain

$$\mathbb{E}\left[T\right] = \int_0^{+\infty}\mathbb{E}\left[T|y\right]f(y)dy \geq \frac{\lambda}{v(1-v^2)}\int_0^{+\infty}\left(\sqrt{\frac{(1-v^2)W^2}{9}+y^2}-vy\right)\mathrm{e}^{-\lambda y/v}dy. \tag{6.7}$$

The right hand side can be evaluated using the software Maple® and equals

$$\frac{\pi W}{2\cdot 3\sqrt{1-v^2}}\left[\mathbf{H}_1\left(\frac{\lambda W\sqrt{1-v^2}}{3v}\right)-\mathbf{Y}_1\left(\frac{\lambda W\sqrt{1-v^2}}{3v}\right)\right]-\frac{v^2}{\lambda(1-v^2)},$$

where $\mathbf{H}_1 : \mathbb{R} \to \mathbb{R}$ is the 1st order Struve function and $\mathbf{Y}_1 : \mathbb{R} \to \mathbb{R}$ is 1st order Bessel function of the second kind (Newman, 1984).Using a Taylor series expansion of the function $\mathbf{H}_1(z)-\mathbf{Y}_1(z)$ about $z=0$, followed by a subsequent analysis of the higher order terms, one can show that

$$\mathbf{H}_1(z)-\mathbf{Y}_1(z) \geq \frac{1}{\pi}\left(\frac{2}{z}+C_{\text{nec}}z-z\log(z)\right), \quad \forall z \geq 0,$$

where $C_{\text{nec}} = 1/2 + \log(2) - \gamma$, and γ is the Euler constant. This inequality implies that equation (6.7) can be written as

$$\mathbb{E}\left[T\right] \geq \frac{v}{\lambda(1+v)} + \frac{\lambda W}{18v}\left(C_{\text{nec}} - \log\left(\frac{\lambda W\sqrt{1-v^2}}{3v}\right)\right),$$

where we have used the fact that

$$\frac{v}{\lambda(1-v)^2} - \frac{v^2}{\lambda(1-v^2)} = \frac{v}{\lambda(1+v)}.$$

To obtain a stability condition on λ we wish to remove λ from the log term. To do this, note that from equation (6.5) we have $\lambda W/3 < 1$, and thus

$$\begin{aligned}\mathbb{E}\left[T\right] &\geq \frac{v}{\lambda(1+v)} + \frac{\lambda W}{18v}\left(C_{\text{nec}} - \log\frac{W\lambda}{3} - \log\frac{W\sqrt{1-v^2}}{3v}\right)\\ &\geq \frac{v}{\lambda(1+v)} + \frac{\lambda W}{18v}\left(C_{\text{nec}} - \log\left(\frac{\sqrt{1-v^2}}{v}\right)\right).\end{aligned}$$

The necessary stability condition is $\lambda\mathbb{E}\left[T\right] \leq 1$, from which a necessary condition for stability is

$$\frac{\lambda^2 W}{18v}\left(C_{\text{nec}} - \log\left(\frac{\sqrt{1-v^2}}{v}\right)\right) \leq 1 - \frac{v}{1+v} = \frac{1}{1+v}.$$

Solving for λ when $\log(\sqrt{1-v^2}/v) < C_{\text{nec}}$, we obtain that

$$\lambda \leq \frac{3\sqrt{2v}}{W\sqrt{(1+v)\left(C_{\text{nec}} - \log\left(\frac{\sqrt{1-v^2}}{v}\right)\right)}}, \tag{6.8}$$

The condition $C_{\text{nec}} > \log(\sqrt{1-v^2}/v)$, implies that the above bound holds for all $v > 1/\sqrt{1+\mathrm{e}^{2C_{\text{nec}}}}$. We now have two bounds; equation (6.5) which holds for all $v \in]0,1[$, and equation (6.8) which holds for $v > 1/2$. The final step is to determine the values of v for which each bound is active. To do this, we set the right-hand side of equation (6.5) equal to the right-hand side of equation (6.8) and denote the solution by v^*_{nec}. Thus, the necessary condition for stability is given by equation (6.5) when $v \leq v^*_{\text{nec}}$, and by equation (6.8) when $v > v^*_{\text{nec}}$. $\square$

6.2.3 A Sufficient Condition for FCFS stability

In Section 6.2.2, we determined a necessary condition for stability of the FCFS policy. In this subsection, we will derive the following sufficient condition on the arrival rate that ensures stability for the FCFS policy. To establish this condition, we apply the sufficient condition for the existence of a stabilizing policy, which is that $\lambda\mathbb{E}[T] < 1$, where $\mathbb{E}[T]$ is the expected time to service a demand (i.e., the travel time between demands).

Lemma 6.13 (Sufficient stability condition for FCFS)**.** *The FCFS policy is stable if*

$$\lambda < \begin{cases} \dfrac{3}{W}\sqrt{\dfrac{1-v}{1+v}}, & \textit{for } v \le v^*_{\text{suf}}, \\ \dfrac{\sqrt{12v}}{W\sqrt{(1+v)\left(C_{\text{suf}} - \log\left(\frac{1-v}{v}\right)\right)}}, & \textit{otherwise,} \end{cases}$$

where $C_{\text{suf}} = \pi/2 - \log(0.5\cdot\sqrt{3}/\sqrt{2})$, and v^*_{suf} is the solution to

$$\sqrt{12v^*} - 3\sqrt{(1-v^*)(C_{\text{suf}} - \log(1-v^*) + \log v^*)} = 0,$$

and is approximately equal to $2/3$.

Proof. We begin with the expression for the travel time between two consecutive demands using the constant bearing control (see Definition 6.5),

$$T = \frac{\sqrt{(1-v^2)\Delta x^2 + \Delta y^2}}{1-v^2} - \frac{v\Delta y}{1-v^2} \le \frac{|\Delta x|}{\sqrt{1-v^2}} + \frac{\Delta y}{1-v^2}, \tag{6.9}$$

where we used the inequality $\sqrt{a^2+b^2} \le |a| + |b|$. Taking expectation,

$$\mathbb{E}[T] \le \frac{W}{3\sqrt{1-v^2}} + \frac{v}{\lambda(1-v^2)},$$

since the demands are distributed uniformly in the x-direction and Poisson in the y-direction. A sufficient condition for stability is

$$\lambda\mathbb{E}[T] < 1 \quad\iff\quad \lambda < \frac{3}{W}\sqrt{\frac{1-v}{1+v}}. \tag{6.10}$$

The upper bound on T given by equation (6.9) is very conservative except for the case when v is very small. Alternatively, taking expected value of T conditioned on Δy, and applying Jensen's inequality to the square-root part, we obtain

$$\mathbb{E}[T|\Delta y] \le \frac{1}{1-v^2}\left(\sqrt{(1-v^2)W^2/6 + \Delta y^2} - v\Delta y\right),$$

since $\mathbb{E}[\Delta x^2] = W^2/6$. Following steps which are similar to those between equation (6.6) and equation (6.7), we obtain

$$\mathbb{E}\left[T\right] \leq \frac{\pi W}{2 \cdot \sqrt{6}\sqrt{1-v^2}} \left[\mathbf{H}_1\left(\frac{\lambda W\sqrt{1-v^2}}{\sqrt{6}v}\right) - \mathbf{Y}_1\left(\frac{\lambda W\sqrt{1-v^2}}{\sqrt{6}v}\right)\right] - \frac{v^2}{\lambda(1-v^2)}. \tag{6.11}$$

In (Newman, 1984), polynomial approximations have been provided for the Struve and Bessel functions in the intervals $[0,3]$ and $[3,+\infty)$. We seek an upper bound for the right-hand side of (6.11) when v is sufficiently large, i.e., when the argument of $\mathbf{H}_1$ and $\mathbf{Y}_1$ is small. From (Newman, 1984), we know that

$$\mathbf{H}_1(z) \leq \frac{z}{2}, \quad \mathbf{Y}_1(z) \geq \frac{2}{\pi}\left(\mathbf{J}_1(z)\log\frac{z}{2} - \frac{1}{z}\right), \quad \text{and } \mathbf{J}_1(z) \leq \frac{z}{2}, \quad \text{for } 0 \leq z \leq 3,$$

where $z := \lambda W\sqrt{1-v^2}/(\sqrt{6}v)$, and $\mathbf{J}_1 : \mathbb{R} \to \mathbb{R}$ denotes the Bessel function of the first kind. To obtain a lower bound on $\mathbf{Y}_1(z)$, we observe that if $0 \leq z \leq 2$, then due to the log term in the above lower bound for $\mathbf{Y}_1(z)$, we can substitute $z/2$ in place of $\mathbf{J}_1(z)$. Thus, we obtain

$$\mathbf{H}_1(z) \leq \frac{z}{2}, \quad \mathbf{Y}_1(z) \geq \frac{2}{\pi}\left(\frac{z}{2}\log\frac{z}{2} - \frac{1}{z}\right), \quad \text{for } 0 \leq z \leq 2. \tag{6.12}$$

Substituting into equation (6.11), we obtain

$$\mathbb{E}\left[T\right] \leq \frac{\pi W}{2 \cdot \sqrt{6}\sqrt{1-v^2}} \left[\frac{\lambda W\sqrt{1-v^2}}{2\sqrt{6}v} + \frac{2}{\pi}\left(\frac{\sqrt{6}v}{\lambda W\sqrt{1-v^2}} \right.\right.$$
$$\left.\left. - \frac{\lambda W\sqrt{1-v^2}}{2\sqrt{6}v}\log\frac{\lambda W\sqrt{1-v^2}}{2\sqrt{6}v}\right)\right] - \frac{v^2}{\lambda(1-v^2)},$$

which yields

$$\mathbb{E}\left[T\right] \leq \frac{\lambda W^2}{12v}\left(\frac{\pi}{2} - \log\frac{\lambda W}{3} - \log\frac{\sqrt{3}\sqrt{1-v^2}}{2\sqrt{2}v}\right) - \frac{1}{\lambda(1+v)}. \tag{6.13}$$

Now, let λ^* be the least upper bound on λ for which the FCFS policy is unstable, i.e., for every $\lambda < \lambda^*$, the FCFS policy is stable. To obtain λ^*, we need to solve $\lambda^*\mathbb{E}\left[T\right] = 1$. Using equation (6.13), we can obtain a lower bound on λ^* by simplifying

$$\frac{\lambda^{*2}W^2}{12v}\left(\frac{\pi}{2} - \log\frac{\lambda^* W}{3} - \log\frac{\sqrt{3}\sqrt{1-v^2}}{2\sqrt{2}v}\right) - \frac{1}{1+v} \geq 1.$$

From the condition given by equation (6.10), the second term in the parentheses satisfies

$$\frac{\lambda^* W}{3} > \sqrt{\frac{1-v}{1+v}}.$$

Thus, we obtain,

$$\lambda^* \geq \frac{\sqrt{12v}}{W\sqrt{(1+v)\left(C_{\text{suf}} - \log\left(\frac{1-v}{v}\right)\right)}},$$

where $C_{\text{suf}} = \pi/2 - \log(0.5 \cdot \sqrt{3}/\sqrt{2})$. Since $\lambda < \lambda^*$ implies stability, a sufficient condition for stability is

$$\lambda < \frac{\sqrt{12v}}{W\sqrt{(1+v)\left(C_{\text{suf}} - \log\left(\frac{1-v}{v}\right)\right)}}. \tag{6.14}$$

To determine the value of the speed v^*_{suf} beyond which this is a less conservative condition than equation (6.10), we solve

$$\frac{\sqrt{12v^*_{\text{suf}}}}{W\sqrt{(1+v^*_{\text{suf}})\left(C - \log\left(\frac{1-v^*_{\text{suf}}}{v^*_{\text{suf}}}\right)\right)}} = \frac{3}{W}\sqrt{\frac{1-v^*_{\text{suf}}}{1+v^*_{\text{suf}}}}.$$

For $v > v^*_{\text{suf}}$, one can verify that the numerical value of the argument of the Struve and Bessel functions is less than 2, and so the bounds given by equation (6.12) used in this analysis are valid. Thus, a sufficient condition for stability is given by equation (6.10) for $v \leq v^*_{\text{suf}}$, and by equation (6.14) for $v > v^*_{\text{suf}}$. □

Remark 6.14 (Tightness in low speed regime)**.** *As $v \to 0^+$, the sufficient condition for FCFS stability becomes $\lambda < 3/W$, which is exactly equal to the necessary condition given by part (ii) of Lemma 6.12. Thus, the condition for stability is asymptotically tight in this limiting regime.*

Figure 6.7 shows a comparison of the necessary and sufficient stability conditions for the FCFS policy. It should be noted that λ converges to 0^+ extremely slowly as v tends to 1^-, and still satisfy the sufficient stability condition in Lemma 6.13. For example, with $v = 1 - 10^{-6}$, the FCFS policy can stabilize the system for an arrival rate of $3/(5W)$. •

6.3 Proofs of the Main results

In this section, we present the proofs of the main results which were presented in Section 6.1.3.

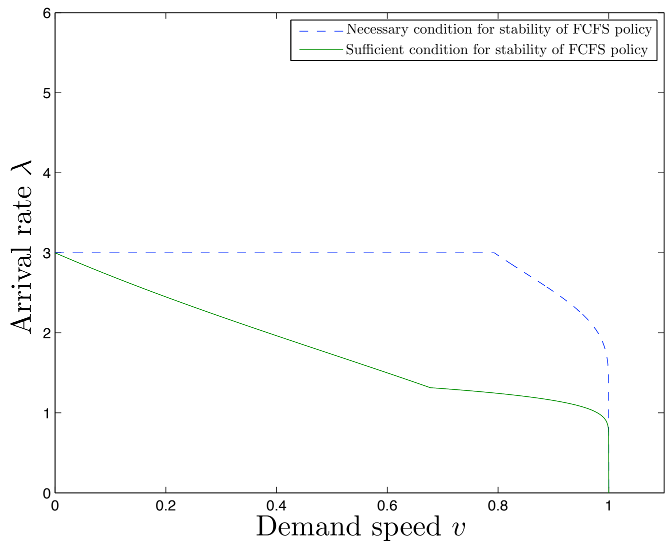


Figure 6.7: The necessary and sufficient conditions for the stability for the FCFS policy. The dashed curve is the necessary condition for stability as established in Lemma 6.12; while the solid curve is the sufficient condition for stability as established in Lemma 6.13.

6.3.1 Proof of Theorem 6.1 (Necessary stability condition)

We first present the proof of part (i). We begin by looking at the distribution of demands in the service region.

Lemma 6.15 (Distribution of outstanding demands)**.** *Suppose the generation of demands commences at time* 0 *and no demands are serviced in the interval* $[0,t]$*. Let* $\mathcal{Q}$ *denote the set of all demands in* $[0,W]\times[0,vt]$ *at time* t*. Then, given a measurable compact region* $\mathcal{R}$ *of area* $|\mathcal{R}|$ *contained in* $[0,W]\times[0,vt]$*,*

$$\mathbb{P}[|\mathcal{R}\cap\mathcal{Q}|=N]=\frac{\mathrm{e}^{-\bar{\lambda}|\mathcal{R}|}(\bar{\lambda}|\mathcal{R}|)^N}{N!},\quad \textit{where } \bar{\lambda}:=\lambda/(vW).$$

Proof. Let $\mathcal{R}=[\ell,\ell+\Delta\ell]\times[h,h+\Delta h]$ be a rectangle contained in $[0,W]\times[0,vt]$ with area $|\mathcal{R}|=\Delta\ell\Delta h$. Let us calculate the probability that at time t, $|\mathcal{R}\cap\mathcal{Q}|=N$ (that is, the probability that $\mathcal{R}$ contains n points in $\mathcal{Q}$). We have

$$\mathbb{P}[|\mathcal{R}\cap\mathcal{Q}|=N]=\sum_{i=N}^{\infty}\mathbb{P}\left[i \text{ demands arrived in } \left[\frac{h}{v},\frac{h+\Delta h}{v}\right]\right] \\ \times\mathbb{P}[n \text{ of } i \text{ are generated in } [\ell,\ell+\Delta\ell]].$$

Since the generation process is temporally Poisson and spatially uniform the above equation can be rewritten as

$$\mathbb{P}[|\mathcal{R} \cap \mathcal{Q}| = N] = \sum_{i=N}^{\infty} \mathbb{P}\left[i \text{ demands arrived in } [0, \Delta h/v]\right] \times \mathbb{P}[n \text{ of } i \text{ are generated in } [0, \Delta\ell]]. \quad (6.15)$$

Now we compute

$$\mathbb{P}\left[i \text{ demands arrived in } [0, \Delta h/v]\right] = \frac{\mathrm{e}^{-\lambda\Delta h/v}(\lambda\Delta h/v)^i}{i!},$$

and

$$\mathbb{P}[n \text{ of } i \text{ are in } [0, \Delta\ell]] = \binom{i}{N}\left(\frac{\Delta\ell}{W}\right)^N\left(1 - \frac{\Delta\ell}{W}\right)^{i-N},$$

so that, substituting these expressions and adopting the shorthands $\mathcal{L} := \Delta\ell/W$ and $H := \Delta h/v$, equation (6.15) becomes

$$\mathbb{P}[|\mathcal{R} \cap \mathcal{Q}| = N] = \mathrm{e}^{-\lambda H}\mathcal{L}^N \sum_{i=N}^{\infty} \frac{(\lambda H)^i}{i!}\binom{i}{N}(1-\mathcal{L})^{i-N}. \quad (6.16)$$

Rewriting $(\lambda H)^i$ as $(\lambda H)^N(\lambda H)^{i-N}$, and using the definition of binomial $\binom{i}{N} = \frac{i!}{N!(i-N)!}$, equation (6.16) reads

$$\begin{aligned}\mathbb{P}[|\mathcal{R} \cap \mathcal{Q}| = N] &= \mathrm{e}^{-\lambda H}\frac{(\lambda\mathcal{L}H)^N}{N!}\sum_{j=0}^{\infty}\frac{(\lambda H(1-\mathcal{L}))^j}{j!} \\ &= \mathrm{e}^{-\lambda H + \lambda H(1-\mathcal{L})}\frac{(\lambda\mathcal{L}H)^N}{N!} = \mathrm{e}^{-\lambda\mathcal{L}H}\frac{(\lambda\mathcal{L}H)^N}{N!}.\end{aligned}$$

Finally, since $\mathcal{L}H = |\mathcal{R}|/(vW)$, we obtain

$$\mathbb{P}[|\mathcal{R} \cap \mathcal{Q}| = N] = \mathrm{e}^{-\bar{\lambda}|\mathcal{R}|}\frac{(\bar{\lambda}|\mathcal{R}|)^N}{N!},$$

where $\bar{\lambda} := \lambda/(vW)$. Thus, the result is established for rectangles. However, every measurable, compact region can be written as a countable union of rectangles, and thus the result follows. □

Remark 6.16 (Uniformly distributed demands). *Lemma 6.15 shows us that the number of demands in an unserviced region with area A is Poisson distributed with parameter $\lambda A/(vW)$, and conditioned on this number, the demands are distributed uniformly.* •

Lemma 6.17 (Travel time bound)**.** *Consider the set $\mathcal{Q}$ of demands that are uniformly distributed in $\mathcal{E}$ at time t. Let T_d be a random variable giving the minimum amount of time required to travel to a demand in $\mathcal{Q}$ from a vehicle position selected* a priori*. Then*

$$\mathbb{E}\left[T_d\right] \geq \frac{1}{2}\sqrt{\frac{vW}{\lambda}}.$$

Proof. Let $\mathbf{p} = (X, Y)$ denote the vehicle location selected *a priori*. To obtain a lower bound on the minimum travel time, we consider the best-case scenario, when no demands have been serviced in the time interval $[0, t]$, and when the set $\mathcal{Q}$ contains many demands (i.e., t is very large). Consider a demand in $\mathcal{Q}$ with position (x, y) at time t. Using Proposition 6.6, we can write the travel time T from $\mathbf{p}$ to $\mathbf{q} := (x, y)$ implicitly as

$$T(\mathbf{p}, \mathbf{q})^2 = (X - x)^2 + ((Y - y) - vT(\mathbf{p}, \mathbf{q}))^2. \tag{6.17}$$

Next, define the set S_T as the collection of demands that can be reached from (X, Y) in T or fewer time units. From equation (6.17) we see that when $v < 1$, the set S_T is a disk of radius T centered at $(X, Y - vT)$. That is,

$$S_T := \{(x, y) \in \mathcal{E} \mid (X - x)^2 + ((Y - vT) - y)^2 \leq T^2\},$$

where we have omitted the dependence of T on $\mathbf{p}$ and $\mathbf{q}$. The area of the set S_T, denoted $|S_T|$, is upper bounded by πT^2, and the area is equal to πT^2 if the S_T does not intersect a boundary of $\mathcal{E}$. Now, by Lemma 6.15 the demands in an unserviced region are uniformly randomly distributed with density $\bar{\lambda} = \lambda/(vW)$. Let us compute the distribution of $T_d := \min_{\mathbf{q}\in\mathcal{Q}} T(\mathbf{p}, \mathbf{q})$. For every vehicle position $\mathbf{p}$ chosen before the generation of demands, the probability that $T_d > T$ is given by

$$\mathbb{P}[T_d > T] = \mathbb{P}[|S_T \cap \mathcal{Q}| = 0] \geq \mathrm{e}^{-\bar{\lambda}|S_T|} \geq \mathrm{e}^{-\lambda\pi T^2/(vW)}.$$

Hence we have

$$\mathbb{E}\left[T_d\right] \geq \int_0^{+\infty} \mathbb{P}[T_d > T]dT \geq \int_0^{+\infty} \mathrm{e}^{-\lambda\pi T^2/(vW)}dT = \frac{\sqrt{\pi}}{2\sqrt{\lambda\pi/(vW)}} = \frac{1}{2}\sqrt{\frac{vW}{\lambda}}.$$

□

We can now prove part (i) of Theorem 6.1.

Proof of part (i) of Theorem 6.1. A necessary condition for the stability of any policy is

$$\lambda \mathbb{E}[T] \leq 1,$$

where $\mathbb{E}[T]$ is the steady-state expected travel time between demands i and $i+1$. For every policy $\mathbb{E}[T] \geq \mathbb{E}[T_d] \geq \frac{1}{2}\sqrt{\frac{vW}{\lambda}}$. Thus, a necessary condition for stability is that

$$\lambda \frac{1}{2}\sqrt{\frac{vW}{\lambda}} \leq 1 \qquad \Longleftrightarrow \qquad \lambda \leq \frac{4}{vW}.$$

□

Remark 6.18 (Constant fraction service)**.** *A necessary condition for the existence of a policy which services a fraction $\xi \in]0,1]$ of the demands is that*

$$\lambda \leq \frac{4}{\xi^2 vW}.$$

Thus, for a fixed $v \in]0,1[$ no policy can service a constant fraction of the demands as $\lambda \to +\infty$. This follows because in order to service a fraction ξ we require that $\xi\lambda\mathbb{E}[T_d] < 1$.

In order to service a fraction ξ of the demands, we consider a subset of the generator having length ξW, with the arrival rate on that subset being equal to $\xi\lambda$. The use of the TMHP-Based *policy on this subset and with the arrival rate $\xi\lambda$ gives a sufficient condition for stability analogous to Theorem 6.2, but with an extra term of ξ^2 in the denominator.* •

For the proof of part (ii) of Theorem 6.1, we first recall from Lemma 6.12 that for stability of the FCFS policy, although λ must go to zero as $v \to 1^-$, it can go very slowly to 0. Specifically, λ goes to zero as

$$\frac{1}{\sqrt{-\log(1-v)}}.$$

This quantity goes to zero more slowly than any polynomial in $(1-v)$. We are now ready to complete the proof of Theorem 6.1.

Proof of part (ii) of Theorem 6.1. Observe that the condition on λ in the statement of part (ii) is the expression given by the necessary condition for FCFS stability in the asymptotic regime as $v \to 1^-$, from Lemma 6.12. Therefore, suppose there is a policy P that does not serve demands FCFS, but can stabilize the system with

$$\lambda = B(1-v)^p,$$

for some $p > 0$, and $B > 0$. Let t_i be the first instant at which policy P deviates from FCFS. Then, the demand served immediately after i is demand $i+k$ for some $k > 1$. When the vehicle reaches demand $i + k$ at time t_{i+1}, demand $i + 1$ has moved above the vehicle. To ensure stability, demand $i + 1$ must eventually be served. The time to travel to demand $i + 1$ from any demand $i + j$, where $j > 1$, is

$$\begin{aligned} T(\mathbf{q}_{i+j}, \mathbf{q}_{i+1}) &= \sqrt{\left(\frac{\Delta x}{\sqrt{1-v^2}}\right)^2 + \left(\frac{\Delta y}{1-v^2}\right)^2} + \frac{v\Delta y}{1-v^2} \\ &\geq \frac{\Delta y}{1-v^2} + \frac{v\Delta y}{1-v^2} = \frac{\Delta y}{1-v}, \end{aligned}$$

where Δx and Δy are now the minimum of the x and y distances from $\mathbf{q}_{i+j}$ to the $\mathbf{q}_{i+1}$. The random variable Δy is Erlang distributed with shape $j - 1 \geq 1$ and rate λ, implying

$$\mathbb{P}[\Delta y \leq c] \leq 1 - \mathrm{e}^{-\lambda c/v}, \quad \text{for each } c > 0, \text{ and in particular, for } c = (1-v)^{1/2-p}.$$

Now, since $\lambda = B(1 - v)^p$ as $v \to 1^-$, almost surely $\Delta y > (1 - v)^{1/2-p}$. Thus

$$T(\mathbf{q}_{i+j}, \mathbf{q}_{i+1}) \geq (1-v)^{-(p+1/2)},$$

almost surely as $v \to 1^-$. Thus, the expected number of demands that arrive during $T(\mathbf{q}_{i+j}, \mathbf{q}_{i+1})$ is

$$\lambda T(\mathbf{q}_{i+j}, \mathbf{q}_{i+1}) \geq B(1-v)^p(1-v)^{-(p+1/2)} \geq B(1-v)^{-1/2} \to +\infty,$$

as $v \to 1^-$. This implies that almost surely the policy P becomes unstable when it deviates from FCFS and that any deviation must occur with probability zero as $v \to 1^-$. Thus, a necessary condition for a policy to be stabilizing with $\lambda = B(1 - v)^p$ is that, as $v \to 1^-$, the policy must serve demands in the order in which they arrive. But this needs to hold for every p and, by letting p go to infinity, $B(1 - v)^p$ converges to zero for all $v \in (0, 1]$. Thus, a non-FCFS policy cannot stabilize the system no matter how quickly $\lambda \to 0^+$ as $v \to 1^-$. Hence, as $v \to 1^-$, every stabilizing policy must serve the demands in the order in which they arrive. Additionally, notice that the definition of the FCFS policy is that it uses the minimum time control (i.e., constant bearing control) to move between demands, thus the expression in part (ii) of Theorem 6.1 is a necessary condition for all stabilizing policies as $v \to 1^-$. □

6.3.2 Proof of Theorems 6.2 and 6.4 (TMHP-Based stability)

We first present the proof of Theorem 6.2. We begin with the proof of part (i).

Proof of part (i) of Theorem 6.2. If there are any demands "above" the vehicle initially, at the end of the first iteration of the TMHP-BASED policy, all outstanding demands have their y-coordinates less than or equal to that of the vehicle, and hence would be located "below" the vehicle as shown in the first of Figure 6.3. Hence at the end of every iteration of the TMHP-BASED policy, all outstanding demands would be located "below" the vehicle.

Let the vehicle be located at $\mathbf{p}(t_i) = (X(t_i), Y(t_i))$ and $\mathbf{q}_{\text{last}}$ denote the demand with the least y-coordinate at time instant t_i. Let $|\mathcal{Q}|$ denote the number of demands in the set $\mathcal{Q}$. If there exists a non-empty set of unserviced demands $\mathcal{Q}$ below the vehicle at time t_i, then letting $\tilde{\mathcal{Q}} = \mathcal{Q} \setminus \mathbf{q}_{\text{last}}$ we have

$$\begin{aligned} Y(t_{i+1}) &= v\mathrm{TMHP}_v(\mathbf{p}(t_i), \tilde{\mathcal{Q}}(t_i), \mathbf{q}_{\text{last}}(t_i)) + y_{\text{last}}(t_i)), \\ &\qquad\qquad \text{w.p. } \mathbb{P}(|\mathcal{Q}| = k), \text{ for } k \in \{1, 2, \dots\}, \end{aligned}$$

where $y_{\text{last}}(t_i)$ is the y-coordinate of $\mathbf{q}_{\text{last}}(t_i)$ and $\mathrm{TMHP}_v(\mathbf{p}(t_i), \tilde{\mathcal{Q}}(t_i), \mathbf{q}_{\text{last}}(t_i))$ is the time taken for the vehicle to travel the TMHP that begins at $\mathbf{p}(t_i)$, serves all demands in $\mathcal{Q}$ other than $\mathbf{q}_{\text{last}}$ and ends at the demand $\mathbf{q}_{\text{last}}$ (see Section 2.2.3).

We seek an upper bound for the length TMHP_v of the TMHP for which we use the CONVERT-TO-EMHP method (see Section 2.2.3). Invoking Lemma 2.4, and writing $Y_i := Y(t_i)$ for convenience, we have

$$\begin{aligned} &\mathrm{TMHP}_v(\mathbf{p}(t_i), \tilde{\mathcal{Q}}(t_i), \mathbf{q}_{\text{last}}(t_i)) \\ &= \mathrm{EMHP}(\mathrm{cnvrt}_v(\mathbf{p}(t_i)), \mathrm{cnvrt}_v(\tilde{\mathcal{Q}}(t_i), \mathrm{cnvrt}_v(\mathbf{q}_{\text{last}}(t_i))) + \frac{v(y_{\text{last}}(t_i) - Y_i)}{1 - v^2} \\ &\le \sqrt{\frac{2W(Y_i - y_{\text{last}}(t_i))k}{(1 - v^2)^{3/2}}} + \frac{Y_i - y_{\text{last}}(t_i)}{1 + v} + \frac{5W}{2\sqrt{1 - v^2}} \\ &\le \sqrt{\frac{2WY_i k}{(1 - v^2)^{3/2}}} + \frac{Y_i}{1 + v} + \frac{5W}{2\sqrt{1 - v^2}}, \end{aligned}$$

where the first inequality is obtained using Lemma 2.1, and the second inequality follows since $y_{\text{last}}(t_i) \ge 0$.

If $\mathcal{Q}$ is empty at time t_i, then the vehicle moves towards the optimal location (X^*, Y^*). When a new demand arrives, the vehicle moves towards it. If $Y_i \le W$, then in the worst-case, the vehicle is very close to an endpoint of the generator and the next demand arrives at the other endpoint. In this case, the vehicle moves with a vertical velocity component equal to v and horizontal component equal to $\sqrt{1 - v^2}$. So in the worst-case, the vehicle is at a height $vW/\sqrt{1 - v^2}$ at the beginning of the next iteration. The other possibility is if $Y_i > W$. In this case, to get an upper bound on the height of the vehicle at the next iteration, we consider the vehicle motion when it first moves horizontally so that the x-coordinate equals that of the demand, and then moves vertically down to meet the demand. This gives an upper bound on the height of the vehicle at the next iteration as $v(Y_i - vW)/(1+v)$. Thus, if $\mathcal{Q}$ is empty, then the sum of these two upper bounds is trivially an upper bound on the height of the vehicle at the beginning of the next iteration. Thus, if $\mathcal{Q}$ is empty, then

$$Y_{i+1} \leq \frac{vW}{\sqrt{1-v^2}} + \frac{v}{1+v}(Y_i - vW) \leq \frac{vW}{\sqrt{1-v^2}} + \frac{vY_i}{1+v}.$$

Conditioned on Y_i, we have

$$\mathbb{E}\left[Y_{i+1}\middle|Y_i\right] \leq \left(\frac{vW}{\sqrt{1-v^2}} + \frac{vY_i}{1+v}\right)\mathbb{P}(|\mathcal{Q}|=0|Y_i) + v\sum_{k=1}^{\infty}\left(\sqrt{\frac{2WY_ik}{(1-v^2)^{3/2}}} + \frac{Y_i}{1+v} + \frac{5W}{2\sqrt{1-v^2}}\right)\mathbb{P}(|\mathcal{Q}|=k|Y_i).$$

Collecting the terms with $vY_i/(1+v)$ together, we obtain

$$\begin{aligned}\mathbb{E}\left[Y_{i+1}\middle|Y_i\right] &\leq \frac{vW}{\sqrt{1-v^2}}\mathbb{P}(|\mathcal{Q}|=0|Y_i) + \frac{vY_i}{1+v}\sum_{k=0}^{\infty}\mathbb{P}(|\mathcal{Q}|=k|Y_i) \\ &\quad + \sum_{k=1}^{\infty}\left(\sqrt{\frac{2v^2WY_ik}{(1-v^2)^{3/2}}} + \frac{5vW}{2\sqrt{1-v^2}}\right)\mathbb{P}(|\mathcal{Q}|=k|Y_i) \\ &\leq \frac{vW}{\sqrt{1-v^2}} + \frac{vY_i}{1+v} + \sqrt{\frac{2v^2W}{(1-v^2)^{3/2}}}\mathbb{E}\left[\sqrt{|\mathcal{Q}|Y_i}\middle|Y_i\right] \\ &\quad + \frac{5vW}{2\sqrt{1-v^2}}\sum_{k=1}^{\infty}\mathbb{P}(|\mathcal{Q}|=k|Y_i) \\ &\leq \frac{vY_i}{1+v} + \sqrt{\frac{2v^2W}{(1-v^2)^{3/2}}}\sqrt{Y_i}\mathbb{E}\left[\sqrt{|\mathcal{Q}|}\middle|Y_i\right] + \frac{7vW}{2\sqrt{1-v^2}}. \end{aligned} \tag{6.18}$$

Applying Jensen's inequality to the conditional expectation in the second term in the right hand side of equation (6.18), we have

$$\mathbb{E}\left[\sqrt{|\mathcal{Q}|}\middle|Y_i\right] \leq \sqrt{\mathbb{E}\left[|\mathcal{Q}|\middle|Y_i\right]} = \sqrt{\lambda\frac{Y_i}{v}},$$

where the equality follows since the arrival process is Poisson with rate λ and for a time interval Y_i/v. Substituting into equation (6.18), we obtain

$$\mathbb{E}\left[Y_{i+1}\middle|Y_i\right] \leq \left(\frac{v}{1+v} + \sqrt{\frac{2v\lambda W}{(1-v^2)^{3/2}}}\right) Y_i + \frac{7vW}{2\sqrt{1-v^2}}.$$

Using the law of iterated expectation, we have

$$\mathbb{E}\left[Y_{i+1}\right] = \mathbb{E}\left[\mathbb{E}\left[Y_{i+1}|Y_i\right]\right] \leq \left(\frac{v}{1+v} + \sqrt{\frac{2v\lambda W}{(1-v^2)^{3/2}}}\right) \mathbb{E}\left[Y_i\right] + \frac{7vW}{2\sqrt{1-v^2}}, \quad (6.19)$$

which is a linear recurrence in $\mathbb{E}[Y_i]$. Thus, $\lim_{i\to+\infty}\mathbb{E}[Y_i]$ is finite if

$$\frac{v}{1+v} + \sqrt{\frac{2Wv\lambda}{(1-v^2)^{3/2}}} < 1 \qquad \iff \qquad \lambda < \frac{(1-v^2)^{3/2}}{2Wv(1+v)^2}.$$

Thus, if λ satisfies the above condition, then expected number of demands in the environment is finite and the TMHP-BASED policy is stable.

Finally, from Lemma 6.7, the region of stability for the FCFS policy is contained in the region of stability for the TMHP-BASED policy. Thus, the TMHP-BASED policy is stable for all arrival rates satisfying the bound in Lemma 6.13. This gives us the desired result. □

Remark 6.19 (Upper bound on expected delay)**.** *Since equation* (6.19) *is a linear recurrence in* $\mathbb{E}[Y_i]$*, we can easily obtain an upper bound for* $\lim_{i\to+\infty}\mathbb{E}[Y_i]$*. Moreover, we may upper bound the expected delay for a demand by*

$$\frac{7W}{2\sqrt{1-v^2}}\left(\frac{1}{1/(1+v) - \sqrt{2Wv\lambda/(1-v^2)^{3/2}}}\right). \qquad \bullet$$

Proof of part (ii) of Theorem 6.2. In this part, we make use of the following two facts. First, as $v \to 0^+$, the length of the TMHP constrained to start at the vehicle location and end at the lowest demand, is equal to the length of the EMHP in the corresponding static instance under the map cnvrt_v, as described in Lemma 2.4. Second, consider a set $\mathcal{Q}$ of N points which are uniformly distributed in a region with finite area. Then, in the limit as $N \to +\infty$, the length of a constrained EMHP through $\mathcal{Q}$ tends to the length of the ETSP tour through $\mathcal{Q}$.

More specifically, consider the ith iteration of the TMHP-BASED policy, and let $Y_i > 0$ be the position of the service vehicle. In the limit as $\lambda \to +\infty$, the number of outstanding demands in that iteration $N_i \to +\infty$, and in addition, conditioned on N_i, the demands are uniformly distributed in the region $[0, W] \times [0, Y_i]$ (see Remark 6.16). Now using the above two facts, we can apply Theorem 2.3 to obtain an expression for the length of the TMHP constrained to start at the vehicle location and ending at the lowest demand. As $\lambda \to +\infty$, the position of the vehicle at the end of the ith iteration is given by

$$Y_{i+1} = v\beta_{\text{TSP}}\sqrt{N_i A} = v\beta_{\text{TSP}}\sqrt{N_i Y_i W},$$

where $A := Y_i W$ is the area of the region below the vehicle at the ith iteration. Thus, conditioned on Y_i being bounded away from 0, we have

$$\mathbb{E}\left[Y_{i+1}|Y_i\right] = v\beta_{\text{TSP}}\sqrt{Y_i}\mathbb{E}\left[\sqrt{W n_i}\right] \leq v\beta_{\text{TSP}}\sqrt{W Y_i \mathbb{E}\left[N_i\right]},$$

where we have applied Jensen's inequality. Using Lemma 6.15, $\mathbb{E}\left[N_i\right] = W Y_i \lambda/(vW)$ and thus

$$\mathbb{E}\left[Y_{i+1}|Y_i\right] \leq v\beta_{\text{TSP}}\sqrt{W^2 Y_i^2 \frac{\lambda}{vW}} = \beta_{\text{TSP}}\sqrt{\lambda v W}Y_i.$$

Thus, the sufficient condition for stability of the TMHP-Based policy as $\lambda \to +\infty$ (and thus $v \to 0^+$) is

$$\lambda < \frac{1}{\beta_{\text{TSP}}^2 vW} \approx \frac{1.9726}{vW}.$$

□

Finally, we present the proof of Theorem 6.4.

Proof of Theorem 6.4. The proof of part (i) of Theorem 6.4 follows from Lemma 6.9 and Lemma 6.7. The proof of part (ii) follows from part (ii) of Theorem 6.1 and Lemma 6.7 along with the fact that the TMHP-Based policy spends the minimum amount of time to travel between demands. □

6.4 Simulations

In this section, we present a numerical study to determine stability of the TMHP-Based policy. We numerically determine the region of stability of the TMHP-Based policy, and compare it with the theoretical results from the previous sections.

The `linkern`solver was used to generate approximations to the TMHP at every iteration of the policy, by generating an approximate solution to the corresponding constrained EMHP problem. The stability properties of the TMHP-Based policy were determined according to the following procedure.

(i) For a collection of instructive pairs of the demand speed v and λ in the region of interest, we set the generator width $W = 1$, we set the number of outstanding demands $N_0 = 1000$ and we determined the corresponding initial height of the environment of interest by $h_0 = vN_0/\lambda$.

(ii) We repeated 10 times the following procedure. The vehicle is placed at the height h_0 and at a uniformly random location in the horizontal direction. A number N_0 of outstanding demands are uniformly randomly placed in the environment (see Lemma 6.15). The vehicle uses the TMHP-Based policy to serve all outstanding demands and we store the height h_1 of the vehicle at the end of the single iteration of the policy. Finally, we compute the average height $\bar{h}_1$ of the 10 iterations.

(iii) If $\bar{h}_1 \leq h_0$, then the policy is deemed to be stable for the chosen value of (v, λ). Otherwise the policy is deemed to be unstable.

The results of this numerical experiment are presented in Figure 6.8. For the purpose of comparison, we overlay the plots for the theoretical curves, which were presented in Figure 6.1. We observe that the numerically obtained stability boundary for the TMHP-Based policy falls between the necessary and the sufficient conditions which were established in parts (i) of Theorems 6.1 and 6.2 respectively. In addition, notice that the sufficient condition in the low speed regime, characterized in part (ii) of Theorem 6.2, serves as a very good approximation to the stability boundary, for nearly the entire range of demand speeds.

6.5 Summary

We introduced a dynamic vehicle routing problem with translating demands. We determined a necessary condition on the arrival rate of the demands for the existence of a stabilizing policy. In the limit when the demands move as fast as the vehicle, we showed that every stabilizing policy must service the demands in the FCFS order. We proposed a novel receding horizon policy that services the moving demands as per a translational minimum Hamiltonian path. In the asymptotic regime when the demands move as fast as the vehicle, we showed that the TMHP-Based policy minimizes the expected time to service a demand. We derived a sufficient condition for stability of the TMHP-Based policy, and showed that in the asymptotic regime of low demand speed, the sufficient condition is within a constant factor of the necessary condition for stability. In a third asymptotic regime when arrival rate tends to zero for a fixed demand speed, we showed that the TMHP-Based policy is optimal in terms of minimizing the expected time to service a demand. Finally, we presented an implementation of the TMHP-Based policy to numerically determine its region of stability. We observe that the sufficient condition for the asymptotic regime

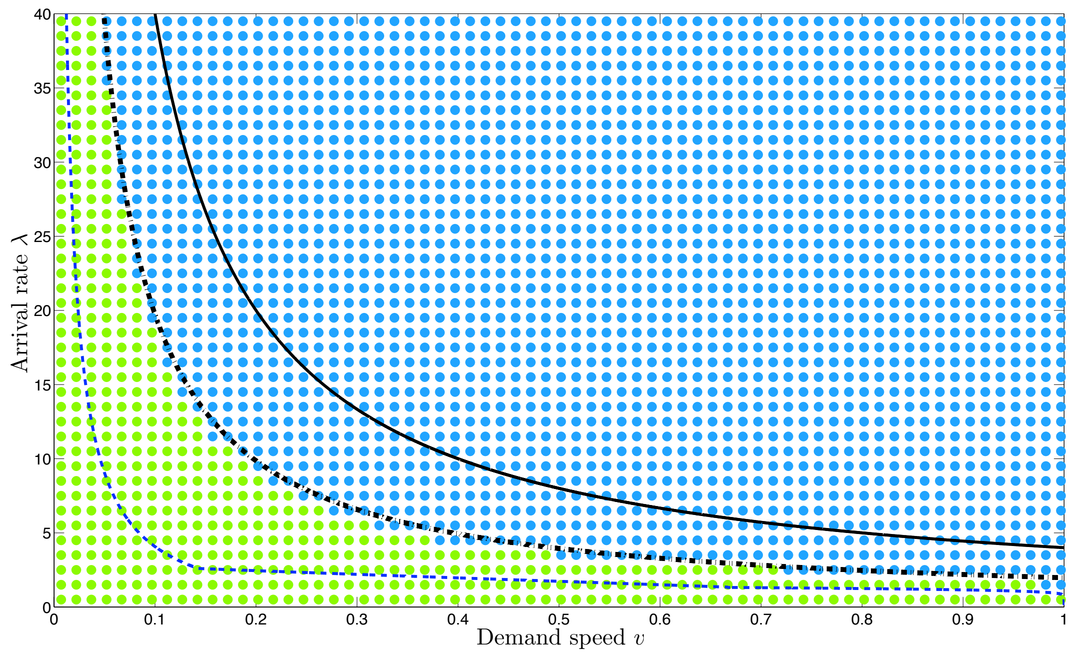


Figure 6.8: Numerically determined region of stability for the TMHP-Based policy. A lightly shaded (green-coloured) dot represents stability while a darkly shaded (blue–coloured) dot represents instability. The uppermost (thick solid) curve is the necessary condition for stability for any policy as derived in Theorem 6.1. The lowest (dashed) curve is the sufficient condition for stability of the TMHP-Based policy as established by Theorem 6.2. The broken curve between the two curves is the sufficient stability condition of the TMHP-Based policy in the low speed regime as derived in part (ii) of Theorem 6.2. The environment width is $W = 1$.

of low demand speeds serves as a good approximation to the boundary of the stability region for a significantly large interval of values of demand speed.

Chapter Seven

Boundary Guarding for Translating Demands

In this chapter we study a variation of the translating demands problem of Chapter 6, which can be described as follows (see Fig. 7.1): Demands arrive according to a stochastic process on a line segment of length W. Upon arrival the demands move with fixed speed v towards a deadline which is at a distance L from the generator. A unit speed service vehicle seeks to capture the demands before they reach the deadline (i.e., within L/v time units of being generated). The performance metric is the fraction of demands that are captured before reaching the deadline.

The contributions of this chapter are as follows. We introduce a dynamic boundary (deadline) guarding problem. For analysis, we assume that the arrival process is uniform along the line segment and temporally Poisson with rate λ. In the case when the demands are faster than the service vehicle (i.e., $v \geq 1$) we introduce the novel LONGEST PATH policy, which is based on computing longest paths in a directed acyclic *reachability graph*. When $L \geq vW$, we derive a lower bound on the capture fraction as a function of the system parameters. We show that the LONGEST PATH policy is the optimal policy when L is much greater than vW. In the case when the demands are slower than the service vehicle (i.e, $v < 1$), we propose a policy based on the translational minimum Hamiltonian path called the TMHP-FRACTION policy. In the limit of low demand speed and high arrival rate, the capture fraction of this policy is within a small constant factor of the optimal. We present numerical simulations which verify our results, and show that the LONGEST PATH policy performs very near the optimal even when $L < vW$.

The chapter is organized as follows. In Section 7.1 we formulate the problem. In Section 7.2 we consider the case of $v \geq 1$ and introduce the LONGEST PATH policy. In Section 7.3 we study $v < 1$ and introduce the TMHP-FRACTION policy. Finally, in Section 7.4 we present simulations results which show that the LONGEST PATH policy performs very near optimal over a large set of problem parameters.

The work in this chapter was primarily performed by Stephen L. Smith, but in collaboration with Shaunak D. Bopardikar.

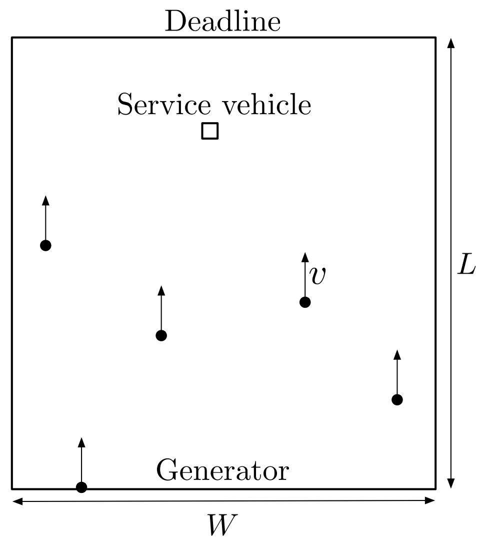


Figure 7.1: The problem setup. Demands are shown as black disks approaching the deadline at speed v. The service vehicle is a square.

7.1 Problem Formulation

Consider an environment $\mathcal{E} := [0, W] \times [0, L] \subset \mathbb{R}^2$ as shown in Figure 7.1. The line segment $[0, W] \times \{0\} \subset \mathcal{E}$ is termed the *generator*, and the segment $[0, W] \times \{L\} \subset \mathcal{E}$ is termed the *deadline*. The environment contains a single vehicle with position $\mathbf{p}(t) = [X(t), Y(t)]^T \in \mathcal{E}$, modeled as a first-order integrator with unit speed. Demands (or targets) arrive in the environment according to a temporal Poisson process with rate $\lambda > 0$. Upon arrival, each demand assumes a uniformly distributed location on the generator, and then moves with constant speed $v > 0$ in the positive y-direction towards the deadline. If the vehicle intercepts a demand before the demand reaches the deadline, then the demand is captured. On the other hand, if the demand reaches the deadline before being intercepted by the vehicle, then the demand escapes. Thus, to capture a demand, it must be intercepted within L/v time units of being generated.

We let $\mathcal{Q}(t) \subset \mathcal{E}$ denote the set of all outstanding demand locations at time t. If the ith demand to arrive is captured, then it is removed from $\mathcal{Q}$ and placed in the set $\mathcal{Q}_{\text{capt}}$ with cardinality N_{capt}. If the ith demand escapes, then it is removed from $\mathcal{Q}$ and placed in $\mathcal{Q}_{\text{esc}}$ with cardinality N_{esc}.

Causal Policy In this chapter, and in previous chapters, we are interested in causal policies. These are policies defined as a map $P : \mathcal{E} \times \text{FIN}(\mathcal{E}) \to \mathbb{R}^2$, where $\text{FIN}(\mathcal{E})$ is the set of finite subsets of $\mathcal{E}$, which assigns a commanded velocity to the service vehicle as a function of the current state of the system: $\dot{\mathbf{p}}(t) = P(\mathbf{p}(t), \mathcal{Q}(t))$.

Non-causal Policy In a non-causal feedback control policy the commanded velocity of the service vehicle is a function of the current and future state of the system. Such policies are not physically realizable, but they will prove useful in the upcoming analysis.

Formally, let the generation of demands commence at time $t = 0$, and consider the sequence of demands $(\mathbf{q}_1, \mathbf{q}_2, \ldots)$ arriving at increasing times $(t_1, t_2, \ldots)$, with x-coordinates $(x_1, x_2, \ldots)$. We can also model the arrival process by assuming that at time $t = 0$, all demands are located in $[0, W] \times (-\infty, 0]$, move in the y-direction at speed v for all $t > 0$, and are revealed to the service vehicle when they cross the generator. Thus, at time $t = 0$, the position of the ith demand is $(x_i, v(t - t_i))$. We can define a set containing the position of all demands in the region $[0, W] \times (-\infty, 0]$ at time t as $\mathcal{Q}_{\text{unarrived}}(t)$. Then, a non-causal policy is one for which $\dot{\mathbf{p}}(t) = P(\mathbf{p}(t), \mathcal{Q}(t) \cup \mathcal{Q}_{\text{unarrived}}(t))$.

Problem Statement The goal in this chapter is to find causal policies P that maximize the fraction of demands that are captured $\mathbb{F}_{\text{cap}}(P)$, termed the *capture fraction*, where

$$\mathbb{F}_{\text{cap}}(P) := \lim_{t \to +\infty} \mathbb{E}\left[\tfrac{N_{\text{capt}}(t)}{N_{\text{capt}}(t) + N_{\text{esc}}(t)}\right].$$

7.2 Demand Speed Greater Than Vehicle Speed

Here we develop a policy for the case when the demand speed $v \geq 1$. In this policy, the service vehicle remains on the deadline and services demands as per the longest path in a directed acyclic reachability graph. In this section we begin by introducing the reachability graph, and then proceed to state and analyze the Longest Path policy.

7.2.1 Reachable Demands

Consider a demand generated at time $t_1 \geq 0$ at position $(x, 0)$. The demand moves in the positive y-direction at speed $v \geq 1$, and thus $(x(t), y(t)) = (x, v(t - t_1))$ for each $t \in [t_1, t_f]$, where t_f is either the time of escape (i.e., $t_f = L/v + t_1$), or it is the time of capture. Now, given the service vehicle location $(X(t), Y(t))$, a demand with position $(x, y(t))$ is reachable if and only if

$$v|X(t) - x| \leq Y(t) - y(t). \tag{7.1}$$

That is, the service vehicle must be at a height of at least $v|X(t) - x|$ above the demand in order to capture it.

Definition 7.1 (Reachable set). *The reachable set from a position $(X, Y) \in \mathcal{E}$ is*

$$R(X, Y) := \{(x, y) \in \mathcal{E} : v|X - x| \leq |Y - y|\}.$$

If the service vehicle is located at (X, Y), then a demand can be captured if and only if it lies in the set $R(X, Y)$.

An example of the reachable set is shown in Figure 7.2. Next, given a demand in the reachable set, the following motion gives a method of capture.

Definition 7.2 (Intercept motion). *Consider a vehicle position $((X(\bar{t}), Y(\bar{t}))$ and a demand position $(x, y(\bar{t})) \in R(X(\bar{t}), Y(\bar{t}))$ at time $\bar{t} \geq 0$. In intercept motion, the service vehicle captures the demand by first moving horizontally at unit speed to the position $(x_i, Y(\bar{t}))$, and then waiting at the location for the demands arrival.*

Lemma 7.3 (Optimality of intercept motion). *Consider $v \geq 1$, and let the service vehicle be initially positioned on the deadline. Then, there is an optimal policy in which the service vehicle uses only intercept motion.*

Proof. Let the service vehicle be positioned at (X, L), and consider a demand at $(x, y) \in R(X, L)$. From equation (7.1), we have $v|X - x| \leq L - y$. If $v|X - x| = L - y$, then deadline motion is the only way in which the demand can be captured. Thus, assume that $v|X - x| < L - y$, and consider two cases; Case 1 in which intercept motion is used, and Case 2 in which the demand is captured at a location (x, Y), where $Y < L$.

Notice that the position of each outstanding demand relative to the service vehicle position at capture is the same in Case 1 as in Case 2. Thus, the reachable set in Case 2 is a strict subset of reachable set in Case 1 and the vehicle gains no advantage by moving off of the deadline. □

Next, consider the set of demands in $R(X(\bar{t}), Y(\bar{t}))$, and suppose the vehicle chooses to capture demand i, with position $\mathbf{q}_i(\bar{t}) = (x_i, y_i(\bar{t})) \in R(X(\bar{t}), Y(\bar{t}))$. Upon capture at time t_f, the service vehicle can recompute the reachable set, and select a demand that lies within. Since all demands translate together, every demand that was reachable from $\mathbf{q}_i(\bar{t})$, is reachable from $\mathbf{q}_i(t_f)$. Thus, the service vehicle can "look ahead" and compute the demands that will be reachable from each captured demand position. This idea motivates the concept of a reachability graph.

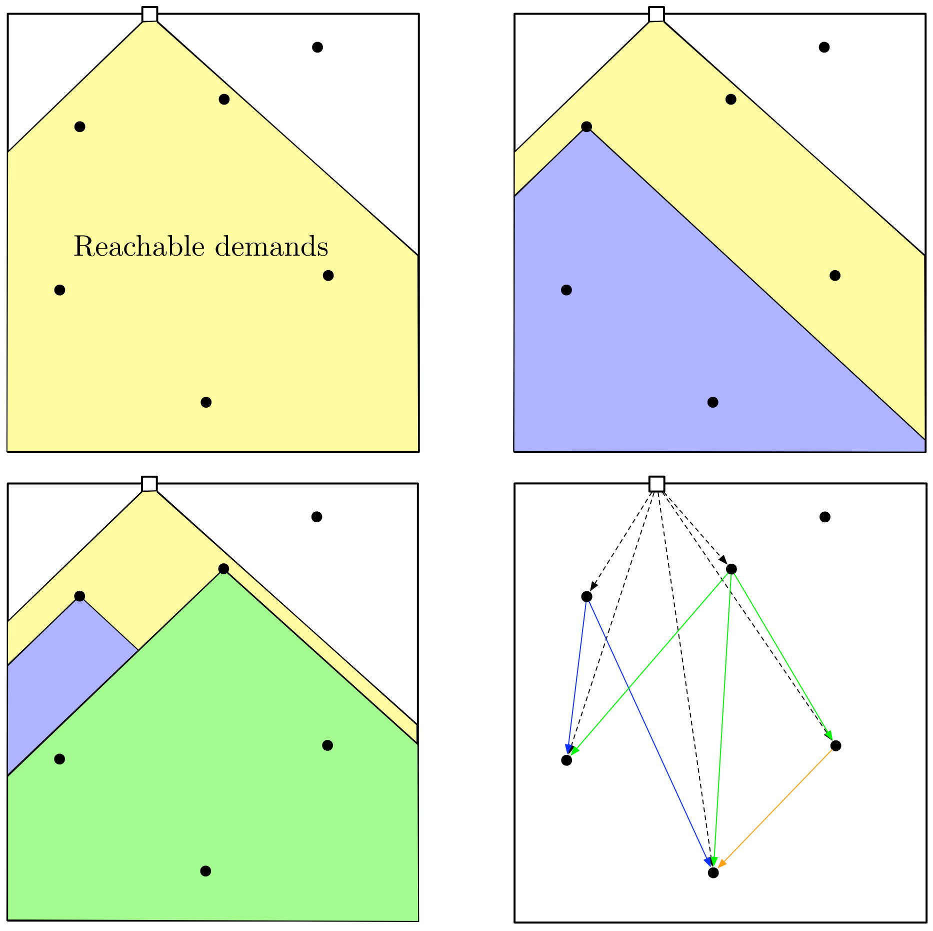


Figure 7.2: The construction of the reachability graph. The top-left figure shows the set of reachable points from a vehicle positioned on the deadline. The top-right and bottom-left figures show the reachable set from demand locations. The bottom-right figure shows the reachability graph.

Definition 7.4 (Reachability graph)**.** *For $v \geq 1$, the reachability graph of a set of points $\{\mathbf{q}_1, \ldots, \mathbf{q}_n\} \in \mathcal{E}$, is a directed acyclic graph with vertex set $V := \{1, \ldots, n\}$, and edge set E, where for $i, j \in V$, the edge (i, j) is in E if and only if $\mathbf{q}_j \in R(\mathbf{q}_i)$ and $j \neq i$.*

Given a set $\mathcal{Q}$ of N outstanding demands, and a vehicle position (X, Y), we can compute the corresponding reachability graph (see Fig. 7.2) in $O(N^2)$ computation time. In addition, by Section 2.2.4 we can compute the longest path in a reachability graph in $O(N^2)$ computation time.

7.2.2 A Non-causal Policy and Upper Bound

To derive an upper bound for $v \geq 1$, we begin by considering a non-causal policy, shown in Algorithm 7.1.

```
Algorithm 7.1: NON-CAUSAL LONGEST PATH (NCLP) policy
  Assumes: Vehicle is located on deadline and $v \geq 1$.
1 Compute the reachability graph of the vehicle position and all demands in
  $\mathcal{Q}(0) \cup \mathcal{Q}_{\text{unarrived}}(0)$.
2 Compute a longest path in this graph, starting at the service vehicle
  location.
3 Capture demands in the order they appear on the path, intercepting each
  demand on the deadline.
```

Figure 7.3 shows an example of a path generated by the NON-CAUSAL LONGEST PATH policy. Note that the service vehicle will intercept each demand on the deadline, and thus the path depicts which demands will be captured, and in what order.

Lemma 7.5 (Optimal non-causal policy)**.** *If $v \geq 1$, then the* NON-CAUSAL LONGEST PATH *policy is an optimal non-causal policy. Moreover, if $v \geq 1$, then for every causal policy P,*

$$\mathbb{F}_{\text{cap}}(P) \leq \mathbb{F}_{\text{cap}}(\text{NCLP}).$$

Proof. The reachability graph $\mathcal{Q}(0) \cup \mathcal{Q}_{\text{unarrived}}(0)$ contains every possible path that the service vehicle can follow. When $v \geq 1$ the graph is a directed acyclic graph and thus the longest path (i.e., the path which visits the most vertices in the graph) is well defined. The vehicle uses intercept motion, and thus by Lemma 7.3 the NCLP policy is an optimal non-causal policy, and its capture fraction upper bounds every causal policy. □

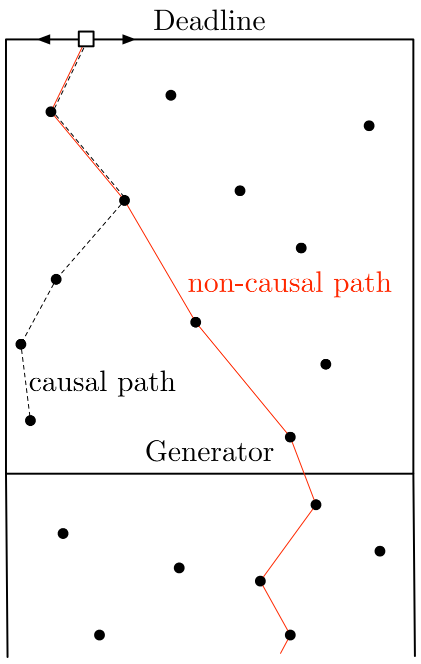


Figure 7.3: A snapshot in the evolution of the Non-causal Longest Path. The vehicle has planned the solid red path through all demands, including those that have not yet arrived. In comparison, a dashed causal longest path is shown, which only considers demands that have arrived.

7.2.3 The Longest Path Policy

We now introduce the Longest Path policy, shown in Algorithm 7.2.

Algorithm 7.2: Longest Path (LP) policy

```
  Assumes: Vehicle is located on deadline and $v \geq 1$
1 Compute the reachability graph of the vehicle position and all demands in
  $\mathcal{Q}(0)$.
2 Compute a longest path in this graph, starting at the service vehicle
  location.
3 Capture demands in the order they appear on the path, intercepting each
  demand on the deadline.
4 Once a fraction $\eta \in ]0, 1]$ of the demands on the path have been serviced,
  recompute the reachability graph of all outstanding demands and return to
  step 2.
```

In the LP policy, the fraction η is a design parameter. The lower η is chosen, the better the performance of the policy, but this comes at the expense of increased computation. In the following theorem, we relate the Longest Path policy to its non-causal relative.

Theorem 7.6 (Optimality of Longest Path policy). *If $v \geq 1$, then*

$$\mathbb{F}_{\text{cap}}(\text{LP}) \geq \left(1 - \frac{vW}{L}\right) \mathbb{F}_{\text{cap}}(\text{NCLP}),$$

and thus the LP policy is optimal as $vW/L \to +\infty$.

Proof. Suppose that the generation of demands begins at $t = 0$ and let us consider two scenarios; (a) the vehicle uses the Longest Path policy, and (b) the vehicle uses the Non-causal Longest Path policy. Then, at any instant in time $t_1 > 0$ we can compare the number of demands captured in scenario (a) to the number captured in scenario (b).

Let us consider a time instant t_1 where in scenario (a), the vehicle is recomputing the longest path through all outstanding demands $\mathcal{Q}(t_1)$. Let us denote by $\mathbf{p}_a(t_1)$ and $\mathbf{p}_b(t_1)$, the vehicle position in scenario (a) and scenario (b), respectively, at time t_1. In scenario (b), let the path that the vehicle will take through $\mathcal{Q}(t_1)$ be given by $(\mathbf{q}_{b_1}, \mathbf{q}_{b_2}, \dots, \mathbf{q}_{b_\ell})$, where $\mathbf{q}_{b_i} \in \mathcal{Q}(t_1)$ for each $i \in \{1, \dots, \ell\}$. The demand $\mathbf{q}_{b_1}$ is reachable from $\mathbf{p}_b(t_1)$, but it may not be reachable from $\mathbf{p}_a(t_1)$. However, a lower bound on the length of the longest path in scenario (a) is: $(\mathbf{q}_{b_{a+1}}, \mathbf{q}_{b_{a+2}}, \dots, \mathbf{q}_{b_\ell})$, where $\mathbf{q}_{b_{a+1}}$, with $a \in \{0, \dots, \ell - 1\}$, is the highest demand (i.e., the demand with the largest y-coordinate) that can be captured from $\mathbf{p}_a(t_1)$. Thus, the length of the longest path in scenario (a), $\mathcal{L}_a$, is at least

$$\mathcal{L}_a \geq \ell - a, \tag{7.2}$$

where ℓ is the length of the path in scenario (b).

Now, since the deadline has width W, the vehicle in scenario (a) can capture any demand (x, y) with $y \leq L - vW$. Thus, the demands $\mathbf{q}_{b_1}, \mathbf{q}_{b_2}, \ldots, \mathbf{q}_{b_\ell}$ must all have y-coordinates in $]L - vW, L]$. Let the total number of outstanding demands at time t_1 be N_{tot}. Then, conditioned on N_{tot}, by Lemma 6.15, the expected number of outstanding demands contained in $[0, W] \times]L - vW, L]$ is $N_{\text{tot}} vW/L$. Hence,

$$\mathbb{E}\left[a | N_{\text{tot}}\right] = N_{\text{tot}} \frac{vW}{L} \mathbb{F}_{\text{cap}}(\text{NCLP}). \tag{7.3}$$

Similarly, for the length of the path through $\mathcal{Q}(t_1)$ in scenario (b), we have

$$\mathbb{E}\left[\ell | N_{\text{tot}}\right] = N_{\text{tot}} \mathbb{F}_{\text{cap}}(\text{NCLP}). \tag{7.4}$$

Combining equations (7.3) and (7.4) with equation (7.2) we obtain

$$\mathbb{E}\left[\mathcal{L}_a | N_{\text{tot}}\right] \geq N_{\text{tot}} \left(1 - \frac{vW}{L}\right) \mathbb{F}_{\text{cap}}(\text{NCLP}),$$
$$\mathbb{E}\left[\tfrac{\mathcal{L}_a}{N_{\text{tot}}} | N_{\text{tot}}\right] \geq \left(1 - \frac{vW}{L}\right) \mathbb{F}_{\text{cap}}(\text{NCLP}).$$

But $\mathcal{L}_a / N_{\text{tot}}$ is the fraction of outstanding demands in $\mathcal{Q}(t_1)$ that will be captured in scenario (a), and it does not depend on the value of N_{tot}. By the law of total expectation

$$\mathbb{E}\left[\tfrac{\mathcal{L}_a}{N_{\text{tot}}}\right] = \mathbb{E}\left[\mathbb{E}\left[\tfrac{\mathcal{L}_a}{N_{\text{tot}}} | N_{\text{tot}}\right]\right] \geq \left(1 - \frac{vW}{L}\right) \mathbb{F}_{\text{cap}}(\text{NCLP}).$$

Each time the longest path is recomputed, the path in scenario (b) will capture at least this fraction of demands. Thus, we have $\mathbb{F}_{\text{cap}}(\text{LP}) \geq \mathbb{E}\left[\mathcal{L}_a / N_{\text{tot}}\right]$ and have proved the result. □

Remark 7.7 (Conservativeness of bound)**.** *The bound in Theorem 7.6 is conservative. This is primarily due to bounding the expected distance between the causal and non-causal paths by W. The distance between two independently and uniformly distributed points in $[0, W]$, is $W/3$. The distance is even less if the points are positively correlated (as is likely the case for the distance between paths). Thus, it seems that it may be possible to increase the bound to*

$$\mathbb{F}_{\text{cap}}(\text{LP}) \geq \left(1 - \frac{vd}{L}\right) \mathbb{F}_{\text{cap}}(\text{NCLP}),$$

where $d < W/3$. •

The previous theorem establishes the performance of the LONGEST PATH policy relative to a non-causal policy. However, the LP policy is difficult analyze directly. This is due to the fact that the position of the vehicle at time t depends on the positions of all outstanding demands in $\mathcal{Q}(t)$. Thus, our approach is to lower bound the capture fraction of the LP policy with a greedy policy, shown in Algorithm 7.3.

Algorithm 7.3: GREEDY PATH (GP) policy

```
  Assumes: Vehicle is located at $(X, L)$
1 Compute the reachability set $R(X, L)$.
2 Capture the demand in $R(X, L)$ with the highest $y$-coordinate using
  intercept motion.
3 Repeat.
```

Given a set of outstanding demands $\mathcal{Q}(t)$ at time t, the GREEDY PATH policy generates a suboptimal longest path through $\mathcal{Q}(t)$. In addition, the vehicle position is independent of all outstanding demands, except the demand currently being captured. Thus, the capture fraction of the GREEDY PATH policy provides a lower bound for the capture fraction of the LONGEST PATH policy. We are now able to establish the following result.

Theorem 7.8 (Lower Bound for LONGEST PATH policy). *If* $L \geq vW$*, then for the* LONGEST PATH *policy*

$$\mathbb{F}_{\text{cap}}(\text{LP}) \geq \mathbb{F}_{\text{cap}}(\text{GP}) \geq \frac{1}{\sqrt{\pi\omega}\,\text{erf}(\sqrt{\omega}) + \text{e}^{-\omega}},$$

where $\omega = \lambda W/2$ *and* $\text{erf} : \mathbb{R} \to [-1, 1]$ *is the error function.*

Proof. We begin by looking at the expression for the capture fraction. Notice that if $N_{\text{capt}}(t) > 0$ for some $t > 0$, then

$$\begin{aligned} \lim_{t\to+\infty} \mathbb{E}\left[\tfrac{N_{\text{capt}}(t)}{N_{\text{capt}}(t)+N_{\text{esc}}(t)}\right] &= \lim_{t\to+\infty} \mathbb{E}\left[\tfrac{1}{1+\frac{N_{\text{esc}}(t)}{N_{\text{capt}}(t)}}\right] \\ &\geq \left(1 + \lim_{t\to+\infty} \mathbb{E}\left[\tfrac{N_{\text{esc}}(t)}{N_{\text{capt}}(t)}\right]\right)^{-1}, \end{aligned} \tag{7.5}$$

where the last step comes from an application of Jensen's inequality. Thus, we can determine a lower bound on the capture fraction by studying the number of demands that escape per captured demand.

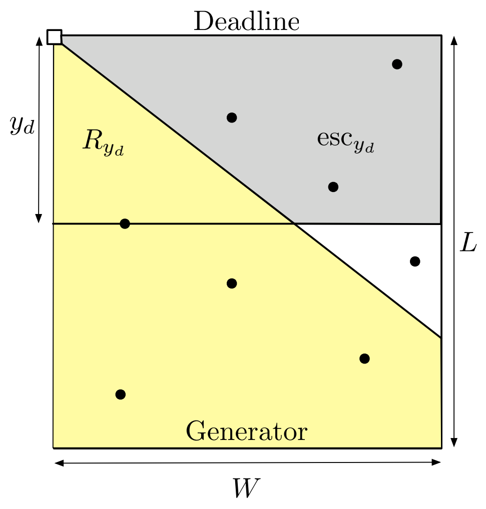


Figure 7.4: The setup for the proof of Theorem 7.8. The service vehicle is located at $(0, L)$. All demands in the region esc_{y_d} escape while capturing the demand with the highest y-coordinate.

Let us study the time instant t at which the service vehicle captures its ith demand, and determine an upper bound on the number of demands that escape before the service vehicle captures its $(i+1)$th demand. Since we seek a lower bound on the capture fraction of the LP policy, we may consider the path generated by the GREEDY PATH policy. In addition, we consider the worst-case service vehicle position; namely, the position $(0, L)$ (or equivalently (W, L)).

From the position $(0, L)$, the reachable set is

$$R(0, L) = \{(x, y) \in \mathcal{E} : vx \leq L\}.$$

Let R_y denote the reachable set intersected with $[0, W] \times [L - y, L]$, where $y \in [0, L]$, and let $|R_y|$ denote its area. Then,

$$|R_y| = \begin{cases} \frac{y^2}{2v}, & \text{if } y \leq vW, \\ yW - \frac{vW^2}{2}, & \text{if } y > vW. \end{cases}$$

An illustration of the set R_y is shown in Figure 7.4. Let y_d be the y-distance to the reachable demand with the highest y-coordinate. That is,

$$y_d = \min_{(x,y) \in \mathcal{Q}(t) \cap R(0,L)} \{L - y\},$$

where $\mathcal{Q}(t)$ is the set of outstanding demands at time t. By Lemma 6.15, the probability that a subset $\mathcal{B} \subset \mathcal{E}$ with area $|\mathcal{B}|$ contains zero demands is given by

$$\mathbb{P}[|\mathcal{B} \cap \mathcal{Q}(t)| = 0] = \mathrm{e}^{-\lambda|\mathcal{B}|/(vW)},$$

where $|\mathcal{B} \cap \mathcal{Q}(t)|$ denotes the cardinality of the finite set $\mathcal{B} \cap \mathcal{Q}(t)$. Thus,

$$\mathbb{P}[y_d > y] = \mathbb{P}[|R_y \cap \mathcal{Q}(t)| = 0] = \mathrm{e}^{-\lambda|R_y|/(vW)}.$$

The probability density function of y_d for $y_d \leq vW$ is

$$\begin{aligned} f(y) = \frac{d}{dy}(1 - \mathbb{P}[y_d > y]) &= \frac{d}{dy}\mathrm{e}^{-\lambda y^2/(2v^2W)} \\ &= \frac{\lambda}{v^2W} y \mathrm{e}^{-\lambda y^2/(2v^2W)}. \end{aligned}$$

Now, given y_d, all demands residing in the region $\mathrm{esc}_{y_d} := ([0, W] \times [L - y_d, L]) \setminus R_{y_d}$ will escape (see Fig. 7.4). The area of esc_{y_d} is

$$|\mathrm{esc}_{y_d}| = \begin{cases} y_dW - \frac{y_d^2}{2v}, & \text{if } y_d \leq vW, \\ \frac{vW^2}{2}, & \text{if } y_d \geq vW. \end{cases}$$

From Lemma 6.15 and Remark 6.16 in Chapter 6, the expected number of outstanding demands in an unserviced region of area A is $\lambda A/(vW)$. Thus, given that the vehicle is located at $(0, L)$, the expected number of demands that escape while the service vehicle is capturing its $(i + 1)$th demand is given by

$$\begin{aligned} \mathbb{E}[N_{\mathrm{esc},i}] &= \frac{\lambda}{vW}\mathbb{E}[|\mathrm{esc}_{y_d}|] \\ &= \frac{\lambda}{vW}\left[\int_0^{vW}\left(yW - \frac{y^2}{2v}\right) f(y)dy + \frac{vW^2}{2}\mathbb{P}[y_d > vW]\right]. \end{aligned}$$

Applying the probability density function and cumulative distribution function of y_d we obtain

$$\mathbb{E}[N_{\mathrm{esc},i}] = \frac{\lambda^2}{v^3W^2}\int_0^{vW}\left(yW - \frac{y^2}{2v}\right) y\mathrm{e}^{-\lambda y^2/(2v^2W)}dy + \frac{\lambda W}{2}\mathrm{e}^{-\lambda W/2}. \tag{7.6}$$

To evaluate the integral, consider the change of coordinates $z := y/vW$, and define $\omega := \lambda W/2$. After simplifying, the integral becomes

$$4\omega^2\int_0^1\left(z^2 - \frac{z^3}{2}\right)\mathrm{e}^{-\omega z^2}dz.$$

Integrating by parts we obtain

$$\sqrt{\pi\omega}\,\mathrm{erf}(\sqrt{\omega}) + \omega \mathrm{e}^{-\omega} + \mathrm{e}^{-\omega} - 1, \tag{7.7}$$

where $\mathrm{erf} : \mathbb{R} \to [-1, 1]$ is the error function:

$$\mathrm{erf}(x) = \frac{2}{\pi} \int_0^x \mathrm{e}^{-t^2} dt.$$

Substituting equation (7.7) into equation (7.6) we obtain

$$\mathbb{E}\left[N_{\mathrm{esc},i}\right] = \sqrt{\pi\omega}\,\mathrm{erf}(\sqrt{\omega}) + \mathrm{e}^{-\omega} - 1.$$

Since $\mathbb{E}\left[N_{\mathrm{esc},i}\right]$ is computed for the worst-case vehicle position $(0, L)$, and since this expression holds at every capture, we have that

$$\lim_{t\to+\infty} \mathbb{E}\left[\tfrac{N_{\mathrm{esc}}(t)}{N_{\mathrm{capt}}(t)}\right] \leq \sqrt{\pi\omega}\,\mathrm{erf}(\sqrt{\omega}) + \mathrm{e}^{-\omega} - 1,$$

and thus by equation (7.5) we obtain the desired result. □

7.3 Demand speed less than vehicle speed

In this section we study the case when the demand speed $v < 1$. For this case, an upper bound on the capture fraction has been derived in in Remark 6.18. We introduce a policy which is a variant of the TMHP-based policy in Chapter 6, and lower bound its capture fraction in the limit of low demand speed and high demand arrival rate.

7.3.1 Capture Fraction Upper Bound

The following theorem upper bounds the capture fraction of every policy for the case of $v < 1$, and follows from Remark 6.18 in Chapter 6.

Theorem 7.9 (Capture fraction upper bound). *If $v < 1$, then for every causal policy P*

$$\mathbb{F}_{\mathrm{cap}}(P) \leq \min\left\{1, \frac{2}{\sqrt{v\lambda W}}\right\}.$$

Notice that for low demands speed, i.e., $v \ll 1$, it may be possible to achieve a capture fraction of one, even for high arrival rates.

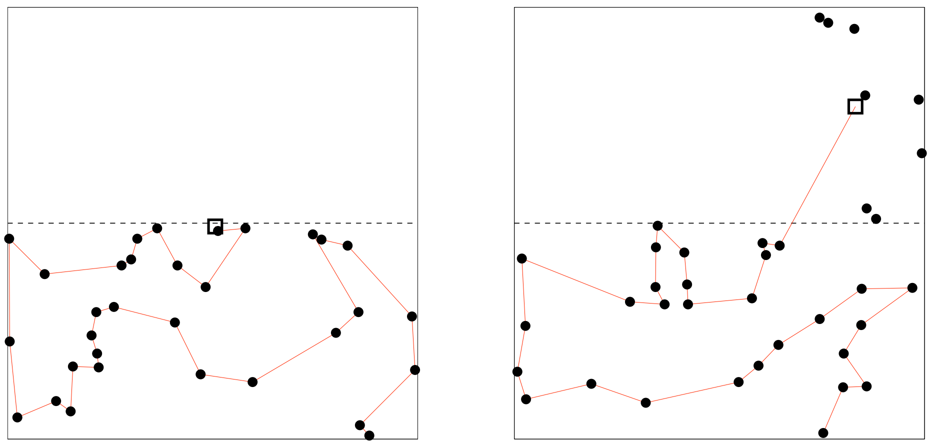


Figure 7.5: The TMHP-FRACTION policy. The left-hand figure shows a TMHP through all outstanding demands. The right-figure shows the instant when the vehicle has followed the path for $L/(2v)$ time units and recomputes its path, allowing some demands to escape.

7.3.2 The TMHP-fraction Policy

In Section 2.2.3 we reviewed the translational minimum Hamiltonian Path (TMHP) through a set of demands. The TMHP-FRACTION policy, shown in Algorithm 7.4, utilizes this path to service demands.

Algorithm 7.4: TMHP-FRACTION (TF) policy

```
  Assumes: Vehicle is located on the line $y = L/2$.
1 Compute a translational minimum Hamiltonian path through all
  outstanding demands in $[0, W] \times [0, L/2]$, starting at the service vehicle
  position, and terminating at the demand with the lowest $y$-coordinate.
2 if time to travel entire path is less than $L/(2v)$ then
3     Service all outstanding demands by following the computed path.
4 else
5     Service outstanding demands along the computed path for $L/(2v)$ time
      units.
6 Repeat.
```

Figure 7.5 shows an example of the TMHP-FRACTION policy. In contrast with the LP policy, where the vehicle remains on the deadline, in the TMHP-FRACTION policy the vehicle follows the TMHP using minimum time motion between demands as described in Section 2.2.3. Notice that none of the demands in the region $[0, W] \times [0, L/2]$ at time t will have escaped before time $t + L/(2v)$.

Thus, the vehicle is guaranteed that for the first $L/(2v)$ time units, all demands on the TMHP path are still in the environment. For the TMHP-Fraction policy we have the following result.

Theorem 7.10 (TMHP-Fraction policy lower bound)**.** *In the limit as $v \to 0^+$ and $\lambda \to +\infty$, the capture fraction of the* TMHP-Fraction *policy satisfies*

$$\mathbb{F}_{\text{cap}}(\text{TF}) \geq \min\left\{1, \frac{1}{\beta_{\text{TSP}}\sqrt{v\lambda W}}\right\}.$$

Proof. Consider the beginning of an iteration of the policy, and assume that the duration of the previous iteration was $L/(2v)$. In this case, the vehicle has y-coordinate $Y \in [L/2, L]$, and by Lemma 6.15, the region $\mathcal{R} := [0, W] \times [0, L/2]$ contains a number of demands N that is Poisson distributed with parameter $\lambda L/(2v)$. Conditioned on N, the demands are independently and uniformly distributed in $\mathcal{R}$.

Now, we make use of the following three facts. First, as $v \to 0^+$, the length of the TMHP constrained to start at the vehicle location and end at the lowest demand, is equal to the length of the EMHP in the corresponding static instance, as described in Lemma 2.4. Second, from Section 2.2.2, for uniformly distributed points, the asymptotic length of a constrained EMHP is equal to the asymptotic length of the ETSP tour. Third, as $v \to 0^+$, and $\lambda \to +\infty$, we have that N tends to $+\infty$ with probability one. Using the above facts we obtain that the length of the TMHP starting at the vehicle position, passing through all demands in $\mathcal{R}$, and terminating at the demand with the lowest y-coordinate, has length $\beta_{\text{TSP}}\sqrt{NWL/2}$ in the limiting regime as $v \to 0^+$, and $\lambda \to +\infty$.

The vehicle will follow the TMHP for at most $L/(2v)$ time units, and thus will service ξN demands, where

$$\xi = \min\left\{1, \frac{\sqrt{L}}{\beta_{\text{TSP}} v\sqrt{2NW}}\right\}.$$

Now, the random variable N has expected value $\mathbb{E}[N] = \lambda L/(2v)$ and variance $\sigma_N^2 = \lambda L/(2v)$. By the Chebyshev inequality, $\mathbb{P}[|N - \mathbb{E}[N]| \geq a] \leq \sigma_N^2/a^2$, and thus letting $a = \sqrt{v}\mathbb{E}[N]$, we have

$$\mathbb{P}[N \geq (1+\sqrt{v})\mathbb{E}[N]] \leq \frac{1}{v\mathbb{E}[N]} = \frac{2}{\lambda L}.$$

Thus, we have

$$\xi \geq \min\left\{1, \frac{1}{\beta_{\text{TSP}}\sqrt{(1+\sqrt{v})v\lambda W}}\right\}.$$

with probability at least $1 - 2/(\lambda L)$. In the limit as $\lambda \to +\infty$, with probability 1,

$$\xi \geq \min\left\{1, \frac{1}{\beta_{\mathrm{TSP}}\sqrt{v\lambda W}}\right\}. \tag{7.8}$$

Therefore, if the previous iteration had duration at least $L/(2v)$, then the total fraction of demands captured in the current iteration is given by equation (7.8).

The other case is that the previous iteration had duration $T < L/(2v)$. In this case, all outstanding demands in the region $\mathcal{R} := [0, W] \times [0, L/2]$ lie in a subset $[0, W] \times [0, vT]$, and the subset contains a number of demands N that is Poisson distributed with parameter $\lambda T \leq \lambda L/(2v)$. Thus, in this case there are fewer outstanding demands, and the bound on ξ still holds. Thus, $\mathbb{F}_{\mathrm{cap}}(\mathrm{TF}) \geq \xi$, and we obtain the desired result. □

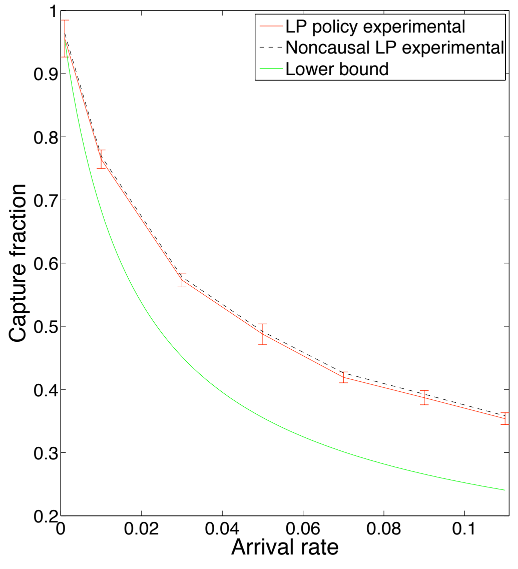
Remark 7.11 (Bound comparison)**.** *In the limit as $v \to 0^+$, and $\lambda \to +\infty$, the capture fraction of the* TMHP-Fraction *policy is within a factor of $2\beta_{\mathrm{TSP}} \approx 1.42$ of the optimal.* •

7.4 Simulations

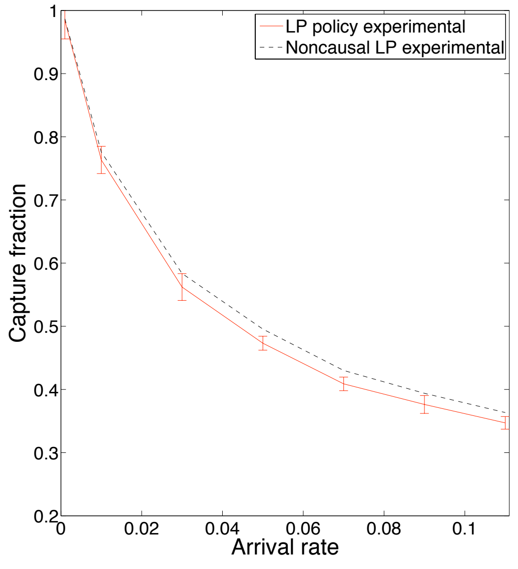
We now present two sets of results from numerical experiments. The first set compares the Longest Path policy with $\eta = 1$ to the Non-causal Longest Path policy and to the theoretical lower bound in Theorem 7.8. The second set compares the TMHP-Fraction policy to the policy independent upper bound in Theorem 7.9 and the lower bound in Theorem 7.10.

To simulate the LP and the NCLP policies, we perform 10 runs of the policy, where each run consists of 5000 demands. A comparison of the capture fractions for the two policies is presented in Figure 7.6. When $L > vW$, the capture fraction of the LP policy is nearly identical to that of the NCLP policy. Even in Figure 7.6(a), where $L < vW$, the capture fraction of the LP policy is within 2% of the NCLP policy, and thus the optimal. This suggests that the Longest Path policy is essentially optimal over a large range of parameter values.

To simulate the TMHP-Fraction policy, the `linkern` solver is used to generate approximations to the optimal TMHP (see Chapter 2 for a review on generating approximate solutions to the TMHP). For each value of arrival rate, we determine the capture fraction by taking the mean over 10 runs of the policy. A comparison of the simulation results with the theoretical results from Section 7.3 are presented in Figure 7.7. For $v = 0.01$ in Fig. 7.7(a), the experimental results are in near exact agreement with the theoretical lower bound in Theorem 7.9. For $v = 0.05$ in Fig. 7.7(b), the experimental results are within 5% of the theoretical lower bound. However, notice that the experimental capture fraction is smaller

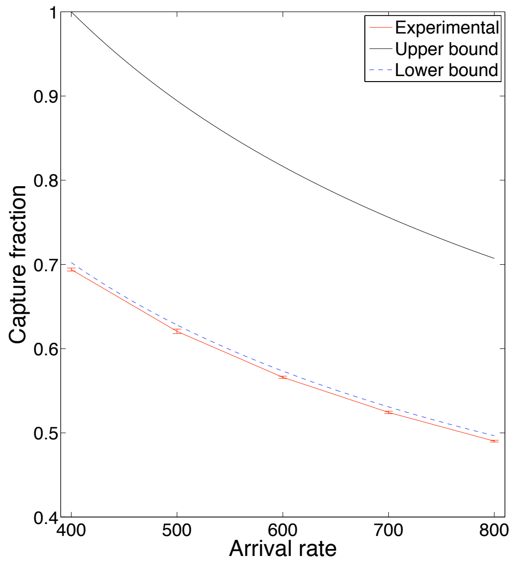


(a) $v = 2$ and $L > vW$.

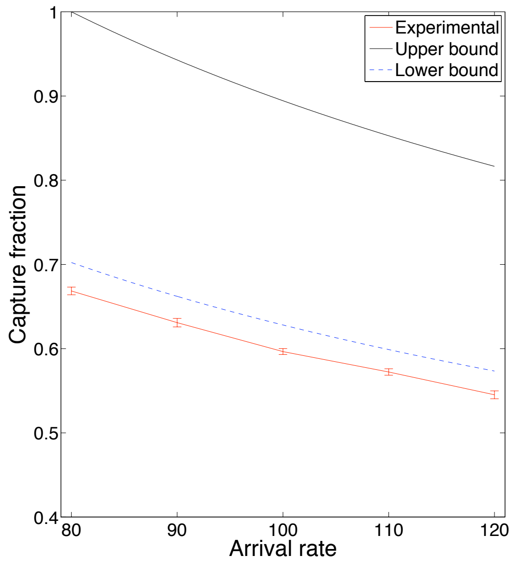


(b) $v = 5$ and $L < vW$.

Figure 7.6: Simulation results for LP policy (solid red line with error bars showing $\pm$ one standard deviation) and the NCLP policy (dashed black line) for an environment of width $W = 100$ and length $L = 500$. In (a), $L > vW$, and the lower bound in Theorem 7.8 is shown in solid green.



(a) Demand speed $v = 0.01$.



(b) Demand speed $v = 0.05$.

Figure 7.7: Simulation results for TMHP-Fraction policy. The solid black curve shows the upper bound in Theorem 7.9 and the dashed line shows the lower bound in Theorem 7.10. Numerical results are shown with error bars.

than the theoretical lower bound. This is due to several factors. First, we have not reached the limit as $v \to 0^+$ and $\lambda \to +\infty$ where the asymptotic value of $\beta_{\text{TSP}} \approx 0.712$ holds. Second, we are using an approximate solution to the optimal TMHP, generated via the linkern solver.

7.5 Summary

In this chapter we introduced a dynamic vehicle routing problem in which a vehicle must defend a deadline from approaching demands. We presented novel policies in the case when the demand speed is greater than the vehicle speed, and in the case when the demand speed is less than the vehicle speed. In the former case we introduced the Longest Path policy which is based on computing longest paths in the directed acyclic reachability graph, and in the latter case we introduced the TMHP-Fraction policy. For each policy, we analyzed the fraction of demands that are captured. It appears that for the Longest Path policy we may be able to extend our results to the case of $L < vW$, and tighten our existing bounds to reflect the near optimal performance shown in simulation.

Chapter Eight

Conclusions

There are many potential applications for large groups of autonomous vehicles, capable of performing complex tasks in unknown an dynamic environments. Examples include environmental monitoring, surveillance, reconnaissance, search and rescue mission, infrastructure monitoring and security, and assisted living. This thesis has focused on two enabling technologies for such applications; distributed task allocation, and dynamic vehicle routing. In each chapter we have considered a different problem within task allocation and vehicle routing. Our technical approach to each problem has followed the same basic plan. First, for each problem we have identified **underlying problem structure**. This typically consists of intrinsic regimes in the space of problem parameters. Second, by leveraging the problem structure, we have determined **fundamental limits on the achievable performance** of any algorithm for solving the problem of interest. Third and finally, by utilizing the fundamental limits and problem structure, we have designed **provably efficient algorithms**. Generally, we have designed algorithms that are guaranteed to perform within a constant factor of the optimal algorithm.

8.1 Summary

In Chapter 3 we considered a task allocation problem called the target assignment problem. We considered two scenarios: (1) each vehicle is given a list containing all target positions (the positions may be given as GPS coordinates); or (2) each vehicle has no initial target information, but has a finite-range target sensor to acquire target positions. For each scenario we studied the scalability of the completion time of target assignment as a function of the number of vehicles. We developed algorithms for each scenario which exhibit worst-case performance within a constant factor of the optimal. In addition, we studied the performance of our target assignment algorithms for stochastic initial conditions.

In Chapters 4, 5, 6, and 7 we studied several problems in the area of dynamic vehicle routing. In Chapter 4 we looked at dynamic vehicle routing problem which consists of a heterogeneous group of vehicles, and service demands that require teams of vehicles for their completion. We developed three unbiased algorithms, each of which has large parameter regimes for which its performance is within a constant factor of the optimal.

In Chapter 5 we looked at a dynamic vehicle routing problem in which vehicles are allowed to bias their service towards certain demands. In particular, we considered a problem in which each demand is a member of one of m priority classes, and the vehicle must minimize a convex combination of the expected delay (i.e., quality of service) of each class. We determined a fundamental limit on the performance of any algorithm, and developed an algorithm that performs within a known constant factor of the optimal in both heavy load and light load conditions.

In Chapter 6 we considered a dynamic vehicle routing problem in which service demands are generated on a line segment, and move perpendicular to the segment at a fixed speed. We determined fundamental limits on the stability region of any policy, and developed a policy whose stability region is optimal or near-optimal in several limiting regimes. Finally, in Chapter 7 we considered a variation of the problem studied in Chapter 6, wherein the demands must be serviced before they reach a deadline. The goal was to maximize the fraction of demands that are serviced before they reach the deadline. We developed two policies for this problem, each of which performs within a constant factor of the optimal for large problem parameter regimes.

8.2 Future Directions

In this thesis we have looked at problems in task allocation and dynamic vehicle routing. For several different problems we have determined bounds on the best achievable performance. In addition, we have designed algorithms that perform well when compared to the bounds. However, while this research has provided answers to many questions in the areas of task allocation and vehicle routing, it has raised many new questions. A few of these will be outlined here.

Realistic environments and vehicle dynamics: Throughout this thesis we have modeled the vehicles as omnidirectional first-order integrators. In addition, we have modeled the environments as convex, or in many cases square regions. It would be very interesting to extend our work to the case of vehicles with motion constraints, such as the Dubins vehicle or double integrator model. There has recently been work by Savla et al. (2009), Savla, Frazzoli and Bullo (2008), and Le Ny et al. (2009) which provide bounds on the length of TSP tours for vehicles with motion constraints, and algorithms for computing approximate solutions. To extend to more realistic environments it would be important to look at general non-convex regions with obstacles. To tackle these complex environments, one would need to incorporate some ideas of robotic motion planning (see LaValle, 2006).

Moving demands: In Chapters 6 and 7, we looked at problems where demands (or targets) moved in straight lines at constant speed. The arrival of demands was Poisson in time, and uniform in space along a generating line segment. A first extension would be to demands with a non-uniform spatial arrival density and to multiple service vehicles. The next step would be to look at demands with velocity vectors selected according to a distribution (i.e., demands have different speeds and headings). Also, we could consider adversarial arrivals, motion, and even cooperation between demands. These considerations would start to bring in aspects of game theory (Başar and Olsder, 1999) and pursuit evasion games (Isaacs, 1965). Another interesting aspect would be to introduce on-site service times. For the case in which the on-site service times are independent and identically distributed with a known expected value, and the vehicle is permitted to move with the demand upon reaching it, the results of Chapter 6 appear to directly extend. Finally, it appears that the Longest Path policy has promising extensions to the cases where demands have different priority levels, arbitrary motion, and where the vehicle has a finite capture radius.

Team forming and biased policies: In Chapter 4 we performed analysis and developed algorithms for unbiased team forming. The term *unbiased* referred to the constraint that policies must provide the same quality of service to every type of demand. That is, a team vehicle was not allowed to give preferential treatment to a certain class of demands. In Chapter 5 we considered the exact opposite, and looked at a problem where a vehicle attempts to provide a high quality of service to "high priority demands" will still providing service to "low priority demands." We captured the concept of high and low priority demands by asking the service vehicles to minimize a convex combination of the delays of priority classes. Thus, a natural question is "Can we use the priority policy developed in Chapter 5 to develop biased policies for the dynamic team forming problem of Chapter 4?" It appears that the priority policy provides some answers, but there is still a significant amount of work to be done in order to provide provably efficient biased team forming policies. Efficient algorithms will require advanced scheduling technique for answering questions about when vehicles should meet, how often, and where.

Other interesting extensions include relaxing other assumptions in the analysis of team forming policies (see Section 4.4.1). And considering vehicles with different speeds and dynamics.

Exploration vs. exploitation In the problems we have considered we generally assume that the vehicles have full information on the demands (the one major exception to this is the sensor-based target assignment problem in Chapter 3. An interesting direction is to look at dynamic vehicle routing problems in which vehicles have some (possibly local) information on demands, but must perform searches to acquire new demand information. When considering a problem of this form, it brings into play a classic trade-off of exploitation versus exploration (see Sutton and Barto, 1998; Russell and Norvig, 2003): Should a vehicle exploit its current knowledge of demand locations and provide service to them, or should it explore in order to find new demand locations? To search for new demands, vehicles might use techniques such as milling and lawn mowing (Arkin et al., 2000), which provide efficient environment sweeping methods.

Cooperative load balancing In this thesis we have essentially looked at two methods for task allocation. The first method was target assignment (Chapter 3), where each vehicle was explicitly assigned to a single target. The second method of task allocation, utilized in Chapters 4 and 5, involved environment partitioning. In this second method the assignment was implicit in that each vehicle had a single region of responsibility, and all demands (or tasks) which arrived in a given region were the responsibility of the corresponding vehicle. This method of allocation works well in heavy load conditions, where the rate of demand arrivals is high, or in light load conditions, where the rate of demand arrivals is very low. However, in the intermediate regime, the performance of partitioning is not fully characterized.

Based on this discussion, an interesting direction for future research is to look into new methods of task allocation, which possibly combine allocation-by-partitioning with explicit allocations to improve performance.

Appendix A

Formal Description the ETSP Assgmt Algorithm

This appendix formally describes the ETSP Assgmt algorithm presented in Chapter 3.

Algorithm A.1: Initialization of vehicle i in ETSP Assgmt.

```
  Assumes: Vehicle $i$ has the target array $\mathbf{q}^{[i]} := \mathbf{q}$, stored in its memory.
1 Compute a TSP tour of $\mathbf{q}^{[i]}$, $\mathrm{tour}(\mathbf{q}^{[i]})$, and set $\mathbf{q}^{[i]} := \mathrm{tour}(\mathbf{q}^{[i]})$.
2 Compute the closest target in $\mathbf{q}^{[i]}$, and set $\mathrm{curr}^{[i]}$ equal to its index:
  $\mathrm{curr}^{[i]} := \arg\min_{j\in\mathcal{I}}\{\|\mathbf{q}_j^{[i]} - \mathbf{p}^{[i]}\|\}$.
3 Set $\mathrm{next}^{[i]} := \mathrm{curr}^{[i]} + 1 \pmod n$.
4 Set $\mathrm{prev}^{[i]} := \mathrm{curr}^{[i]} - 1 \pmod n$.
5 Set $\mathrm{status}^{[i]} := \mathbf{1}_n$ (i.e., an $n$-tuple containing $n$ ones).
```

Algorithm A.2: COMM-RD, executed at each communication round.

```
Assumes: Vehicle $i$ has been initialized as in Algorithm A.1.
1  Compute $\mathrm{dist}^{[i]} := \|\mathbf{p}^{[i]} - \mathbf{q}^{[i]}_{\mathrm{curr}^{[i]}}\|$.
2  Broadcast $\mathrm{msg}^{[i]} := (\mathrm{prev}^{[i]}, \mathrm{curr}^{[i]}, \mathrm{next}^{[i]}, i, \mathrm{dist}^{[i]})$
3  Receive $\mathrm{msg}^{[k]}$, from each $k \neq i$ satisfying $\|\mathbf{p}^{[i]} - \mathbf{p}^{[k]}\| \leq r_{\mathrm{comm}}$.
4  foreach $\mathrm{msg}^{[k]}$ received do
5      for $s = \mathrm{prev}^{[k]} + 1$ to $\mathrm{next}^{[k]} - 1 \pmod{n}$ do
6          if $s \neq \mathrm{curr}^{[i]}$ then Set $\mathrm{status}^{[i]}(s) := 0$
7      if $\mathrm{prev}^{[k]} = \mathrm{next}^{[k]} = \mathrm{curr}^{[k]} \neq \mathrm{curr}^{[i]}$ then Set $\mathrm{status}^{[i]}(\mathrm{curr}^{[k]}) := 0$
8      if $\mathrm{curr}^{[i]} = \mathrm{curr}^{[k]}$ then
9          if $(\mathrm{dist}^{[i]} > \mathrm{dist}^{[k]})$ OR $(\mathrm{dist}^{[i]} = \mathrm{dist}^{[k]}$ AND $i < k)$ then
10             Set $\mathrm{status}^{[i]}(\mathrm{curr}^{[i]}) := 0$.
11         if $\mathrm{next}^{[i]} \neq \mathrm{curr}^{[i]}$ then Set $\mathrm{status}^{[i]}(\mathrm{next}^{[i]}) := 0$.
12         if $\mathrm{next}^{[k]} \neq \mathrm{curr}^{[i]}$ then Set $\mathrm{status}^{[i]}(\mathrm{next}^{[k]}) := 0$.
13 if $\mathrm{status}^{[i]}(j) = 0$ for every target $j$ then Exit ETSP ASSGMT and stop
   motion.
14 while $\mathrm{status}^{[i]}(\mathrm{curr}^{[i]}) = 0$ do $\mathrm{curr}^{[i]} := \mathrm{curr}^{[i]} + 1 \pmod{n}$.
15 Set $\mathrm{next}^{[i]} := \mathrm{curr}^{[i]} + 1 \pmod{n}$.
16 while $\mathrm{status}^{[i]}(\mathrm{next}^{[i]}) = 0$ do $\mathrm{next}^{[i]} := \mathrm{next}^{[i]} + 1 \pmod{n}$.
17 while $\mathrm{status}^{[i]}(\mathrm{prev}^{[i]}) = 0$ do $\mathrm{prev}^{[i]} := \mathrm{prev}^{[i]} - 1 \pmod{n}$.
```

Appendix B

Formal Description of the Grid Assgmt Algorithm

This appendix gives a formal description of the Grid Assgmt algorithms presented in Chapter 3. As noted in Remark 3.17, we have simplified the presentation of the Unassigned algorithm by assuming that every cell initially contains at least one vehicle and one target. It is straightforward to relax this assumption. If a cell has no then any vehicles initially in the cell move to the cell below, and the empty cell is ignored for the rest of the algorithm. If there is a cell that contains targets but no vehicles, then the first vehicles to enter the cell run the Role Assgmt algorithm and one becomes the leader.

Table B.1: Variables for the Grid Assgmt algorithm.

Vehicle role	**Variable**	**Description**	**Value**
all	$\text{currcell}^{[i]}$	cell currently occupied by vehicle i	a cell $C(r,c)$
	$\text{leader}^{[i]}$	cell for which vehicle i is leader	a cell $C(r,c)$ or `null`
	$\text{curr}^{[i]}$	vehicle i's assigned target	a target in $\mathcal{Q}$ or `null`
unassigned	$\text{dircol}^{[i]}$	travel direction in column c	`up` or `down`
	$\text{dirrow}^{[i]}$	travel direction when in row 1	`left` or `right`
	$\text{colstatus}^{[i]}(c)$	records whether or not there are free targets in column c	`full` or `notfull`
	$\text{prevcell}^{[i]}$	previous cell occupied by vehicle i	a cell $C(r,c)$
$C(r,c)$ leader	$\text{taravail}^{[i]}(r,c)$	set of available targets in $C(r,c)$	a subset of $\mathcal{Q}$
	$\Delta^{[i]}(r,c)$	(# of targets) − (# of vehicles) in $C(r,c)$	an integer
	$\Delta^{[i]}_{\text{blw}}(r,c)$	estimate of (# of targets) − (# of vehicles) in $C(r+1,c)$ to $C(b,c)$	an integer or $+\infty$
$C(1,c)$ leader	$\Delta^{[i]}_{\text{rght}}(1,c)$	estimate of (# of targets) − (# of vehicles) in columns $c+1$ to b	an integer or $+\infty$

Algorithm B.1: Role Assgmt, executed at the start of the Grid Assgmt algorithm to assign roles, and initialize vehicle i.

```
   Assumes: Vehicle $i$ is in $C(r,c)$, knows $\mathcal{E}(n)$, and either (1) knows all
            target positions, or (2) has $r_{\text{sense}} \geq \sqrt{2/5} r_{\text{comm}}$.
1  Compute $b$ as in Lemma 3.2, partition $\mathcal{E}(n)$ into the $b^2$ square cells.
2  Set currcell$^{[i]}$ := $C(r,c)$, leader$^{[i]}$ := null and curr$^{[i]}$ := null.
3  Broadcast msg$^{[i]}$ containing UID$^{[i]}$, $\mathbf{p}^{[i]}$, and currcell$^{[i]}$ to vehicles in
   currcell$^{[i]}$.
4  Receive msg$^{[k]}$ from, all vehicles in $C(r,c)$.
5  Use the Maximal Match algorithm to find a maximum matching
   between vehicles and targets $C(r,c)$.
6  Elect a leader among assigned vehicles in $C(r,c)$.
7  case unassigned
8      Set dircol$^{[i]}$ := down, dirrow$^{[i]}$ := right, and colstatus$^{[i]}(c)$ to notfull
       for each $c \in \{1,\dots,b\}$.
9      Run Unassigned algorithm.
10 case assigned to $\mathbf{q} \in \mathcal{C}(r,c)$ and not elected leader
11     Set curr$^{[i]}$ := $\mathbf{q}$, and move to curr$^{[i]}$ at speed $v_{\max}$
12 case assigned and elected leader
13     Set leader$^{[i]}$ := currcell$^{[i]}$, curr$^{[i]}$ := $\mathbf{q}$, and move to curr$^{[i]}$ at speed $v_{\max}$
14     Set $\Delta^{[i]}(r,c)$ to number of targets in $C(r,c)$ minus number of vehicles in
       $C(r,c)$.
15     Set taravail$^{[i]}(r,c)$ to the collection of unassigned targets in $C(r,c)$.
16     Set $\Delta^{[i]}_{\text{blw}}(r,c)$ to $+\infty$ if $r \in \{1,\dots,b-1\}$ and to 0 if $r = b$.
17     if $r = 1$ then Set $\Delta^{[i]}_{\text{rght}}(c)$ to $+\infty$ if $c \in \{1,\dots,b-1\}$ and to 0 if $c = b$.
18     Run Leader algorithm.
```

Algorithm B.2: Leader, executed at each communication round.

```
   Assumes: Vehicle $i$ is the leader of $C(r,c)$.
1  Send $\mathrm{msg}_1^{[i]} := \Delta_{\mathrm{blw}}^{[i]}(r,c) + \Delta^{[i]}(r,c)$ to leader in cell $C(r-1,c)$.
2  if $r = 1$ then
3      Send $\mathrm{msg}_2^{[i]} := \Delta_{\mathrm{rght}}^{[i]}(c) + \Delta_{\mathrm{blw}}^{[i]}(1,c) + \Delta^{[i]}(1,c)$ to leader of $C(1,c-1)$.
4      Receive $\mathrm{msg}_2^{[k]}$ from leader $k$ of $C(1,c+1)$ and set $\Delta_{\mathrm{rght}}^{[i]}(c) := \mathrm{msg}_2^{[k]}$.
5      For each enter msg from an vehicle coming from $C(1,c+1)$, add 1 to
       $\Delta_{\mathrm{rght}}^{[i]}(1,c)$ and for each exit msg from an vehicle going to $C(1,c+1)$
       subtract 1 from $\Delta_{\mathrm{rght}}^{[i]}(1,c)$.
6      If $\Delta_{\mathrm{rght}}^{[i]}(1,c) > 0$ and an enter msg was received from an vehicle
       coming from $C(1,c+1)$, then set $\Delta_{\mathrm{rght}}^{[i]}(1,c) := 0$.
7  Receive $\mathrm{msg}_1^{[k]}$ from leader $k$ of $C(r+1,c)$, and set $\Delta_{\mathrm{blw}}^{[i]}(r,c) := \mathrm{msg}_1^{[k]}$.
8  Subtract 1 from $\Delta^{[i]}(r,c)$ for each enter msg received, and add 1 for each
   exit msg received.
9  For each enter msg from an vehicle coming from $C(r+1,c)$, add 1 to
   $\Delta_{\mathrm{blw}}^{[i]}(r,c)$ and for each exit msg from an vehicle going to $C(r+1,c)$
   subtract 1 from $\Delta_{\mathrm{blw}}^{[i]}(r,c)$.
10 If $\Delta_{\mathrm{blw}}^{[i]}(r,c) > 0$ and an enter msg was received from an vehicle coming
   from $C(r+1,c)$, then set $\Delta_{\mathrm{blw}}^{[i]}(r,c) := 0$.
11 forall queries on availability of target in $C(r,c)$ do
12     if $\mathrm{taravail}^{[i]} \neq \emptyset$ then
13         Select a target in $\mathrm{taravail}^{[i]}$, assign it requesting vehicle, and remove
           it from $\mathrm{taravail}^{[i]}$.
14     else if $\mathrm{taravail} = \emptyset$ then Reply no.
15 forall queries on availability of target below $C(r,c)$ do
16     Respond yes to $\Delta_{\mathrm{blw}}^{[i]}(r,c)$ requests, and no to all others.
17 if $r = 1$ then
18     forall queries on availability of target to right of column $c$ do
19         Respond yes to $\Delta_{\mathrm{rght}}^{[i]}(c)$ requests, and no to all others.
```

Algorithm B.3: Unassigned, executed each time a new cell is entered.

```
   Assumes: Vehicle $i$ has run Role Assgmt, and $\text{currcell}^{[i]} = C(r,c)$.
1  Query leader of $C(r,c)$ on free targets in $\text{currcell}^{[i]}$.
2  if leader returns a target $\mathbf{q} \in C(r,c)$ then Set $\text{curr}^{[i]} := \mathbf{q}$, and move to
   target.
3  else if leader returns no then
4      case $\text{dircol}^{[i]} = \texttt{down}$
5          Query leader on availability of target below $C(r,c)$.
6          if leader returns yes then
7              Set $\text{prevcell}^{[i]} := \text{currcell}^{[i]}$ and $\text{currcell}^{[i]} := C(r+1,c)$
8          else if leader returns no then
9              Set $\text{dircol}^{[i]} := \texttt{up}$, $\text{prevcell}^{[i]} := \text{currcell}^{[i]}$ and
               $\text{currcell}^{[i]} := C(r-1,c)$.
10     case $(\text{dircol}^{[i]} = \texttt{up})$ and $(r > 1)$
11         Set $\text{prevcell}^{[i]} := \text{currcell}^{[i]}$, $\text{currcell}^{[i]} := C(r-1,c)$, and
           $\text{dircol}^{[i]} := \texttt{up}$
12     case $(\text{dircol}^{[i]} = \texttt{up})$ and $(r = 1)$ and $(\text{dirrow}^{[i]} = \texttt{right})$
13         Set $\text{colstatus}^{[i]}(c) := \texttt{full}$.
14         Query leader on availability to the right of column $c$.
15         if leader returns yes then
16             Set $\text{prevcell}^{[i]} := \text{currcell}^{[i]}$, $\text{currcell}^{[i]} := C(1,c+1)$
17             if $\text{colstatus}^{[i]}(c+1) = \texttt{notfull}$ then $\text{dircol}^{[i]} := \texttt{down}$.
18         else if leader returns no then
19             Set $\text{prevcell}^{[i]} := \text{currcell}^{[i]}$, $\text{currcell}^{[i]} := C(1,c-1)$,
               $\text{dirrow}^{[i]} := \texttt{left}$.
20             Set $\text{colstatus}^{[i]}(c^*) := \texttt{full}$ for each $c^* \in \{c+1,\dots,b\}$.
21             if $\text{colstatus}^{[i]}(c-1) = \texttt{notfull}$ then $\text{dircol}^{[i]} := \texttt{down}$.
22     case $(\text{dircol}^{[i]} = \texttt{up})$ and $(r = 1)$ and $(\text{dirrow}^{[i]} = \texttt{left})$
23         Set $\text{prevcell}^{[i]} := \text{currcell}^{[i]}$ and $\text{currcell}^{[i]} := C(1,c-1)$.
24         if $\text{colstatus}^{[i]}(c-1) = \texttt{notfull}$ then $\text{dircol}^{[i]} := \texttt{down}$.
25     Send exit to leader in $\text{prevcell}^{[i]}$, enter to leader in $\text{currcell}^{[i]}$, and
       move to $\text{currcell}^{[i]}$.
```

Bibliography

Alighanbari, M. and How, J. P. (2008), 'A robust approach to the UAV task assignment problem', *International Journal on Robust and Nonlinear Control* **18**(2), 118–134.

Applegate, D. L., Bixby, R. E. and Chvátal, V. (2006), *The Traveling Salesman Problem: A Computational Study*, Applied Mathematics Series, Princeton University Press.

Arkin, E. M., Fekete, S. P. and Mitchell, J. S. B. (2000), 'Approximation algorithms for lawn mowing and milling', *Computational Geometry - Theory and Applications* **17**(1-2), 25–50.

Arora, S. (1998), 'Polynomial-time approximation schemes for the Euclidean TSP and other geometric problems', *Journal of the ACM* **45**(5), 753–782.

Arslan, G., Marden, J. R. and Shamma, J. S. (2007), 'Autonomous vehicle-target assignment: A game theoretic formulation', *ASME Journal on Dynamic Systems, Measurement, and Control* **129**(5), 584–596.

Asahiro, Y., Miyano, E. and Shimoirisa, S. (2008), 'Grasp and delivery for moving objects on broken lines', *Theory of Computing Systems* **42**(3), 289–305.

Bailey, N. T. J. (1954), 'On queueing processes with bulk service', *Journal of the Royal Statistical Society. Series B* **16**(1), 80–87.

Bansal, N., Blum, A., Chawla, S. and Meyerson, A. (2004), Approximation algorithms for deadline-TSP and vehicle routing with time-windows, *in* 'ACM Symposium on the Theory of Computing', Chicago, IL, pp. 166 – 174.

Başar, T. and Olsder, G. J. (1999), *Dynamic Noncooperative Game Theory*, 2 edn, SIAM.

Beard, R. W., McLain, T. W., Goodrich, M. A. and Anderson, E. P. (2002), 'Coordinated target assignment and intercept for unmanned air vehicles', *IEEE Transactions on Robotics and Automation* **18**(6), 911–922.

Beard, R. W., McLain, T. W., Nelson, D. B., Kingston, D. and Johanson, D. (2006), 'Decentralized cooperative aerial surveillance using fixed-wing miniature UAVs', *Proceedings of the IEEE* **94**(7), 1306–1324.

Beardwood, J., Halton, J. and Hammersly, J. (1959), The shortest path through many points, *in* 'Proceedings of the Cambridge Philosophy Society', Vol. 55, pp. 299–327.

Bertsekas, D. P. and Tsitsiklis, J. N. (1997), *Parallel and Distributed Computation: Numerical Methods*, Athena Scientific.

Bertsimas, D. J. and van Ryzin, G. J. (1991), 'A stochastic and dynamic vehicle routing problem in the Euclidean plane', *Operations Research* **3**9, 601–615.

Bertsimas, D. J. and van Ryzin, G. J. (1993*a*), 'Stochastic and dynamic vehicle routing in the Euclidean plane with multiple capacitated vehicles', *Operations Research* **4**1(1), 60–76.

Bertsimas, D. J. and van Ryzin, G. J. (1993*b*), 'Stochastic and dynamic vehicle routing with general interarrival and service time distributions', *Advances in Applied Probability* **2**5, 947–978.

Bertsimas, D., Paschalidis, I. C. and Tsitsiklis, J. N. (1994), 'Optimization of multiclass queueing networks: Polyhedral and nonlinear characterizations of achievable performance', *The Annals of Applied Probability* **4**(1), 43–75.

Bethke, B., Valenti, M. and How, J. P. (2008), 'UAV task assignment', *IEEE Robotics & Automation Magazine* **1**5(1), 39–44.

Blum, A., Chalasani, P., Coppersmith, D., Pulleyblank, B., Raghavan, P. and Sudan, M. (1994), The minimum latency problem, *in* 'ACM Symposium on the Theory of Computing', Montreal, Canada, pp. 163–171.

Blum, A., Chawla, S., Karger, D. R., Lane, T., Meyerson, A. and Minkoff, M. (2007), 'Approximation algorithms for orienteering and discounted-reward TSP', *SIAM Journal on Computing* **3**7(2), 653–670.

Bopardikar, S. D., Smith, S. L., Bullo, F. and Hespanha, J. P. (2009*a*), 'Dynamic vehicle routing for translating demands: Stability analysis and receding-horizon policies', *IEEE Transactions on Automatic Control* . Submitted.

Bopardikar, S. D., Smith, S. L., Bullo, F. and Hespanha, J. P. (2009*b*), Dynamic vehicle routing with moving demands – Part I: Low speed demands and high arrival rates, *in* 'American Control Conference', St. Louis, MO, pp. 1454–1459.

Borodin, A. and El-Yaniv, R. (1998), *Online Computation and Competitive Analysis*, Cambridge University Press.

Brunet, L., Choi, H. L. and How, J. P. (2008), Consensus-based auction approaches for decentralized task assignment, *in* 'AIAA Conf. on Guidance, Navigation and Control', Honolulu, HI. Electronic Proceedings.

Burkard, R. (2002), 'Selected topics on assignment problems', *Discrete Applied Mathematics* **1**23, 257–302.

Casbeer, D. W., Kingston, D. B., Beard, R. W., Mclain, T. W., Li, S.-M. and Mehra, R. (2006), 'Cooperative forest fire surveillance using a team of small unmanned air vehicles', *International Journal of Systems Sciences* **3**7(6), 351–360.

Castañón, D. A. and Wu, C. (2003), Distributed algorithms for dynamic reassignment, *in* 'IEEE Conf. on Decision and Control', Maui, HI, pp. 13–18.

Chalasani, P. and Motwani, R. (1999), 'Approximating capacitated routing and delivery problems', *SIAM Journal on Computing* **2**8(6), 2133–2149.

Chong, E. K. P., Kreucher, C. M. and Hero III, A. O. (2008), Monte-Carlo-based partially observable Markov decision process approximations for adaptive sensing, *in* 'Int. Workshop on Discrete Event Systems', Göteborg, Sweden, pp. 173–180.

Christofides, N. (1975), *Graph Theory: An Algorithmic Approach*, Academic Press.

Christofides, N. (1976), Worst-case analysis of a new heuristic for the traveling salesman problem, Technical Report 388, Carnegie Mellon University, Pittsburgh, PA.

Clark, J. and Fierro, R. (2007), 'Mobile robotic sensors for perimeter detection and tracking', *ISA Transactions* **4**6(1), 3–13.

Coffman Jr., E. G. and Mitrani, I. (1980), 'A characterization of waiting time performance realizable by single-server queues', *Operations Research* **2**8(3), 810–821.

Cordeau, J. and Laporte, G. (2003), 'The dial-a-ride problem (darp): Variants, modeling issues and algorithms', *4OR: A Quarterly Journal of Operations Research* **1**(2), 89–101.

Cormen, T. H., Leiserson, C. E., Rivest, R. L. and Stein, C. (2001), *Introduction to Algorithms*, 2 edn, MIT Press.

D'Andrea, R. and Murray, R. M. (2003), The RoboFlag competition, *in* 'American Control Conference', Denver, CO, pp. 650–655.

Enright, J. J. and Frazzoli, E. (2006), Cooperative UAV routing with limited sensor range, *in* 'AIAA Conf. on Guidance, Navigation and Control', Keystone, CO. Electronic Proceedings.

Enright, J. J., Savla, K., Frazzoli, E. and Bullo, F. (2009), 'Stochastic and dynamic routing problems for multiple UAVs', *AIAA Journal of Guidance, Control, and Dynamics* **3**4(4), 1152–1166.

Few, L. (1955), 'The shortest path and the shortest road through n points in a region', *Mathematika* **2**, 141–144.

Frazzoli, E. and Bullo, F. (2004), Decentralized algorithms for vehicle routing in a stochastic time-varying environment, *in* 'IEEE Conf. on Decision and Control', Paradise Island, Bahamas, pp. 3357–3363.

Gamal, A. E., Mammen, J., Prabhakar, B. and Shah, D. (2006), 'Optimal throughput-delay scaling in wireless networks. Part I: The fluid model', *IEEE Transactions on Information Theory* **5**2(6), 2568–2592.

Gerkey, B. P. and Mataric, M. J. (2004), 'A formal analysis and taxonomy of task allocation in multi-robot systems', *International Journal of Robotics Research* **2**3(9), 939–954.

Godwin, M. F., Spry, S. and Hedrick, J. K. (2006), Distributed collaboration with limited communication using mission state estimates, *in* 'American Control Conference', Minneapolis, MN, pp. 2040–2046.

Golden, B., Raghavan, S. and Wasil, E. (2008), *The Vehicle Routing Problem: Latest Advances and New Challenges*, Vol. 43 of *Operations Research/Computer Science Interfaces*, Springer.

Gupta, P. and Kumar, P. R. (2000), 'The capacity of wireless networks', *IEEE Transactions on Information Theory* **4**6(2), 388–404.

Hammar, M. and Nilsson, B. J. (2002), 'Approximation results for kinetic variants of TSP', *Discrete and Computational Geometry* **2**7(4), 635–651.

Helvig, C. S., Robins, G. and Zelikovsky, A. (2003), 'The moving-target traveling salesman problem', *Journal of Algorithms* **4**9(1), 153–174.

Hopcroft, J. E. and Karp, R. M. (1973), 'An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs', *SIAM Journal on Computing* **2**(4), 225–231.

Isaacs, R. (1965), *Differential games*, John Wiley.

Itani, S., Frazzoli, E. and Dahleh, M. A. (2008), Dynamic travelling repairperson problem for dynamic systems, *in* 'IEEE Conf. on Decision and Control', Cancun, Mexico, pp. 465–470.

Jaillet, P. and Wagner, M. R. (2006), 'Online routing problems: Value of advanced information and improved competitive ratios', *Transportation Science* **40**(2), 200–210.

Jin., Y., Minai, A. A. and Polycarpou, M. M. (2003), Cooperative real-time search and task allocation in UAV teams, *in* 'IEEE Conf. on Decision and Control', Maui, Hawaii, pp. 7–12.

Kingston, D. B. and Schumacher, C. J. (2005), Time-dependent cooperative assignment, *in* 'American Control Conference', Portland, OR, pp. 4084–4089.

Kleinrock, L. (1975), *Queueing Systems. Volume I: Theory*, John Wiley.

Kleinrock, L. (1976), *Queueing Systems. Volume II: Computer Applications*, John Wiley, New York.

Korte, B. and Vygen, J. (2005), *Combinatorial Optimization: Theory and Algorithms*, Vol. 21 of *Algorithmics and Combinatorics*, 3 edn, Springer.

Koutsoupias, E. and Papadimitriou, C. H. (1995), 'On the k-server conjecture', *Journal of of the ACM* **42**(5), 971–983.

Krumke, S. O., de Paepe, W. E., Poensgen, D. and Stougie, L. (2003), 'News from the online traveling repairmain', *Theoretical Computer Science* **295**(1-3), 279–294.

Kuhn, H. W. (1955), 'The Hungarian method for the assignment problem', *Naval Research Logistics* **2**, 83–97.

Larsen, A. (2000), The Dynamic Vehicle Routing Problem, PhD thesis, Technical University of Denmark, Lyngby, Denmark.

Larson, R. C. and Odoni, A. R. (1981), *Urban Operations Research*, Prentice Hall.

LaValle, S. M. (2006), *Planning Algorithms*, Cambridge University Press. Available at `http://planning.cs.uiuc.edu`.

Le Ny, J., Dahleh, M. A., Feron, E. and Frazzoli, E. (2008), Continuous path planning for a data harvesting mobile server, *in* 'IEEE Conf. on Decision and Control', Cancun, Mexico, pp. 1489–1494.

Le Ny, J., Dahleh, M. and Feron, E. (2008), Multi-UAV dynamic routing with partial observations using restless bandit allocation indices, *in* 'American Control Conference', Seattle, WA, pp. 4220–4225.

Le Ny, J., Feron, E. and Frazzoli, E. (2009), 'On the curvature-constrained traveling salesman problem', *IEEE Transactions on Automatic Control* . To appear.

Leonard, N. E., Paley, D., Lekien, F., Sepulchre, R., Fratantoni, D. M. and Davis, R. (2007), 'Collective motion, sensor networks and ocean sampling', *Proceedings of the IEEE* **9**5(1), 48–74.

Lin, S. and Kernighan, B. W. (1973), 'An effective heuristic algorithm for the traveling-salesman problem', *Operations Research* **2**1, 498–516.

Luenberger, D. G. (1984), *Linear and Nonlinear Programming*, 2 edn, Addison-Wesley.

Manasse, M. S., McGeoch, L. A. and Sleater, D. D. (1990), 'Competitive algorithms for server problems', *Journal of Algorithms* **1**1(2), 208–230.

Martínez, S., Cortés, J. and Bullo, F. (2007), 'Motion coordination with distributed information', *IEEE Control Systems Magazine* **2**7(4), 75–88.

McKee, T. A. and McMorris, F. R. (1999), *Topics in Intersection Graph Theory*, Vol. 2 of *Monographs on Discrete Mathematics and Applications*, SIAM.

Megiddo, N. and Supowit, K. J. (1984), 'On the complexity of some common geometric location problems', *SIAM Journal on Computing* **1**3(1), 182–196.

Mitchell, J. S. B. (1999), 'Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric TSP, k-MST, and related problems', *SIAM Journal on Computing* **2**8(4), 1298–1309.

Moore, B. J. and Passino, K. M. (2007), 'Distributed task assignment for mobile agents', *IEEE Transactions on Automatic Control* **5**2(4), 749–753.

Motwani, R. and Raghavan, P. (1995), *Randomized Algorithms*, Cambridge University Press.

Newman, J. N. (1984), 'Approximations for the Bessel and Struve functions', *Mathematics of Computation* **4**3(168), 551–556.

Papastavrou, J. D. (1996), 'A stochastic and dynamic routing policy using branching processes with state dependent immigration', *European Journal of Operational Research* **9**5, 167–177.

Parker, L. E. (1998), 'ALLIANCE: An architecture for fault tolerant multirobot cooperation', *IEEE Transactions on Robotics and Automation* **1**4(2), 220–40.

Pavone, M., Arsie, A., Frazzoli, E. and Bullo, F. (2008), 'Equitable partitioning policies for mobile robotic networks', *IEEE Transactions on Automatic Control* . Submitted.

Pavone, M., Bisnik, N., Frazzoli, E. and Isler, V. (2008), 'A stochastic and dynamic vehicle routing problem with time windows and customer impatience', *ACM/Springer Journal of Mobile Networks and Applications* . To appear.
URL: *http://ares.lids.mit.edu/papers/Pavone.Bisnik.ea.MONE08.pdf*

Pavone, M., Frazzoli, E. and Bullo, F. (2009), 'Distributed and adaptive algorithms for vehicle routing in a stochastic and dynamic environment', *IEEE Transactions on Automatic Control* . Submitted.

Pavone, M., Smith, S. L., Bullo, F. and Frazzoli, E. (2009), Dynamic multi-vehicle routing with multiple classes of demands, *in* 'American Control Conference', St. Louis, MO, pp. 604–609.

Penrose, M. (2003), *Random Geometric Graphs*, Oxford Studies in Probability, Oxford University Press.

Percus, G. and Martin, O. C. (1996), 'Finite size and dimensional dependence of the Euclidean traveling salesman problem', *Physical Review Letters* **7**6(8), 1188–1191.

Powell, W. B., Jaillet, P. and Odoni, A. (1995), Stochastic and dynamic networks and routing, *in* M. O. Ball, T. L. Magnanti, C. L. Monma and G. L. Nemhauser, eds, 'Handbooks in Operations Research and Management Science, 8: Network Routing', Elsevier, pp. 141–295.

Psaraftis, H. N. (1980), 'Dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem', *Transportation Science* **1**4(2), 130–154.

Psaraftis, H. N. (1988), Dynamic vehicle routing problems, *in* B. Golden and A. Assad, eds, 'Vehicle Routing: Methods and Studies', Elsevier (North-Holland), pp. 223–248.

Russell, S. and Norvig, P. (2003), *Artificial Intelligence: A Modern Approach*, 2 edn, Prentice Hall.

Sahni, S. and Gonzalez, T. (1976), 'P-complete approximation problems', *Journal of the ACM* **2**3(3), 555–565.

Savla, K., Bullo, F. and Frazzoli, E. (2009), 'Traveling Salesperson Problems for a double integrator', *IEEE Transactions on Automatic Control* **5**4(4), 788–793.

Savla, K., Frazzoli, E. and Bullo, F. (2008), 'Traveling Salesperson Problems for the Dubins vehicle', *IEEE Transactions on Automatic Control* **53**(6), 1378–1391.

Savla, K., Temple, T. and Frazzoli, E. (2008), Human-in-the-loop vehicle routing policies for dynamic environments, *in* 'IEEE Conf. on Decision and Control', Cancún, México, pp. 1145–1150.
URL: *http://web.mit.edu/ksavla/www/papers/Savla.Temple.ea:CDC08.pdf*

Schumacher, C., Chandler, P. R., Rasmussen, S. J. and Walker, D. (2003), Task allocation for wide area search munitions with variable path length, *in* 'American Control Conference', Denver, CO, pp. 3472–3477.

Sharma, G., Mazumdar, R. and Shroff, N. (2006), Delay and capacity trade-offs in mobile ad hoc networks: A global perspective, *in* 'IEEE Conf. on Computer Communications', Barcelona, Spain, pp. 1–12.

Sharma, V., Savchenko, M., Frazzoli, E. and Voulgaris, P. (2007), 'Transfer time complexity of conflict-free vehicle routing with no communications', *International Journal of Robotics Research* **26**(3), 255–272.

Smith, S. L., Bopardikar, S. D. and Bullo, F. (2009), Finite-time pursuit of translating targets in a dynamic and stochastic environment, *in* 'IEEE Conf. on Decision and Control', Shanghai, China. To appear.

Smith, S. L., Bopardikar, S. D., Bullo, F. and Hespanha, J. P. (2009), Dynamic vehicle routing with moving demands – Part II: High speed demands or low arrival rates, *in* 'American Control Conference', St. Louis, MO, pp. 1466–1471.

Smith, S. L. and Bullo, F. (2007*a*), Target assignment for robotic networks: Asymptotic performance under limited communication, *in* 'American Control Conference', New York, pp. 1155–1160.

Smith, S. L. and Bullo, F. (2007*b*), Target assignment for robotic networks: Worst-case and stochastic performance in dense environments, *in* 'IEEE Conf. on Decision and Control', New Orleans, LA, pp. 3585–3590.

Smith, S. L. and Bullo, F. (2008), Dynamic multi-agent team forming: Asymptotic results on throughput versus delay, *in* 'American Control Conference', Seattle, WA, pp. 1406–1411.

Smith, S. L. and Bullo, F. (2009*a*), 'The dynamic team forming problem: Throughput and delay for unbiased policies', *Systems & Control Letters* . (Submitted Jul 2008) to appear.

Smith, S. L. and Bullo, F. (2009*b*), 'Monotonic target assignment for robotic networks', *IEEE Transactions on Automatic Control* **54**(10). (Submitted June 2007) to appear.

Smith, S. L., Pavone, M., Bullo, F. and Frazzoli, E. (2008), Dynamic vehicle routing with heterogeneous demands, *in* 'IEEE Conf. on Decision and Control', Cancún, México, pp. 1206–1211.

Smith, S. L., Pavone, M., Bullo, F. and Frazzoli, E. (2009), 'Dynamic vehicle routing with priority classes of stochastic demands', *SIAM Journal on Control and Optimization* . Submitted.

Solomon, M. M. (1987), 'Algorithms for the vehicle routing and scheduling problems with time window constraints', *Operations Research* **3**5(2), 254–265.

Spivey, M. Z. and Powell, W. B. (2004), 'The dynamic assignment problem', *Transportation Science* **3**8(4), 399–419.

Steele, J. M. (1990), 'Probabilistic and worst case analyses of classical problems of combinatorial optimization in Euclidean space', *Mathematics of Operations Research* **1**5(4), 749.

Sutton, R. S. and Barto, A. G. (1998), *Reinforcement Learning: An Introduction*, MIT Press.

Swihart, M. R. and Papastavrou, J. D. (1999), 'A stochastic and dynamic model for the single-vehicle pick-up delivery problem', *European Journal of Operational Research* **1**14, 447–464.

Takagi, H. (1986), *Analysis of Polling Systems*, MIT Press, Cambridge, MA.

Tang, Z. and Özgüner, Ü. (2005), 'Motion planning for multi-target surveillance with mobile sensor agents', *IEEE Transactions on Robotics* **2**1(5), 898–908.

Toth, P. and Vigo, D., eds (2001), *The vehicle routing problem*, Monographs on Discrete Mathematics and Applications, SIAM.

Waisanen, H. A., Shah, D. and Dahleh, M. A. (2008), 'A dynamic pickup and delivery problem in mobile networks under information constraints', *IEEE Transactions on Automatic Control* **5**3(6), 1419–1433.

Wolff, R. W. (1989), *Stochastic modeling and the theory of queues*, Prentice Hall.

Xu, H. (1995), Optimal Policies for Stochastic and Dynamic Vehicle Routing Problems, Dept. of Civil and Environmental Engineering, Massachusetts Institute of Technology, Cambridge, MA.

Xue, F. and Kumar, P. R. (2004), ‘The number of neighbors needed for connectivity of wireless networks’, *Wireless Networks* **1**0(2), 169–181.

Zavlanos, M. M. and Pappas, G. J. (2008), ‘Dynamic assignment in distributed motion planning with local coordination’, *IEEE Transactions on Robotics* **2**4(1), 232–242.

Zemel, E. (1984), ‘Probabilistic analysis of geometric location problems’, *SIAM Journal on Algebraic and Discrete Methods* **6**(2), 189–200.