Empty Vehicle Redistribution for Personal Rapid Transit

John D. Lees-Miller

A dissertation submitted to the University of Bristol in accordance with the requirements for award of the degree of PhD in the Faculty of Engineering.

July, 2011
24415 words

Abstract

A Personal Rapid Transit (PRT) system uses compact, computer-guided vehicles running on dedicated guideways to carry individuals or small groups directly between pairs of stations. PRT vehicles operate on demand, much like conventional taxis. The empty vehicle redistribution (EVR) problem is to decide when and where to move empty PRT vehicles. These decisions are made in real time by an EVR algorithm. A reactive EVR algorithm moves empty vehicles only in response to known requests; in contrast, a proactive EVR algorithm moves empty vehicles in anticipation of future requests. In this thesis, two new proactive EVR algorithms, here called Sampling and Voting (SV) and Dynamic Transportation Problem (DTP), are developed and evaluated. It is shown that they reduce passenger waiting times substantially below those obtained by reactive EVR algorithms, with a modest increase in empty vehicle travel, and that they usually outperform similar algorithms in the literature. Several new theoretical tools are also developed, including a benchmark for maximum achievable throughput and two benchmarks for minimum achievable passenger waiting times. These provide an absolute measure of the performance of EVR algorithms, and they quantify the potential for further improvement. Finally, preliminary work on a Markov Decision Process formulation of the EVR problem is presented and used to obtain provably optimal policies for small systems.

Dedication

For my friends and family.

Acknowledgements

Prof. R. E. Wilson (supervisor)

Prof. M. G. H. Bell (external examiner)
Dr R. Clifford (internal examiner)

Dr P. H. Bly
Prof. C. J. Budd
Prof. A. R. Champneys
Dr N. Davenport
Dr J. C. Hammersley
Dr R. J. Gibbens
Prof. F. P. Kelly
N. Koren
Prof. M. V. Lowson
Dr J. A. Padget
Dr A. J. Peters

Funding was provided by an Overseas Research Scholarship from the University of Bristol; the CityMobil Sixth Framework Programme for DG Research Thematic Priority 1.6, Sustainable Development, Global Change and Ecosystems, Integrated Project, Contract Number TIP5-CT-2006-031315; and REW's EPSRC Advanced Fellowship EP/E055567/1.

Author's Declaration

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Research Degree Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the dissertation are those of the author.

SIGNED: ....

DATE: ....

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Personal Rapid Transit (PRT) is an emerging urban transport mode. It will use small, computer-guided vehicles to carry individuals and small groups between pairs of stations on a dedicated network of guideways. The vehicles will operate on-demand and provide direct service from origin station to destination station. The most similar existing mode is the street-hail (hackney) taxi; the main differences are that PRT vehicles are driven by computers, and that demand is concentrated at special-purpose PRT stations.

The world's first PRT system opened in Masdar City in November, 2010 with ten vehicles and two stations for passenger use, and a further three vehicles and three stations for freight (2getthere, 2011a). Another PRT system with 21 vehicles (Figure 1.1) and three stations is in the final stages of testing at London's Heathrow Airport (ULTra PRT, 2010). Several larger systems have been proposed for development in the near future (Bly and Teychenne, 2005).

This thesis is concerned with the management of empty vehicles in PRT systems. The need for empty vehicle trips arises because passenger demand between stations is often unbalanced, either in the short term or the long term. The Empty Vehicle Redistribution (EVR) problem is to decide which empty vehicles to move, and where to move them. These decisions must be made in real time as the system is operating and as new passengers' requests for service are received. The two main objectives are to avoid unnecessary

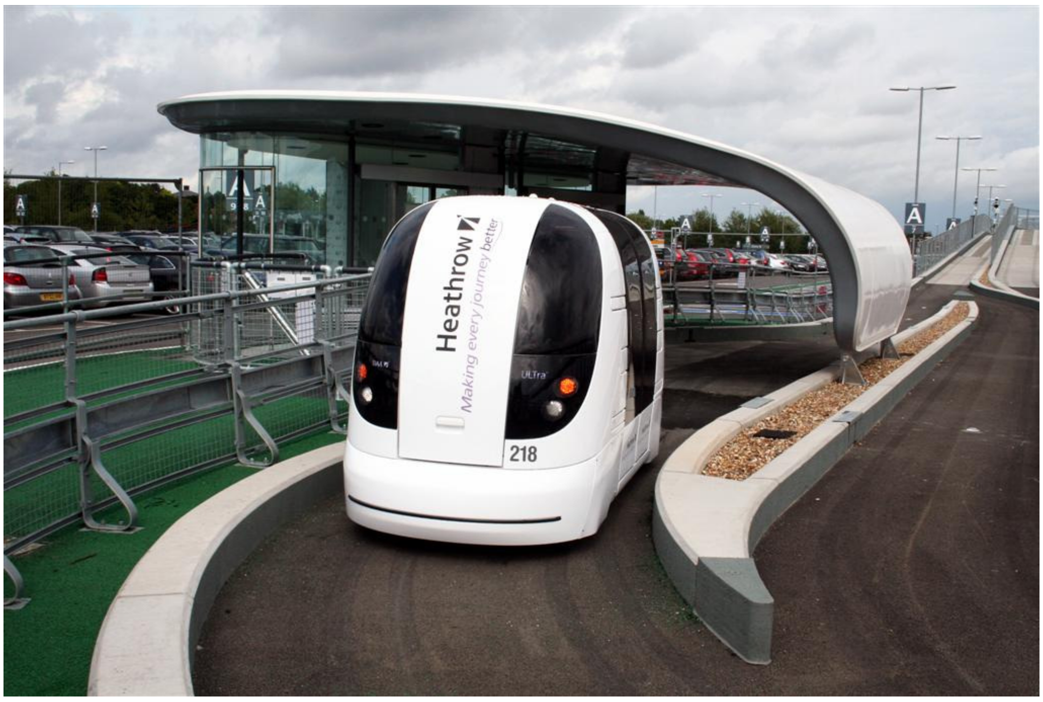


Figure 1.1: Photograph of a PRT vehicle and at-grade station at London Heathrow Airport. PRT vehicles, stations and infrastructure are smaller than typical Automated People Mover and urban rail systems. Vehicle length, width and height are 3.7m, 1.4m and 1.8m, respectively. Photo courtesy of ULTra PRT Ltd.

empty vehicle running, which increases operating costs, system congestion and energy use, and to keep passenger waiting times low, which makes the system more attractive to passengers.

To provide low waiting times, the system must move empty vehicles *proactively*, in anticipation of future demand. It is assumed that, like in the Heathrow and Masdar applications, passengers request immediate travel from their origin station to their chosen destination station (that is, they do not call ahead). A passenger's waiting time is the delay between when the system receives his request and when he is picked up by a vehicle. If the assigned vehicle is not already idle at, or going to, the passenger's origin station, an empty trip is required, and the passenger's waiting time includes this empty trip time. Proactive empty vehicle movement can reduce or eliminate passenger waiting time due to empty trips by starting empty trips earlier, where possible, in anticipation of future requests.

The main outputs of this thesis are two new EVR algorithms that move empty vehicles proactively. In most cases, they give lower mean waiting times and require less empty vehicle running than other algorithms in the literature. These algorithms are described and evaluated in Chapter 4. The rest of the thesis develops the underlying models and theory, as outlined in Section 1.3. First, Section 1.1 introduces PRT for those who are new to the subject, and Section 1.2 describes the basic modelling assumptions and notation that we will use in this thesis.

1.1 Personal Rapid Transit Background

This section introduces the main ideas of PRT. The most similar existing mode is the public hire (hackney) taxi; the key similarities are:

1. The vehicles are small; they carry 2-6 passengers at once. These passengers will ordinarily be travelling together by choice, and they will have the same origin and the same destination.

2. The system is demand-responsive; the vehicles do not run on schedules. Passengers do not usually book in advance.

3. A vehicle takes its passengers directly from their origin to their destination, without stopping to pick up others. The vehicle takes the quickest path, without any detours.

The key differences between PRT and taxis are:

1. PRT vehicles are driven by computers instead of humans; guidance is fully automatic.

2. Safe computer guidance currently requires that PRT vehicles run on dedicated infrastructure that is physically separated from pedestrians and normal road traffic.

3. Passengers can board and alight only at designated PRT stations.

Many PRT implementations are possible, and many PRT vendors have proposed materially different designs. Some of the details remain controversial. This section describes common features of both extant and commonly proposed[1] PRT implementations, with an emphasis on those that are relevant to empty vehicle redistribution.

1.1.1 Networks

PRT vehicles run on dedicated *guideways*, which are typically grade-separated from the roads and pavements used by cars and pedestrians (Figure 1.2). Because passengers cannot access the guideway directly, they must board and alight vehicles at designated PRT *stations*.

In order to provide taxi-like service, PRT stations must be sited close together, and they must cover the whole area to be served. The aim is to make it comfortable for passengers to walk to their nearest PRT station. PRT stations can also be integrated directly into the buildings that they serve (Figure 1.3). It has been observed that people will not walk much more than 400m to reach a bus station (Anderson, 1978, p. 137), which implies that PRT stations will typically be less than 800m (half a mile) apart.

[1]For simplicity, the present tense will be used throughout to refer to features of both extant and commonly proposed implementations.

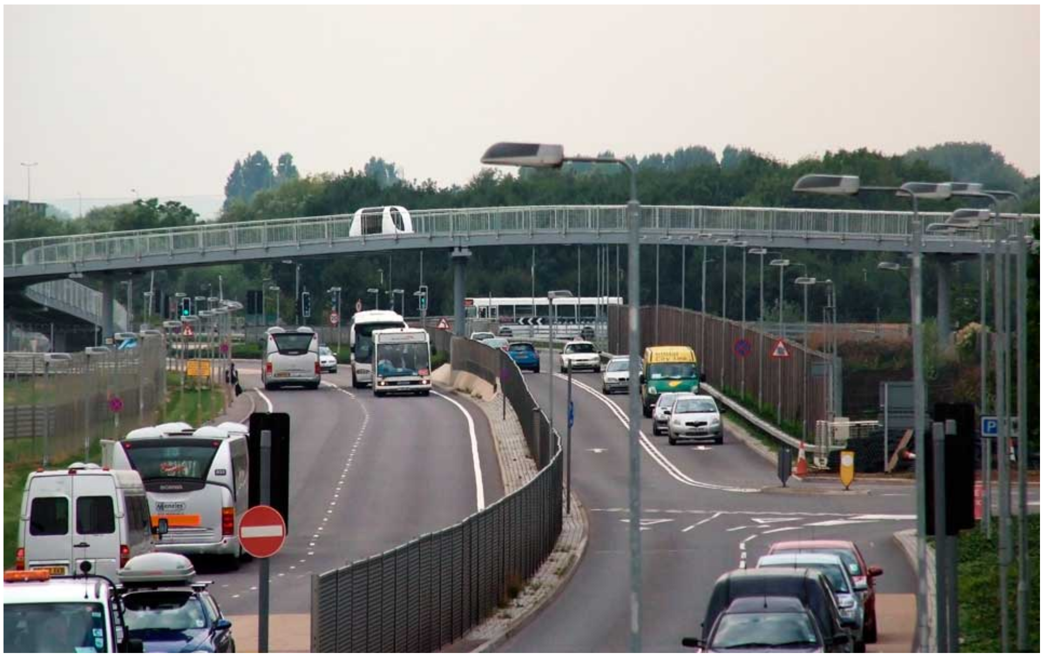


Figure 1.2: Photograph of an elevated two-lane PRT guideway section at London Heathrow Airport. Nominal guideway elevation over a main road is 5.7m (18' 8"). An ULTra vehicle is moving on the guideway, just left of the central column. Photo courtesy of ULTra PRT Ltd.

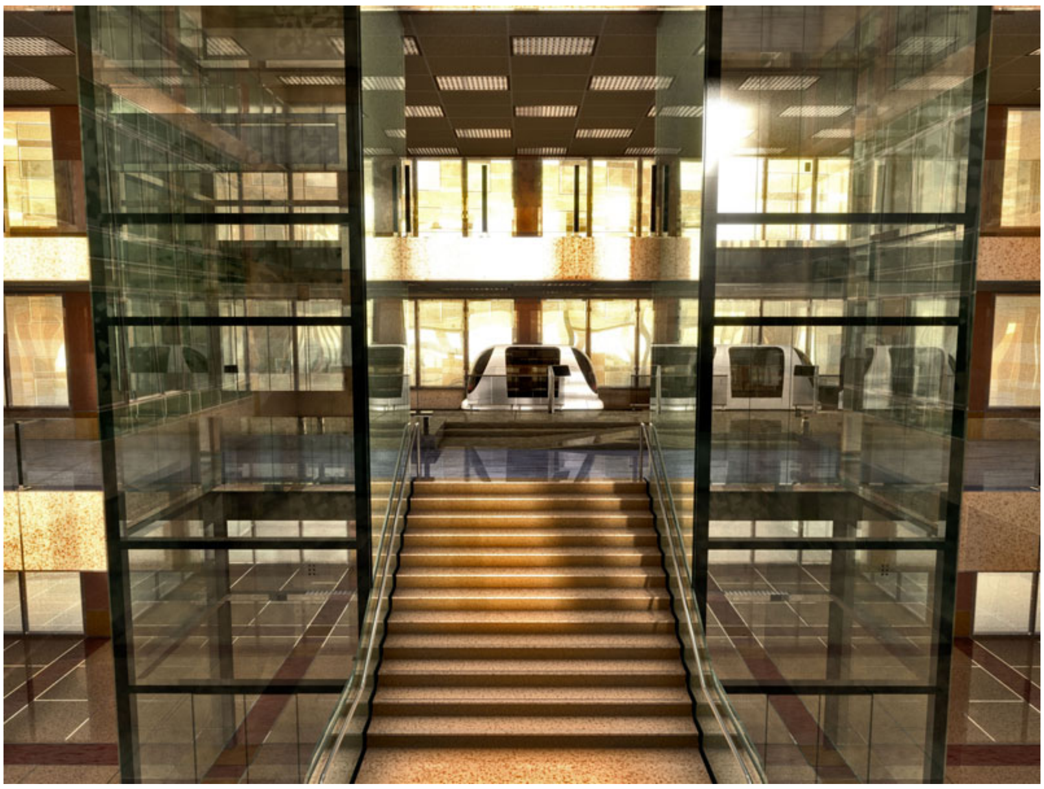
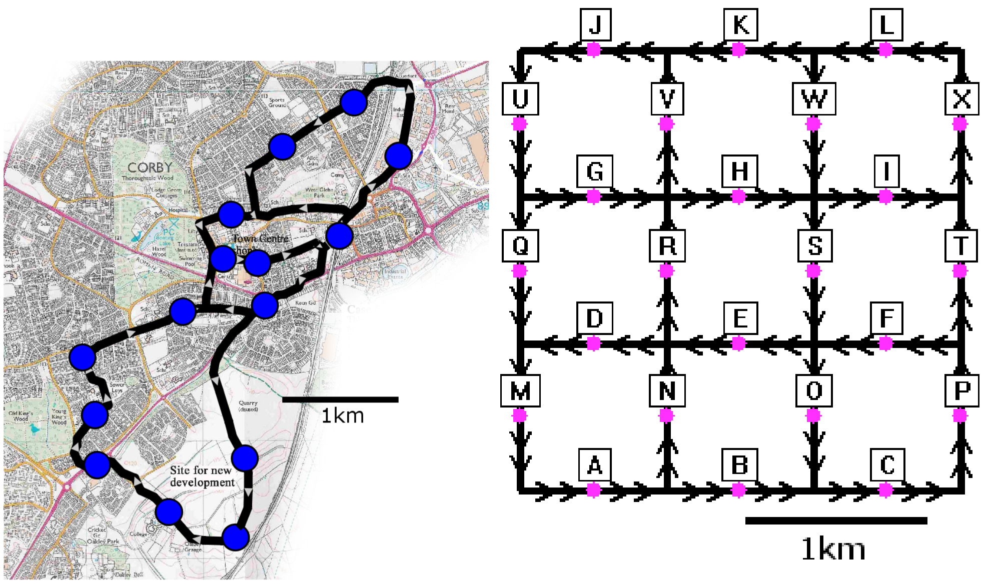


Figure 1.3: Conceptual rendering of a PRT station built into the first floor of a building. Rendering courtesy of ULTra PRT Ltd.



(a) (b)

Figure 1.4: The (a) Corby and (b) Grid PRT networks. Guideways (black lines) are one-way in the direction indicated; circles represent stations in (a), and letters represent stations in (b). These networks will be used as input data for stochastic simulations in later chapters.

The need to construct a dense and extensive network of guideways is a major drawback of PRT, because it is expensive and takes up scarce urban space. It is therefore important not to use more guideway than necessary. It is common (Anderson, 1978, pp. 64–67, 82–86) to connect the stations with single lane, one-way loops of guideway. This reduces the amount of guideway needed to cover the service area, compared to a network of two lane, two-way guideways. The penalty is that journey times between some pairs of stations are increased, because those trips must go the long way around a loop. This penalty is small when the loops are small. Figure 1.4a shows the Corby case study network from Bly and Teychenne (2005), which is an example of this approach; it consists of four interconnected one-way loops of guideway.

Another commonly proposed network configuration (Fichter, 1964; Irving et al., 1978; Anderson, 1978; Lowson, 2004) is a grid of elevated, one way guideways with alternating directions, as illustrated in Figure 1.4b. This is similar to the road networks found in the centres of many modern cities, except that the PRT 'avenues' and 'streets' do not intersect where they cross; instead, each junction is a fly-over. This increases capacity and precludes some kinds of collisions. Planar networks based on concentric rings (Sirbu, 1974) or regular tilings of octagons and squares, have also been proposed (Lowson, 1999).

1.1.2 Stations

PRT stations are generally built *off-line*, on a bypass lane (Figure 1.5). This allows PRT vehicles to continue past stations that are on the way to their destination without stopping. This contrasts with rail stations, which are typically *on-line*: when a train stops at the station, it remains on the main line, so it prevents other trains from passing through the station. The bypass must be long enough for vehicles to decelerate from main line speed to a standstill, and later to accelerate back to main line speed.

Within a station, passengers board and alight at *berths*. Berths can be arranged in a line, or they can be staggered in a saw-tooth configuration, as in Figure 1.5. A linear station has a smaller footprint than a saw-tooth

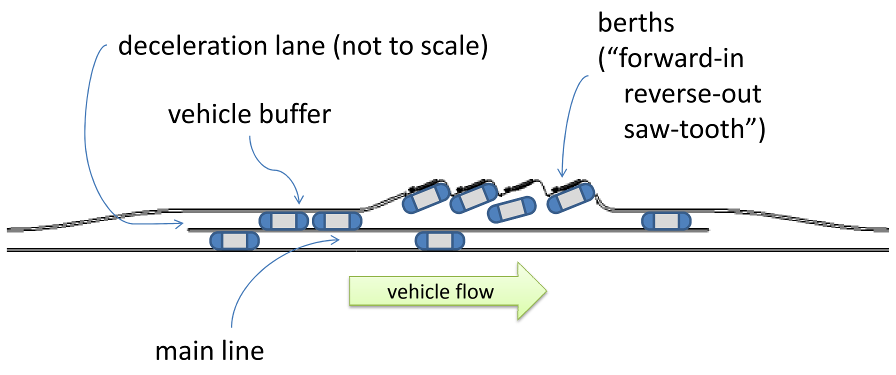

deceleration lane (not to scale)
berths
("forward-in reverse-out saw-tooth")
vehicle buffer
vehicle flow
main line

Figure 1.5: Schematic of an off-line PRT station. Vehicles (rounded rectangles) that are not stopping at the station continue on the main line. Vehicles that are stopping diverge from the main line and decelerate. Vehicles may be held in a buffer before proceeding to a berth. Vehicles leaving the station merge back onto the main line.

station; this reduces station cost, but it is less flexible, because vehicles in the forward berths impede those behind. A saw-tooth station allows vehicles some degree of random access to berths.

A station can also contain *queue points*, where vehicles can be stored when all of the berths are taken. Queue points provide 'buffer space' that can help deal with variability in the flow of vehicles in the station. Both occupied and empty vehicles can wait at queue points. Queue points are typically placed upstream of the berths, at the end of the deceleration lane; in systems with asynchronous control (Section 1.1.4), it may also be desirable to put queue points between the berths and the acceleration lane. Queue points are typically arranged in one or more lanes; each lane is first-in-first-out. The total number of berths and queue points limits the number of vehicles that can be stored in a station and determines the maximum throughput of the station. This 'station sizing' problem has been studied extensively (Waddell et al., 1973; Sirbu, 1974; Johnson et al., 1976; Anderson, 2003; Won et al., 2006; Schweizer and Mantecchini, 2007; Schweizer et al., 2010).

On the passenger side of the station, procedures are required for destination selection and, in most cases, fare payment. These may be done at the berth (Figure 1.6), in the vehicle, or at the station entrance (Irving et al., 1978, ch. 3). These affect empty vehicle redistribution mainly in terms of how well the system can measure the number of waiting passengers at each station, and also when it finds out about each passenger's intended destination.

1.1.3 Vehicles

The details of automatic vehicle guidance are beyond the scope of this thesis. Many different vehicle guidance and propulsion systems have been proposed (Caudill et al., 1979; Lowson, 2003; Anderson, 2005; Featherstone, 2005). Vehicles are typically electric. If power is supplied by on-board batteries then this affects empty vehicle redistribution, because some empty vehicles may have to be removed from service or delayed in order to charge. However, we will not deal with this issue here.

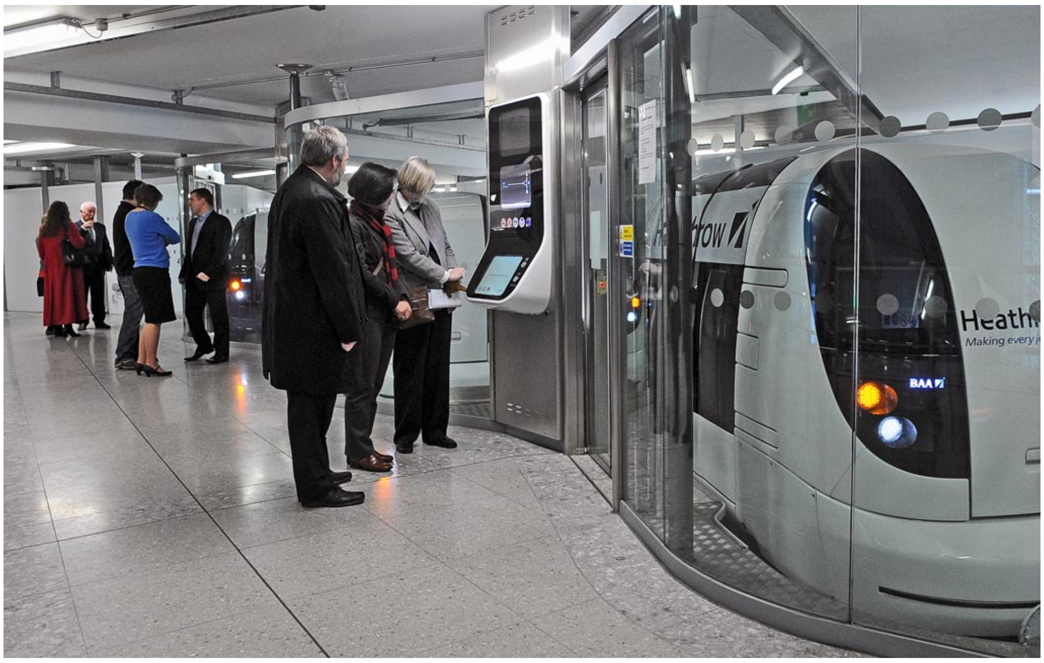


Figure 1.6: Passengers at berths in the Terminal 5 station at Heathrow during a capacity trial in March, 2010. Passengers use touch-screen destination selection panels at berths to tell the system which station they want to travel to. Photo courtesy of ULTra PRT Ltd.

1.1.4 Control Systems

Collisions are avoided by maintaining a minimum safe separation between vehicles at all times; this is called the *minimum headway*. It is common to use a 'brick wall stop' minimum headway, which is long enough to ensure that if the lead vehicle were to stop instantaneously (as though it had become a brick wall), the following vehicle would be able to stop in time to avoid a collision. The minimum headway is 6.4s in the Heathrow application (ULTra PRT, 2010) and 4s in the Masdar application (2getthere, 2011b). Shorter minimum headways permit higher line capacity; for example, 4-passenger PRT vehicles at 3s minimum headway give a line capacity of 4800 passengers per hour, which is equivalent to 400-passenger trains running at 5 minute headways. Minimum headways as low as 0.5s have been proposed for PRT (Anderson, 1998). Research in Automated Highway Systems also indicates that small minimum headways are technically achievable (Bender, 1991).

The two main methods for maintaining safe separation are known as *synchronous* and *asynchronous* control. In synchronous control (also called *clear-path* control), a central computer schedules each vehicle's whole trip before it leaves its origin station, ensuring that the chosen path does not intersect that of any previously scheduled trip (Irving et al., 1978). To avoid collisions, it is sufficient for the vehicles to stay on their allocated schedules. The Heathrow application uses synchronous control (Lowson, 2003). Asynchronous control is more decentralised. Each merge typically has a local controller that coordinates the vehicles as they approach the merge. Many systems have been proposed for this process (Munson, 1972; York, 1974; Andréasson, 1994; Szillat, 2001; Anderson, 2003).

From the perspective of empty vehicle redistribution, the main difference is that asynchronous control naturally allows vehicles (occupied or empty) to be rerouted once they are moving, which gives the system more flexibility to optimise empty vehicle routes (Andréasson, 2003). In principle, rerouting is also possible with synchronous control using rescheduling, but it requires that the central control system be able to communicate revised schedules to all affected vehicles.

1.1.5 History

PRT has, perhaps surprisingly, a long history. The first published description of PRT dates from 1964 (Fichter, 1964), and its author claims that he first began work on the idea in 1953. Many other PRT implementations were proposed from the late 1960s to the late 1970s, and several of these progressed to full-scale test tracks. Notable examples include CVS in Japan (Ishii et al., 1976), Aramis in France (Levy, 1976; Latour, 1996) and Cabintaxi in Germany (Becker, 1976). All of these failed for political, technical or financial reasons. In the United States, a PRT system was commissioned in 1970 to connect three campuses of West Virginia University in Morgantown, West Virginia, but in 1975, after several delays, budget overruns and redesigns, the result was not a PRT system (Rydell, 2001). While this system, which still operates today, is called the 'Morgantown PRT', it uses minibus-sized vehicles that carry up to 20 passengers, so it is more properly called Group Rapid Transit (GRT). Several other implementations were proposed and developed in the 1980s and 1990s, but none of these came to fruition. The reader is referred to Anderson (1996) and Cottrell (2005) for more on the history of PRT.

1.1.6 Capacity

The *capacity* of a transport system is the highest rate at which it can serve passengers under ideal conditions. The capacity of a PRT system depends on several factors, and the fact that it is a network system makes it difficult to define any single number that captures the system's capacity. The main limitations on capacity are as follows.

1. **Line capacity.** The capacity of a single lane of PRT guideway is determined by the system's minimum headway and the capacity of each vehicle (Anderson, 1978). This is analogous to the capacity of a single road lane or rail line.

2. **Station capacity.** The capacity of an individual PRT station is determined by the number of berths and queue points in the station and

the way in which they are arranged and managed (Won et al., 2006; Schweizer et al., 2010).

3. **Fleet capacity.** The capacity of the fleet as a whole is determined by the number of vehicles, the capacity of each vehicle and time taken per trip. This includes both the occupied trip time and the empty trip time. The total time taken per trip depends on the spatial distribution of the passenger demand.

Which of these factors is limiting depends on the demand. They are also interrelated; for example, congestion on the line can prevent vehicles from leaving a station, which contributes to congestion in the station. Ride sharing (Lees-Miller et al., 2009) is another important factor in determining overall capacity; when the system is busy, several parties may choose to share a vehicle. The main focus of this thesis will be on fleet capacity, because this is directly affected by empty vehicle redistribution.

1.2 Modelling Assumptions

Microsimulation is the most common method used to model PRT systems. Many detailed system simulation programs have been developed over the past forty years; see Anderson (2007) for a recent survey. Most of these programs are proprietary, though some are freely available (ULTra PRT, 2008; Xithalis, 2011) in limited form, and there are some early-stage open-source simulators (Homerick, 2010). These programs simulate central control, stations, vehicles and other subsystems at high levels of detail. They typically consist of many thousands of lines of code and require considerable effort to develop and describe. Here we will use simpler models that permit more focus on the empty vehicle redistribution aspects of the system.

We will be concerned mainly with passenger waiting times. The three main factors that determine passenger waiting times are congestion on the guideway, congestion in stations and the availability and locations of empty vehicles. For very large systems with many vehicles, congestion effects will often be significant, but most systems proposed for the near and medium term will operate well below the congested limit. This motivates the following simplifying assumptions.

1. Congestion on the guideway is ignored, so vehicles take quickest paths, and the trip times between stations are constants.

2. Congestion in stations is ignored; thus, any delays in stations are constants incorporated in the trip times.

Under these assumptions, the relevant characteristics of a PRT network are the number of stations and the quickest path times between those stations. The shortest path between each pair of stations network can be obtained from a shortest paths algorithm, such as Dijkstra's algorithm (Cormen et al., 2001). Quickest path times can be obtained from shortest path distances by assuming an average vehicle speed (usually 10m/s), or directly from a shortest paths algorithm, if nominal vehicle speed data are available for each network link.

Let S denote the set of stations, and let n_S denote the number of stations. For each pair of distinct stations i and j in S, let t_{ij} denote the quickest path *trip time* from i to j in seconds, and define constants $t_{ij} = 0$ when $i = j$. Taken together, these t_{ij} form a *trip time matrix* with n_S rows, n_S columns and zeros on the diagonal. For any three stations i, j and h in S, the trip times satisfy the triangle inequality

$$t_{ih} + t_{hj} \geq t_{ij}$$

because they are quickest path times. Moreover, because all stations are off-line, it is assumed that the travel times satisfy the *strict* triangle inequality

$$t_{ih} + t_{hj} > t_{ij} \tag{1.1}$$

That is, even if station h is 'on the way' from station i to station j, there is a time penalty for stopping at h. This penalty is assumed to reflect any time needed for acceleration and deceleration, passenger loading and unloading, or congestion in the station.

The unit of passenger demand will be the *request*. Each request results in a single occupied vehicle trip, which may be for an individual passenger or a small party of passengers travelling together by choice. Each request r has associated with it an origin station, i_r, a destination station j_r and the time e_r at which the system receives the request. It is assumed that every request is for immediate travel from its origin station, so the *waiting time* of a request is the delay between e_r and when a vehicle *picks up* the request from i_r.

It is assumed that requests for travel from station i to station j are received according to a Poisson process with rate d_{ij} in requests per minute, where $d_{ij} = 0$ if $i = j$ (no recreational trips). The Poisson processes for pairs of stations are assumed to be independent and stationary (that is, the d_{ij} do not vary with time). Taken together, these d_{ij} form a *demand matrix* with n_S rows, n_S columns and zeros on the diagonal.

The capacity of a vehicle is defined to be one request. It is assumed that the number of vehicles in the fleet is fixed, and that the vehicle fleet is homogeneous (that is, vehicles are interchangeable). The set of vehicles will be denoted K, and n_K will denote the *fleet size*, which is simply the number of vehicles in K.

1.3 Thesis Outline

We will begin Chapter 2 by working in the *fluid limit*, at the level of long-run average flows of vehicles. This fluid limit analysis provides a way to characterise the 'heaviest' demands that a given PRT system could possibly serve, with an ideal EVR algorithm. This provides a benchmark against which particular EVR algorithms can be compared, in terms of their *throughput*.

Chapter 2 then describes two simple EVR algorithms and tests them against this benchmark. One of these algorithms, here called Bell and Wong Nearest Neighbours (BWNN) (Bell and Wong, 2005), comes close to achieving maximum theoretical throughput in a variety of test scenarios. BWNN is a *reactive* algorithm: it moves empty vehicles only in response to requests that have already been received. It will be seen that this leads to large passenger waiting times, even for light to moderate demands. The proactive EVR algorithms introduced later (Chapter 4) are extensions to BWNN that aim to reduce passenger waiting times at light to moderate demand.

Chapter 3 develops two benchmarks for passenger waiting times. The first is based on a queueing theory approximation that uses the results of the fluid limit analysis. The second benchmark is based on solving a *static* problem, in which perfect information is available at the level of individual future requests over some fixed horizon. This static problem is related to several well-studied problems in the vehicle routing literature. It is NP-hard, but small instances can be solved exactly with standard techniques, using a mixed integer linear programming formulation. For larger instances, we develop a simple constructive heuristic, here called Static Nearest Neighbours (SNN), which is based on BWNN. For low to moderate demand (up to about 80% of the theoretical maximum), SNN usually finds solutions with zero waiting time. For higher demand, the heuristic provides a benchmark for achievable waiting times.

Chapter 4 then introduces the two new proactive EVR heuristics, and it compares the obtained waiting times to the benchmarks derived in Chapter 3. The new algorithms substantially reduce mean passenger waiting times compared to reactive BWNN baseline, and they usually outperform algorithms from the PRT literature in evaluations on a set of test scenarios. The strengths and weaknesses of the new algorithms are also discussed, and several cases in which they do not perform well are identified.

Chapter 5 discusses directions for future work and presents conclusions. In particular, Section 5.1 describes the early stages of a different approach to the EVR problem based on the theory of Markov Decision Processes (MDPs). This approach can in principle produce algorithms that have stronger guarantees than the algorithms developed in Chapter 4. Whereas we have so far explicitly partitioned the problem into reactive and proactive subproblems, MDPs may allow us to formulate the problem in a unified way and produce operating policies that are optimal in a well-defined and rigorous sense. However, finding these policies exactly is intractable. This section presents a formal model of a PRT system and uses it to compute optimal policies for some small systems. Finally, Section 5.2 summarises the thesis and discusses several other directions for future work.

Chapter 2

A Benchmark for Throughput

The motivating question for this chapter is, 'what demands can a given PRT system serve?' To answer this question, we will look at the behaviour of the system in the *fluid limit*, at the level of long run average flows of vehicles. To be precise, we will say the system can serve a given demand if it can keep the mean number of waiting passengers (and thus their waiting times) finite when that demand is held constant indefinitely. Otherwise, requests are being received faster than the system can serve them, so the number of waiting passengers (and thus their waiting times) will, on average, keep growing forever.

Section 2.1 derives necessary conditions for a system to be able to serve a given demand; these conditions are based on solving a well-known linear program to find flows of empty vehicles in the fluid limit. The set of all demands that satisfy these conditions will be called the system's *capacity region*. If the demand is outside of the the system's capacity region, then the system cannot serve the demand, regardless of which EVR algorithm it uses. If the demand is inside the capacity region, then the system may or may not be able to serve the demand, depending on how efficiently its EVR algorithm uses empty vehicles. The capacity region thus provides a benchmark for throughput, because its boundary describes the 'heaviest' demands that the system could possibly be expected to serve.

Section 2.2 introduces two simple EVR algorithms and Section 2.4 evaluates their throughput in simulation to determine how close they come to achieving the benchmark defined by the capacity region. The simulations use the case study scenarios described in Section 2.3. The case study scenarios are divided into training scenarios, which were used to guide the development of the algorithms developed in this thesis, and testing scenarios, which were used only for evaluation. The use of separate training and testing sets increases confidence in the presented conclusions.

One of the algorithms that we evaluate, which we call Bell and Wong Nearest Neighbours (BWNN), is the simplest of three algorithms proposed in Bell and Wong (2005) in the context of urban taxi systems. The results show that BWNN very nearly achieves the maximum throughput predicted by the capacity region analysis on several test scenarios, particularly when the fleet size is large. BWNN is also of interest because the proactive EVR algorithms developed in Chapter 4 will be formulated as extensions to BWNN, which is itself a reactive EVR algorithm (that is, it does not move vehicles in anticipation of future requests). For comparison, a second EVR algorithm, which we will call Longest-Waiting-Passenger First (LWPF), is also described. As its name suggests, it prioritises the passenger that has been waiting longest, which is a common theme in the PRT literature (Ford et al., 1972; Andréasson, 2003). Results with the LWPF algorithm will also be compared with results from a proprietary PRT microsimulation to validate modelling assumptions.

The main content of this chapter has been published previously in Lees-Miller et al. (2010), but some additional detail is included here. The idea to try to define the capacity region of a PRT system came from discussions with Prof. F. P. Kelly. The connection between the capacity region and flow conservation was suggested by Prof. R. E. Wilson.

2.1 The Fluid Limit and the Capacity Region

This section works with the *fluid limit*, at the level of long-run average flows of vehicles between stations, rather than at the level of individual vehicles and requests. We will begin by reviewing some results in the fluid limit that

that are well known in the PRT literature (Irving et al., 1978; Anderson, 1978; Andréasson, 1998; Delle Site et al., 2005; Won et al., 2006); they can also be derived as a special case of the urban taxi economics model of Yang et al. (2002). These results will then be interpreted in a new way to give necessary conditions for a given demand to be in a system's capacity region.

Consider a PRT system for which the set of stations, S, the trip times, t_{ij}, and the demand matrix entries, d_{ij}, are defined as in Section 1.2. In the fluid limit, we are interested in the *flows* of occupied and empty vehicles between pairs of stations in this system. These flows are measured in vehicle trips per unit time. The flows of occupied vehicles are simply the request rates from the demand matrix, because we have assumed that there is no ride sharing (that is, each request requires exactly one occupied vehicle trip). These flows of occupied vehicles must be balanced by flows of empty vehicles, denoted x_{ij} ($i, j \in S$; $i \neq j$; $x_{ij} \geq 0$), such that the total (occupied plus empty) flow of vehicles is conserved at each station:

$$\sum_{\substack{j \in S \\ j \neq i}} (d_{ij} + x_{ij}) = \sum_{\substack{j \in S \\ j \neq i}} (d_{ji} + x_{ji}) \tag{2.1}$$

for each station i. That is, the total flow into each station must equal the total flow out, because the fleet size must remain constant. The minimum number of vehicles required to sustain these flows, c, is given by the sum-product of the total flows and the trip times,

$$c = \sum_{\substack{i,j \in S \\ j \neq i}} t_{ij} (d_{ij} + x_{ij}) . \tag{2.2}$$

For example, if the trip time for a particular station pair is $t_{ij} = 3$ minutes, and the chosen empty vehicle flow is $x_{ij} = 5$ vehicle trips per minute, then at least 15 vehicles will be required to sustain this flow.

In what follows, it will help to write these equations using vector notation; let $\mathbf{t}$, $\mathbf{d}$ and $\mathbf{x}$ be column vectors that respectively list the elements t_{ij}, d_{ij}, and x_{ij} in order (for $i \neq j$). For example, $\mathbf{t} = (t_{1,2}, t_{1,3}, \ldots, t_{n_S,n_S-1})'$, where n_S is the number of stations and $'$ (prime) denotes the transpose. Let $\mathbf{b}$ be the vector of occupied vehicle surpluses at each station, with

$$b_i = \sum_{\substack{j \in S \\ j \neq i}} (d_{ji} - d_{ij}) \,. \tag{2.3}$$

Equations (2.1) and (2.2) can then be written succinctly as

$$\mathbf{A}\mathbf{x} = \mathbf{b} \quad \text{and} \quad c = \mathbf{t}'(\mathbf{d} + \mathbf{x}) \tag{2.4}$$

where $\mathbf{A}$ is a matrix with n_S rows and $n_S^2 - n_S$ columns that encodes the flow conservation constraints (2.1); it can be shown that each of its columns contains one $+1$ entry and one -1 entry, and that all other entries are zero.

Because the system has only n_K vehicles in total, it can serve demand $\mathbf{d}$ only if $c \leq n_K$. This inequality and (2.4) give a necessary condition for a demand $\mathbf{d}$ to be in the system's capacity region: there must exist empty vehicle flows $\mathbf{x_d}$ for $\mathbf{d}$ such that

$$\mathbf{A}\mathbf{x_d} = \mathbf{b}, \quad \mathbf{x_d} \geq \mathbf{0}, \quad \text{and} \quad \mathbf{t}'(\mathbf{d} + \mathbf{x_d}) \leq n_K. \tag{2.5}$$

If any such empty vehicle flows $\mathbf{x_d}$ exist, it is clear from (2.5) that any flows $\mathbf{x_d^*}$ (not necessarily unique) that solve the linear program

$$\begin{aligned} \min \quad & \mathbf{t}'\mathbf{x} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{aligned} \tag{2.6}$$

satisfy the conditions in (2.5). That is, $\mathbf{x_d^*}$ is chosen to minimise the number of empty vehicles used, subject to flow conservation.

Problem (2.6) is a special kind of linear program known as a *minimum cost network flow* (MCNF) problem, because of the special structure of the constraint matrix, $\mathbf{A}$ (Bertsimas and Tsitsiklis, 1997, p. 277). The MCNF problem can be solved efficiently using a variety of standard techniques, such as the network simplex method (Bertsimas and Tsitsiklis, 1997, ch. 7). It is also common (for example (Irving et al., 1978, ch. 5.6)) to cast this problem

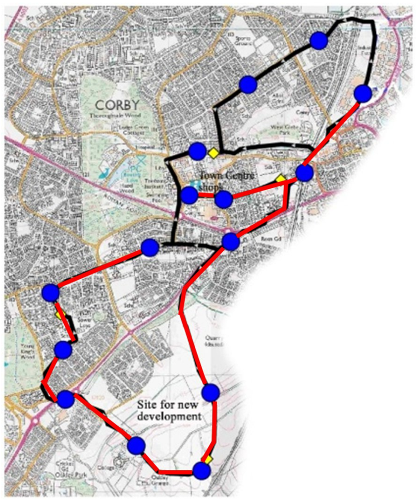


Figure 2.1: For the Corby case study network, an optimal solution to the linear program problem (2.6) sends non-zero empty vehicle flow on the guideways highlighted in red. The flows $\mathbf{x}^*_\mathbf{d}$ for each pair of stations are mapped back onto the network by assuming that vehicles always take the quickest paths between stations. This process also gives an estimate for the vehicle flow (occupied or empty) on each link. Note that the empty vehicle flows never form a cycle; this is a property of any optimal solution to a minimum cost network flow problem (Bertsimas and Tsitsiklis, 1997).

as a *transportation problem*, which is a special case of the MCNF problem. Note that the flows $\mathbf{x}^*_\mathbf{d}$ can be visualised by mapping them back onto the network, as shown in Figure 2.1 for the Corby case study network.

In summary, a necessary condition for a demand $\mathbf{d}$ to be in the capacity region is that

$$\mathbf{t}'(\mathbf{d} + \mathbf{x}^*_\mathbf{d}) \leq n_K. \tag{2.7}$$

It is not known whether (2.7) is also a sufficient condition; that is, there may be systems and demands for which no EVR algorithm can prevent the number of waiting passengers from diverging, even though (2.7) is satisfied. However, simulation results in Chapter 4 show that for some systems, demands and EVR algorithms, the bound (2.7) is very nearly attained.

In what follows, we will treat condition (2.7) as the definition of the capacity region. We will also define the intensity of demand $\mathbf{d}$ as

$$\rho_{\mathbf{d}} = \frac{\mathbf{t}'(\mathbf{d} + \mathbf{x}^*_{\mathbf{d}})}{n_K} \tag{2.8}$$

which is simply the number of vehicles required to serve the demand over the number of vehicles in the fleet. Condition (2.7) is equivalent to requiring $\rho_{\mathbf{d}} \leq 1$. The following examples illustrate the capacity region and intensity concepts on some small networks.

Example 2.1 Consider a system with the two-station ring network shown in Figure 2.2a and a single vehicle. The trip times are t_{12} and t_{21}, the fleet size is n_K, and the demand matrix entries are d_{12} and d_{21}. The demand matrix entries are also the occupied vehicle flows, because we have assumed that there is no ride sharing. For this small example, the required empty vehicle flows can be determined directly as

$$\begin{aligned} x^*_{12} &= \max\{d_{21} - d_{12}, 0\} \\ x^*_{21} &= \max\{d_{12} - d_{21}, 0\} \end{aligned} \tag{2.9}$$

because any surplus in the occupied vehicle flow at one station can only be sent to the other. The number of vehicles required is then given by (2.2) as

$$t_{12}\,(d_{12} + x^*_{12}) + t_{21}\,(d_{21} + x^*_{21})$$

which simplifies to

$$\max\{d_{12}, d_{21}\}\,(t_{12} + t_{21})\,. \tag{2.10}$$

That is, in this case, the number of vehicles required is determined by the larger of the two demands and the total length of the ring. The condition (2.7) is then

$$\max\{d_{12}, d_{21}\}\,(t_{12} + t_{21}) \leq n_K. \tag{2.11}$$

This constraint on the demand matrix entries, d_{12} and d_{21}, defines the system's capacity region, which is shaded in Figure 2.2b. Similarly, the intensity, $\rho_{\mathbf{d}}$, is simply the minimum number of vehicles required (2.10) divided by the

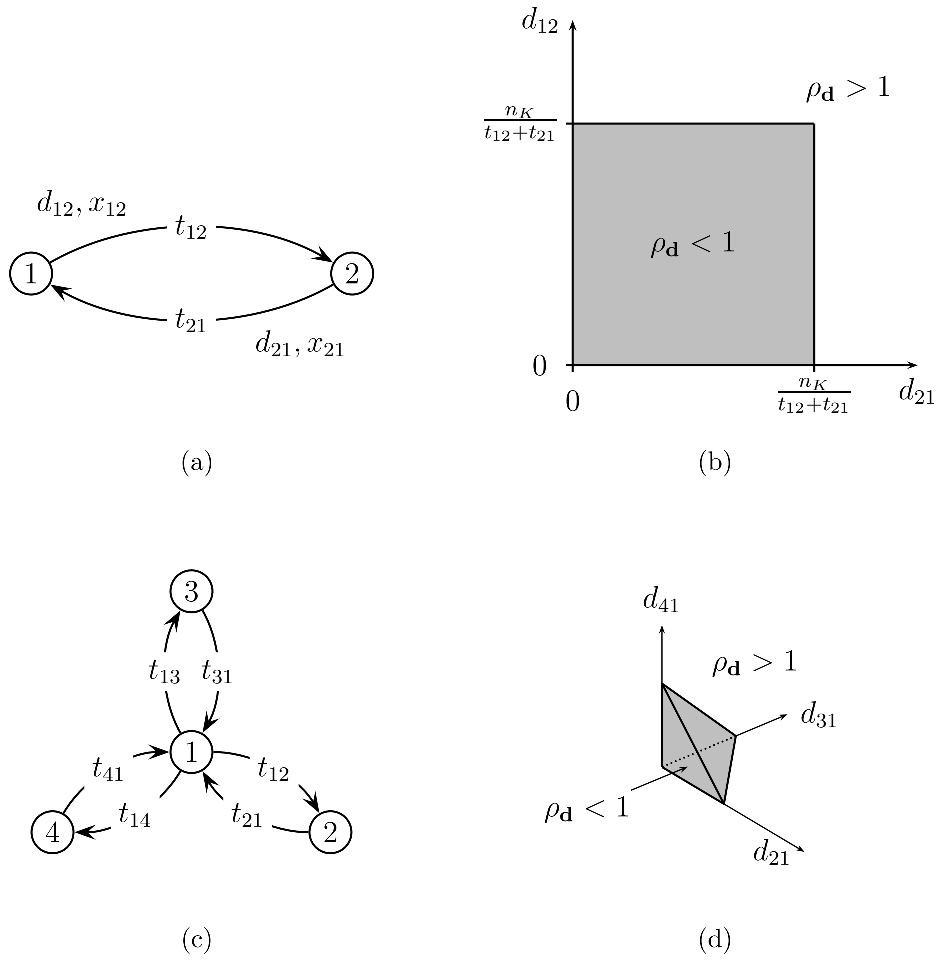


Figure 2.2: A two-station ring (a) and its capacity region (b). A four-station star (c) with three 'spokes' and its capacity region (d) when all of the demand is into the hub and all of the spokes have the same round trip time.

number available, so $\rho_{\mathbf{d}} \leq 1$ in the capacity region. Similar results can be obtained for rings with more than two stations. ■

Example 2.2 The capacity region of the four-station star network in Figure 2.2c has twelve dimensions, because this is the number of entries in the demand matrix (4×4 entries minus the 4 on the diagonal that are defined to be zero). It is therefore not possible to visualise it in its entirety, but we can visualise some special cases, such as when there is tidal demand from the spokes to the hub. In this case, only three of the demand matrix entries can be non-zero, and we can visualise these dimensions.

The demand matrix entries that are allowed to be non-zero are d_{21}, d_{31} and d_{41}. The corresponding empty flows are then simply $x^*_{12} = d_{21}$, $x^*_{13} = d_{31}$ and $x^*_{14} = d_{41}$, because the demand is tidal (one way) from the spokes to hub, and flow conservation requires that the empty vehicles return to the spokes. Condition (2.7) then simplifies to

$$d_{21}(t_{12} + t_{21}) + d_{31}(t_{13} + t_{31}) + d_{41}(t_{14} + t_{41}) \leq n_K \tag{2.12}$$

which defines a plane when satisfied at equality. Together with the non-negativity constraints on the demand matrix entries, this plane defines the boundary of the capacity region, as shown in Figure 2.2d.

Each term in (2.12) is equal to the corresponding minimum number of vehicles required (2.10) for a two-station ring under tidal demand. This means that, so far as the capacity region is concerned, the network decomposes into three independent two-station rings. If the round trip time between the hub and station 2 (say) is increased, the capacity region contracts along the dimension d_{21}, because each trip on that ring takes longer, and therefore ties up a larger fraction of the vehicle's fleet. If the fleet size, n_K, is increased, the capacity region expands proportionally. Similar results can been obtained when the flow is tidal from the hub to the spokes. ■

In both of the examples above, the capacity region is convex. In fact, this is also a general result.

Lemma 2.1 (Convexity) The capacity region is convex. That is, if $\mathbf{d}_1$ and $\mathbf{d}_2$ are demands in the capacity region, then the demand $\alpha\mathbf{d}_1 + (1-\alpha)\mathbf{d}_2$ is in the capacity region for all $\alpha \in [0, 1]$.

PROOF Demands $\mathbf{d}_1$ and $\mathbf{d}_2$ are in the capacity region, so there exist corresponding flows of empty vehicles $\mathbf{x}_1$ and $\mathbf{x}_2$ that satisfy conditions (2.5). Let $\mathbf{d} = \alpha\mathbf{d}_1 + (1-\alpha)\mathbf{d}_2$. To show that demand $\mathbf{d}$ is in the capacity region, we must find a vector $\mathbf{x}_\mathbf{d}$ of empty vehicle flows that also satisfy conditions (2.5). In particular, it can be shown that the vector $\mathbf{x}_\mathbf{d} = \alpha\mathbf{x}_1 + (1-\alpha)\mathbf{x}_2$ is such a vector, because each condition (2.5) is linear. ■

The practical consequence of convexity is that if one has several demands in the capacity region, for example an AM peak demand and a PM peak demand, then any mixture (that is, convex combination) of these will also be in the capacity region. It is also worth noting that the empty vehicle flows $\mathbf{x}_\mathbf{d}$ used in the proof of Lemma 2.1 are not necessarily the optimal flows for the combined demand, $\mathbf{d}$, as illustrated by the following example.

Example 2.3 For the two-station ring network in Figure 2.2a on page 41 with $\mathbf{t} = (1/4, 1/4)'$ and a single vehicle, the demands $\mathbf{d}_1 = (1, 0)'$ and $\mathbf{d}_2 = (0, 1)'$ are both in the capacity region, because they satisfy the inequality (2.11) together with their corresponding optimal empty flows $\mathbf{x}_1^* = (0, 1)'$ and $\mathbf{x}_2^* = (1, 0)'$ from (2.9). Using the notation from the proof of Lemma 2.1, the convex combination of these two demands when $\alpha = 1/2$ is

$$\mathbf{d} = \alpha\mathbf{d}_1 + (1-\alpha)\mathbf{d}_2 = (1/2, 1/2)'$$

and the corresponding convex combination of empty vehicle flows is

$$\mathbf{x}_\mathbf{d} = \alpha\mathbf{x}_1 + (1-\alpha)\mathbf{x}_2 = (1/2, 1/2)'$$

but the optimal empty vehicle flows are $\mathbf{x}_\mathbf{d}^* = (0, 0)'$, because the demand is balanced (that is, because $d_{12} = d_{21}$ in (2.9)). ■

This example shows that adding the optimal empty vehicle flows for two demands does not necessarily give the optimal empty vehicle flows for the combined demand. However, the optimal empty vehicle flows for a given demand do scale in proportion to that demand, as stated in the following lemma.

Lemma 2.2 (Proportionality) Let $\alpha \in \Re$ be a positive constant. If the empty vehicle flows $\mathbf{x}^*_{\mathbf{d}}$ are an optimal solution to the linear program (2.6) with demand $\mathbf{d}$ then the scaled empty vehicle flows $\alpha\mathbf{x}^*_{\mathbf{d}}$ are an optimal solution to the corresponding problem with demands $\alpha\mathbf{d}$. In other words,

$$\mathbf{x}^*_{\alpha\mathbf{d}} = \alpha\mathbf{x}^*_{\mathbf{d}}. \tag{2.13}$$

Proof This follows from well-known sensitivity analysis results for linear programs (Bertsimas and Tsitsiklis, 1997, ch. 5). ∎

Lemma 2.2 implies that the intensity $\rho_{\alpha\mathbf{d}}$ (2.8) of a scaled demand $\alpha\mathbf{d}$ scales in the same way; that is

$$\rho_{\alpha\mathbf{d}} = \frac{\mathbf{t}'(\alpha\mathbf{d} + \mathbf{x}^*_{\alpha\mathbf{d}})}{n_K} = \frac{\mathbf{t}'(\alpha\mathbf{d} + \alpha\mathbf{x}^*_{\mathbf{d}})}{n_K} = \alpha\frac{\mathbf{t}'(\mathbf{d} + \mathbf{x}^*_{\mathbf{d}})}{n_K} = \alpha\rho_{\mathbf{d}}. \tag{2.14}$$

This property will be used to evaluate the throughput of the EVR algorithms introduced in the next section.

2.2 Reactive EVR Algorithms

The preceding analysis of the fluid limit does not include an explicit EVR algorithm. It gives a macroscopic description of the required empty vehicle flows over the long run, but it does not determine the behaviour of the system at the microscopic level, in terms of individual vehicle movements and passenger requests. This section describes two simple EVR algorithms, here called Bell and Wong Nearest Neighbours (BWNN) and Longest-Waiting-Passenger-First (LWPF), that operate at this microscopic level.

2.2.1 Bell and Wong Nearest Neighbours (BWNN)

Following the notation from Section 1.2, let K be the set of vehicles, let S be the set of stations, and let t_{ij} be the trip time from station i to station j. We assume that each vehicle $k \in K$ has a planned route, which consists of a list of stations that it must visit in order. Each pair of adjacent stations defines a trip, during which the vehicle may be occupied or empty. A vehicle's route changes over time: completed trips are deleted from the head, and new occupied or empty vehicle trips are appended to the tail as they are assigned. If a vehicle completes all of the trips in its list, it becomes *idle* at the last station on its route. For simplicity, it is assumed that the lists are not reordered, so for each vehicle k, it is enough to know the last station, d_k, that vehicle k was assigned to visit, and the time, a_k, at which the vehicle will arrive (or has already arrived) at d_k. An idle vehicle is one whose a_k is in the past.

When a request is received, a vehicle is immediately assigned using the following nearest neighbours heuristic, which we call Bell and Wong Nearest Neighbours (BWNN). Each request, r, has associated with it an origin station i_r, a destination station j_r and the time e_r at which the system receives the request. BWNN immediately assigns the vehicle that minimises that request's waiting time, formally

$$k^* = \underset{k}{\operatorname{argmin}} \left[\max(0, a_k - e_r) + t(d_k, i_r)\right] \tag{2.15}$$

where the trip times t_{ij} are written $t(i, j)$ here for readability. The first term, $\max(0, a_k - e_r)$, is the delay before the vehicle can start a new trip; note that the vehicle cannot begin its trip in the past, before e_r. The second term, $t(d_k, i_r)$, is the required empty vehicle trip time; if k is already going to the request's origin station ($d_k = i_r$), then the empty vehicle trip time is zero, because no empty trip is required. Finally, a tie-breaking rule is required for when the argmin in (2.15) is not unique: for simplicity we choose the minimum such vehicle index k.

The state of the assigned vehicle is then updated by setting

$$\begin{aligned} a_{k^*} &\leftarrow \max(e_r, a_{k^*}) + t(d_{k^*}, i_r) + t(i_r, j_r) \\ d_{k^*} &\leftarrow j_r \end{aligned} \tag{2.16}$$

to reflect the fact that vehicle k^*'s planned route now ends when it finishes serving request r at station j_r. The expression

$$\max(e_r, a_{k^*}) + t(d_{k^*}, i_r) \tag{2.17}$$

is the request's *pickup time*, and $t(i_r, j_r)$ is its occupied trip time. Note that minimising the request's waiting time in (2.15) is equivalent to minimising its pickup time (2.17), because $\max(e_r, a_k) = e_r + \max(0, a_k - e_r)$. Figure 2.3 illustrates the handling of vehicle request lists and the operation of the BWNN algorithm on a small example.

2.2.2 Longest-Waiting Passenger First (LWPF)

In contrast with BWNN, the longest-waiting passenger first (LWPF) algorithm requires that each vehicle store only its next destination; once it reaches its destination, the system decides where it should go next. When a vehicle becomes idle at its destination, it is dispatched to the longest-waiting passenger. Precisely, the following steps are carried out at each time step t:

1. Each generated passenger (if there are any) joins the queue at his origin station.
2. For each station i (in order of index, as order does not matter here):
   (a) All vehicles finishing their trips to i at time t become idle at i.
   (b) If there are both waiting passengers and idle vehicles at i, the first passenger is removed from the queue and a vehicle becomes inbound to his destination, j, with arrival time $t + t_{ij}$. This step repeats until there are either no waiting passengers or no idle vehicles at i.

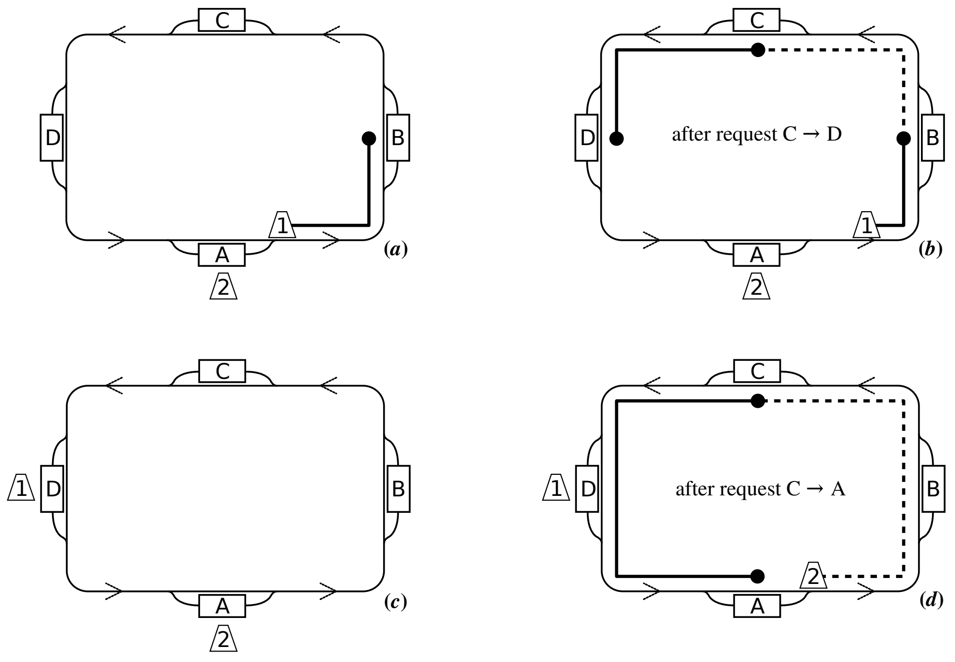


Figure 2.3: Illustration of the BWNN algorithm. There are four off-line stations (labelled A - D) in a ring and two vehicles (labelled 1 and 2). Traffic flow is anticlockwise. (a) Vehicle 1 is initially moving to station B, and vehicle 2 is idle at station A. (b) When a request for travel from C to D is received, vehicle 1 is assigned, because it gives a smaller waiting time than vehicle 2. Both an empty trip (dashed line) from B to C and an occupied trip (solid line) from C to D are appended to vehicle 1's route. Note that vehicle 1 stops at station B and station C (filled circles), but it does not become idle at either station, because it has not finished its route. (c) However, vehicle 1 does become idle at D, because no further requests are assigned to it. (d) When another request is received from C to A, vehicle 2 is assigned, and it begins an empty trip to C. Note that vehicle 2 was idle at station A, so the passenger's waiting time might have been reduced by moving vehicle 2 to station C proactively, before the request was received.

3. For each station i with waiting passengers, let h_i be the arrival time of the longest-waiting passenger.

4. For each station i with more waiting passengers than idle plus inbound vehicles, in ascending order by h_i, consider the stations $j \neq i$ in ascending order by t_{ji} (breaking any ties randomly); choose the first station (if any) that has more idle empty vehicles than waiting passengers, and send an empty vehicle from this station to station i.

By prioritising the longest-waiting passenger, LWPF tries to flatten the tail of the waiting time distribution; however, it will be seen in Section 2.4 that this adversely affects its maximum throughput.

2.2.3 Discussion

The BWNN and LWPF algorithms differ in how they decide which vehicles to assign to incoming requests, and also in when they make these decisions. The BWNN algorithm uses *immediate assignment*, because a vehicle is immediately assigned to each request when it is received, and this assignment is never changed. The LWPF algorithm uses *delayed assignment*, because a vehicle is only assigned to a particular request when it arrives at the request's origin (in step 2(b), above). The two algorithms therefore use different state representations, and they make subtly different assumptions about how the system operates. This section discusses these differences.

BWNN assumes that the destination of each request is known when it is received. This is because queues of passenger requests are stored implicitly in the vehicles' routes. At higher demand intensities, the vehicles' planned routes extend further into the future, as they accommodate more queued requests. The destination of each request must be known in order to update the assigned vehicle's a_k time (2.16). This is consistent with a system in which the passengers select their destinations at the entrance to the station, before they begin queueing. If destinations are instead selected at the berths, then the number of requests at a given origin for which the system knows destinations is limited by the number of berths at the station. This is not important at low intensity, because the number of queued requests is small, but the model is less accurate at very high intensity, when destinations are selected at berths. A similar issue arises if destinations are selected in vehicles.

Both BWNN and LWPF ensure that requests from the same origin station are served in first-come-first-served order. For LWPF, this follows immediately from step 2(b). LWPF also guarantees that no vehicle can start an empty trip from a station at which there are waiting passengers; BWNN, however, does not have this property. It may happen that a passenger arrives at some station after BWNN has assigned an inbound vehicle to an earlier request from another station. This is again unlikely at low intensity, but this behaviour may occur at higher intensities.

In principle, an algorithm that uses delayed assignment should be able to perform better than an algorithm that can make only immediate assignments, because it has more flexibility to adjust to new requests. For example, a delayed assignment version of the BWNN algorithm could be created by allowing re-optimisation of vehicle routes after the initial assignment. When demand intensity is high, and the vehicles' routes contain trips for several requests, it might be possible to reduce empty vehicle travel and passenger waiting times by reordering or swapping requests between vehicles. In particular, it has been shown that re-optimising the destinations of empty PRT vehicles that are already moving can be beneficial (Andréasson, 2003), assuming that the control system is capable of rerouting vehicles (Section 1.1.4). Immediate assignment is therefore a pessimistic assumption.

On the other hand, immediate assignment does have some advantages. One is that the system can tell each passenger when they will be picked up, as soon as they request service, which may make their waiting time less onerous. It is interesting to note that immediate assignment is the norm for lifts (elevators) in Japan, for this reason (Strakosch, 1998), and also because it allows passengers to wait at the correct elevator shaft. Immediate assignment also avoids complications such as the "indefinite deferment" of requests at outlying stations (Psaraftis, 1995); this refers to a situation where greedy re-optimisation of vehicle routes causes some requests never to be served, even though the system has spare capacity.

2.3 Training and Testing Data

To evaluate the performance of the EVR algorithms developed in this thesis, a variety of scenarios have been collected from PRT system 'case studies' in the literature or synthesised to test particular features. Here a *scenario* is defined by a trip time matrix, a demand matrix and a fleet size; note that under the assumptions in Section 1.2, the network topology is not important. The scenarios are partitioned into *training* scenarios, which were used to guide the development of the algorithms, and *testing* scenarios, which were used only for the final evaluation, after the details of the algorithms had been finalised. The use of a separate test set provides increased confidence that conclusions about the performance of the proposed algorithms will generalise to scenarios other than those for which they were developed.

2.3.1 Training Scenarios

The training scenarios consist of a mixture of case studies and synthetic scenarios. The full set of data files is provided on the compact disc that accompanies this thesis. We will frequently make use of the *Corby* and *Grid* case studies described below, because they are representative examples.

The Corby scenario is taken from the case study described in Bly and Teychenne (2005). The network layout (Figure 1.4a on page 23) and demand used in this study are both publicly available as part of the ATS/CityMobil PRT simulator (ULTra PRT, 2008). The demand matrix represents the AM peak for phase one of the proposed system. There are 15 stations. The fleet size is set to 200 vehicles, as is estimated in the case study.

The Grid scenario is synthetic. The network is a regular grid of one-directional guideways with 24 stations located at the line midpoints (Figure 1.4b on page 23); this idealised topology appears several times in the PRT literature (Irving et al. (1978), ch. 2, for example). Lines are spaced at 800m (0.5mi) to provide 400m (0.25mi) maximum walk distances. Assuming 10m/s (22mph) average speed, adjacent stations are 80s apart, and the maximum station-to-station travel time is 12 minutes (e.g. from B to G). The demand matrix for the grid scenario is obtained from a standard gravity model (Chakroborty and Das, 2003) with

$$d_{ij} = \begin{cases} a_i b_j o_i d_j \exp(-\theta t_{ij}) & i \neq j \\ 0 & i = j \end{cases} \tag{2.18}$$

where o_i and d_j are the desired total origin and destination flows, θ is the dispersion parameter, and t_{ij} is the shortest path trip time, in seconds, from i to j. The aim of the gravity model is to find d_{ij} of the form (2.18) that satisfy the constraints

$$o_i = \sum_j d_{ij} \quad \text{and} \quad d_j = \sum_i d_{ij}. \tag{2.19}$$

This is done by finding suitable values for the a_i and b_j coefficients; in particular, (2.18) and (2.19) imply that

$$a_i = \left(\sum_{\substack{j \\ j\neq i}} b_j d_j e^{-\theta t_{ij}} \right)^{-1} \quad \text{and} \quad b_j = \left(\sum_{\substack{i \\ i\neq j}} a_i o_i e^{-\theta t_{ij}} \right)^{-1}, \tag{2.20}$$

which can be solved by fixed-point iteration. The o_i and d_j are chosen to represent an 'AM peak,' as given in Table 2.1. The θ parameter is initially set to 0.01; other values of θ, which generate different demand patterns, are considered in Lees-Miller et al. (2010). The fleet size is set at 200 vehicles.

2.3.2 Testing Scenarios

Six testing scenarios were chosen from a set of thirty scenarios created by second-year engineering design students in a ten-hour course module. The students worked in groups of roughly four; each group was asked to design a PRT system on a site of their choosing. They estimated several demand matrices for different times of day based on the major demand generators

	5.0	5.0	5.0	
5.0	3.8	3.8	5.0	
	3.8	2.5	3.8	
5.0	2.5	2.5	5.0	
	3.8	2.5	3.8	
5.0	3.8	3.8	5.0	
	5.0	5.0	5.0	

(a) Origin flow % (o_i)

	0.8	0.8	0.8	
0.8	3.8	3.8	0.8	
	3.8	15.0	3.8	
0.8	15.0	15.0	0.8	
	3.8	15.0	3.8	
0.8	3.8	3.8	0.8	
	0.8	0.8	0.8	

(b) Destination flow % (d_j)

Table 2.1: Total flows for the Grid scenario gravity model. Table layouts correspond to the station layout in Figure 1.4b. For example, the top left station (labelled J) is the origin of 5.0% of passenger requests and the destination for 0.8%.

Name	**Fleet Size**	**Requests/hour at** $\rho_\mathbf{d} = 1$
T1	504	1345
T2	64	857
T3	134	1330
T4	349	1294
T5	116	1020
T6	55	914

Table 2.2: Basic data for the testing scenarios. Requests/hour at $\rho_\mathbf{d} = 1$ refers to the total request rate (summed over all station pairs) when the demand is scaled to intensity (2.8) one.

for their site. They designed the networks using the ATS/CityMobil PRT network design and simulation tool (ULTra PRT, 2008). To guide the design, a generalised cost function was provided that included infrastructure costs, vehicle costs and passenger waiting times, as estimated in simulation.

The six testing scenarios were selected with a preference for demands in which there were many stations with an overall deficit of empty vehicles, as given by (2.3). This choice was based on the observation that such demands are more challenging for the proactive algorithms described in Chapter 4. The scenarios cover a range of total demands and fleet sizes, as listed in Table 2.2. Figure 2.4 shows the networks and Figure 2.5 shows the demand matrices.

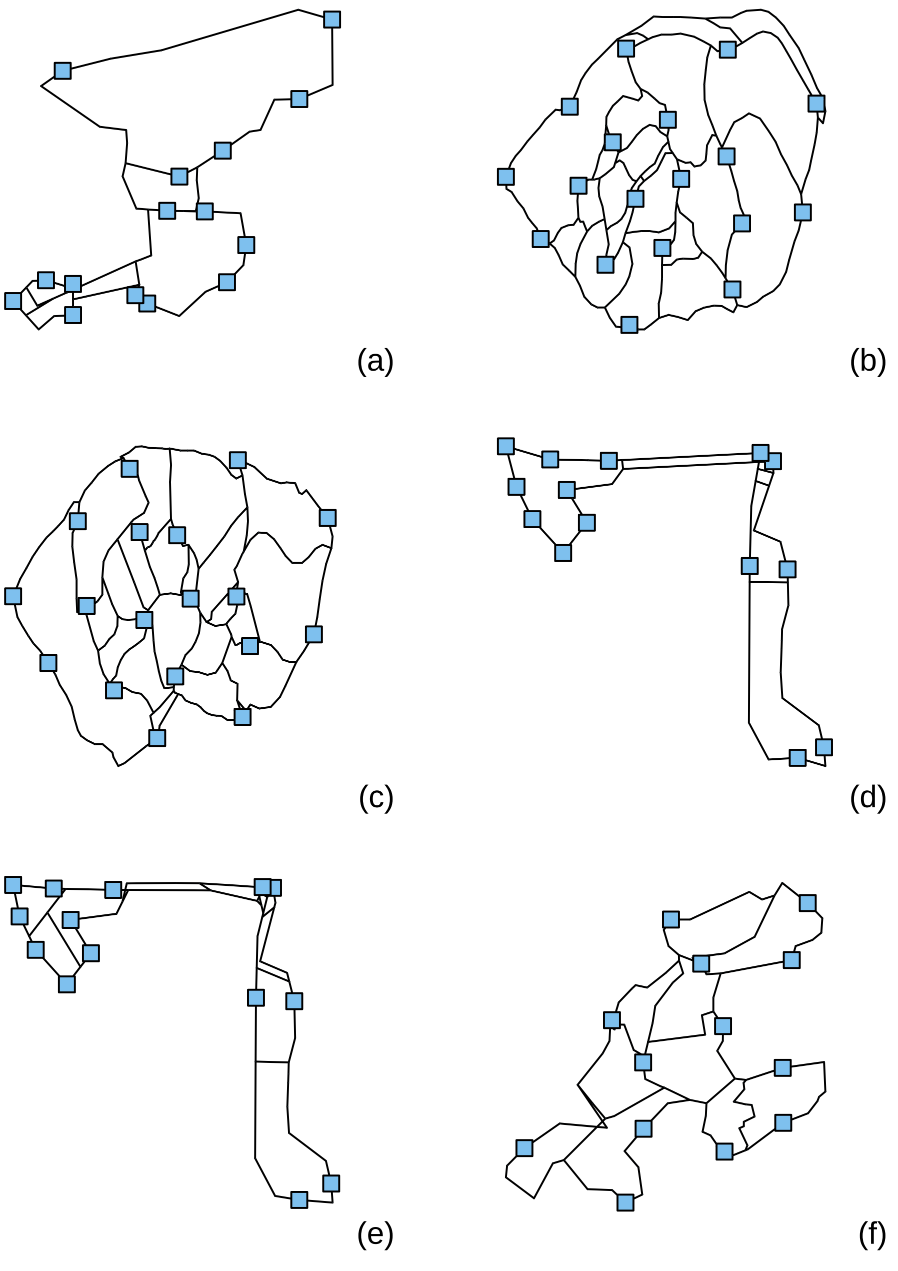


Figure 2.4: Networks for the testing scenarios T1 – T6 ((a) – (f)). Stations are marked with blue squares. Scenarios T2 and T3 have similar networks, because they were created by members of the same group; the same is true of scenarios T4 and T5. However, the scenarios have different demands, as shown in Figure 2.5.

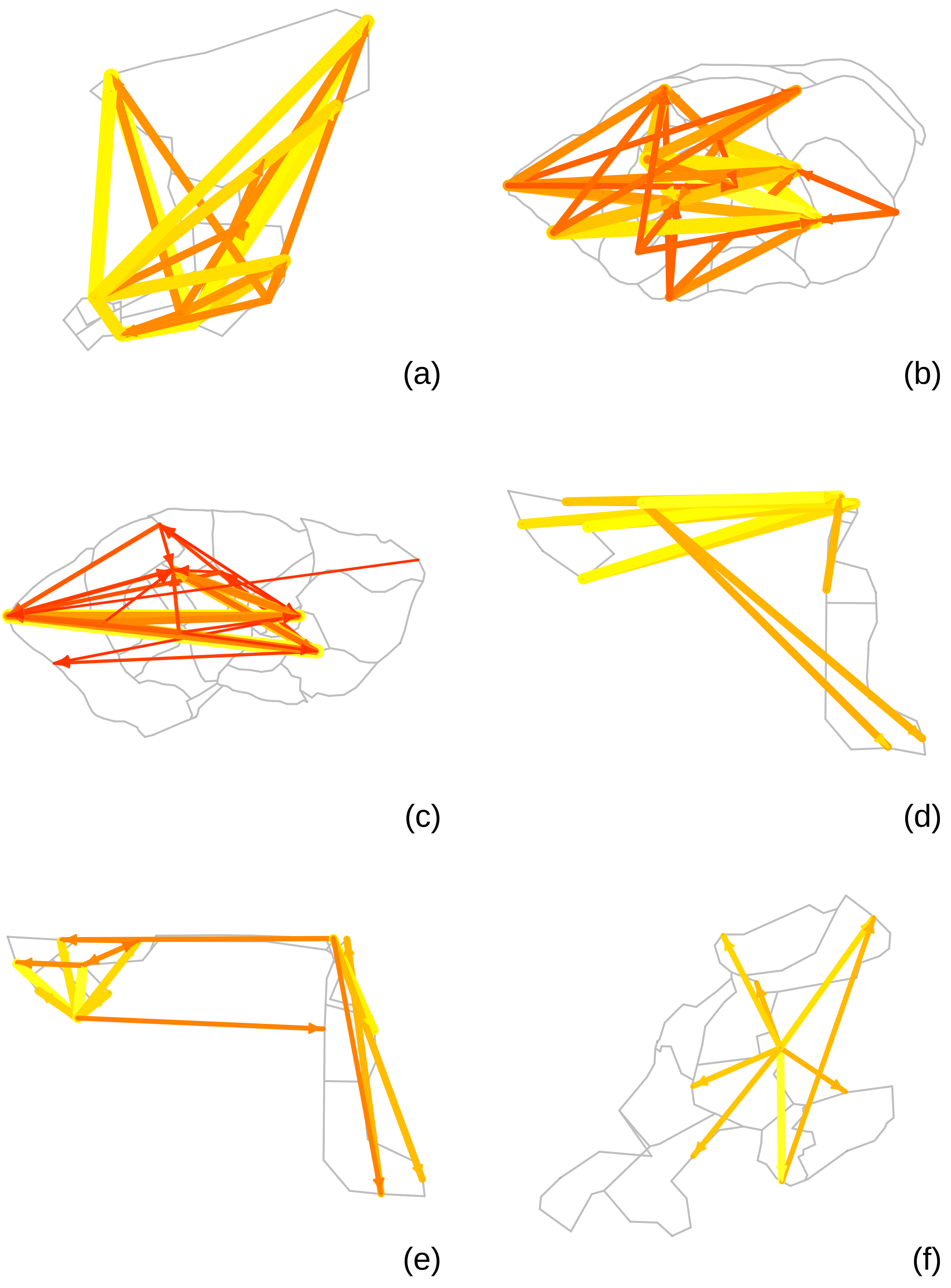


Figure 2.5: Desire line diagrams for the testing scenarios T1 – T6 ((a) – (f)). Each arrow corresponds to an entry in the demand matrix for the given scenario; arrows are only shown for the largest 10% of entries, to reduce clutter. Thicker, brighter arrows indicate more demand.

2.4 Evaluation of Throughput

The BWNN and LWPF algorithms have been implemented in simulation. Passenger requests from station i to station j are generated according to a Poisson process with rate d_{ij}. For convenience, request and trip times are rounded up to the next integer second. The main outputs of interest here are passenger waiting times and vehicle utilisation, which is the fraction of the vehicle fleet that is moving (that is, not idle) in each time step. The steady state distributions of these outputs are estimated by running long simulations with the demand matrix held constant for each run.

To benchmark an EVR algorithm in terms of throughput, we use it in a series of simulation runs, each with a more intense demand than the previous run, and measure the *saturation intensity*, which is the lowest intensity at which the number of waiting passengers begins to diverge. The benchmark is saturation intensity one, which corresponds to maximum theoretical throughput.

To obtain a series of progressively more intense demands, we take the initial demand, $\mathbf{d}$, for a given scenario and scale it up or down to produce a family of demands with different intensities but the same spatial distribution. In particular, each run uses a scaled demand $\alpha\mathbf{d}$ for a positive constant α, and the desired family of demands is obtained by sweeping α from near zero up to $\rho_{\mathbf{d}}^{-1}$. This is because the intensity, $\rho_{\alpha\mathbf{d}}$, of the scaled demand satisfies $\rho_{\alpha\mathbf{d}} = \alpha\rho_{\mathbf{d}}$ by (2.14), and $\alpha\rho_{\mathbf{d}} = 1$ when $\alpha = \rho_{\mathbf{d}}^{-1}$. Geometrically, $\mathbf{d}$ defines the direction of a ray in $n_S^2 - n_S$ dimensions, and $||\mathbf{d}||/\rho_{\mathbf{d}}$ is the distance from the origin to the boundary of the capacity region in the direction of $\mathbf{d}$.

2.4.1 Results for BWNN and LWPF

Figure 2.6 shows the simulation results. Intensity is increased in increments of 0.01, and each point is based on data from 10 independent trials. Each trial consists of a 10-hour (in simulated time) warm up period, in which no statistics are collected, followed by 80 simulated hours of statistics collection. Running the simulation for a long time makes the saturation intensity easier to identify, because the observed queue lengths and mean waiting times in-

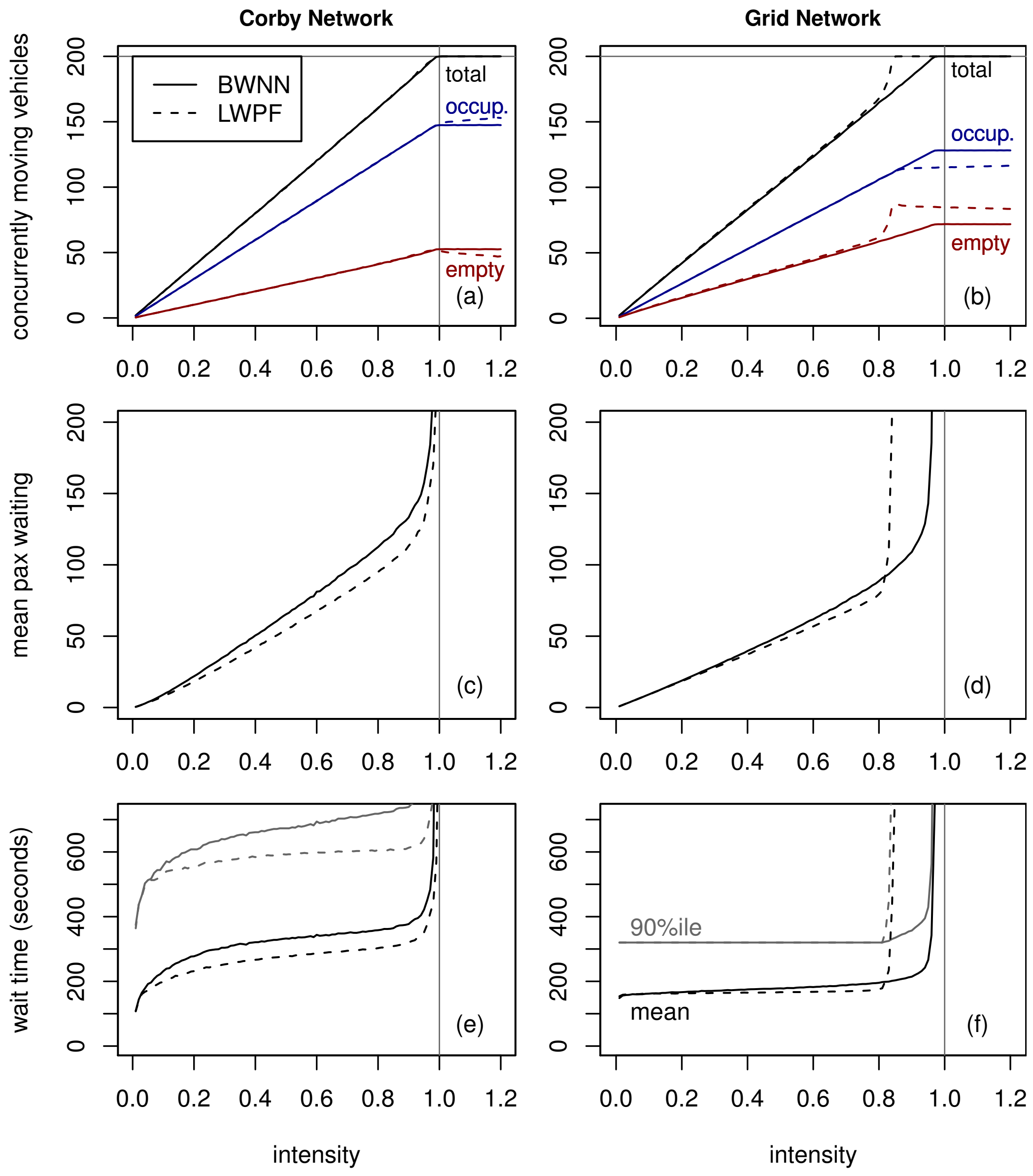


Figure 2.6: Simulation results for the BWNN and LWPF EVR algorithms. Saturation intensities are similar for the Corby network but different for the Grid network; LWPF shows higher empty vehicle use when there are passengers waiting at many stations. Until divergence, LWPF gives lower waiting times and queue lengths. Intensity 1.0 corresponds to 1414 requests/hour on Corby and 2035 requests/hour on Grid; normalising using theoretical capacity helps comparison between different networks.

crease in proportion to the running time when the queue is diverging. The fleet utilisation (Figures 2.6a and 2.6b) also reaches its maximum at the saturation intensity.

Figure 2.6a shows that both algorithms saturate very close to the predicted intensity on the Corby network, because the number of concurrent vehicles reaches the fleet size near intensity 1.0. Figure 2.6b shows the same measure for the Grid network; in this case, the EVR algorithms both saturate at intensities less than 1.0 (LWPF at 0.85 and BWNN at 0.96). This is also apparent in Figures 2.6d and 2.6f, where the mean number of waiting passengers and their waiting times diverge at roughly the same intensities. It appears that neither LWPF nor BWNN attains the theoretical maximum throughput for all networks and demands; it is not yet known whether there is any practical algorithm that does.

One notable feature of Figure 2.6b is that for LWPF the number of concurrent empty vehicles increases suddenly at intensity 0.80. This increase in concurrent empty vehicles prevents an increase in the number of concurrent occupied vehicles, since at intensities above 0.80, the system is serving the same number of passengers with more empty vehicle movement. The reason is that, when a vehicle becomes idle, it must serve the longest-waiting passenger, regardless of his location in the network. When there are standing queues at many of the stations, the average empty vehicle trip may be significantly longer for LWPF than for BWNN.

Figures 2.6e and 2.6f show long waiting times even when intensity is near zero, and they increase only slowly with intensity. This is because the EVR algorithms used here are reactive EVR algorithms – they do not move vehicles in anticipation of future passengers. For example, even if there is tidal flow from an origin i to a destination j, vehicles stay at j until a passenger arrives at i and requests a vehicle, so all passengers wait at least t_{ji}, regardless of the intensity. In this case, it is clear that the system should move vehicles back to i. Proactive EVR algorithms that handle the general case are developed in Chapter 4.

2.4.2 Results on the Test Scenarios

Figure 2.7 shows simulation results with BWNN on the test scenarios (Section 2.3.2). The results are consistent with those obtained on the training scenarios. Saturation intensities range from 0.93 to very nearly 1.0, so BWNN comes close to attaining the theoretical maximum intensity. The scenarios with the lowest saturation intensities are also those with the smallest vehicle fleets. Scenarios T2 and T6 saturate at intensities 0.93 and 0.94, respectively, and have 64 and 55 vehicles, respectively, whereas scenario T5 has 116 vehicles, and it saturates at intensity 0.97. This indicates that the nearest neighbour approach becomes more effective, in terms of throughput, as the fleet size increases.

Bell and Wong (2005) also describe and test two variants on BWNN, called H1 and H2, that modify the BWNN assignment rule (2.15) to consider forecasted future demand. Like BWNN, H1 and H2 are reactive, because they move vehicles only in response to requests that have already been received; however, they may be able to reduce waiting times at high intensity by reducing empty vehicle use, thereby increasing the saturation intensity. The H1 and H2 heuristics have also been evaluated on the test scenarios. H1 measurably increases the saturation intensity for the T2 and T6 networks, which have the smallest fleet sizes and the lowest saturation intensities for BWNN, when its α parameter is around 0.5. No measurable change in saturation intensity was observed for H2.

2.4.3 The Effects of Line Congestion

The analysis and simulation done so far has assumed that line capacity is infinite. There are certainly networks and demands for which this is a poor assumption, so it is prudent to check these results against a more detailed simulator that includes line congestion. Here, a proprietary simulator developed by ULTra PRT Ltd is used.

The line congestion delays for a given demand depend on the control system used to maintain safe separation between vehicles. This control system includes both scheduling and routing components. Because PRT systems are

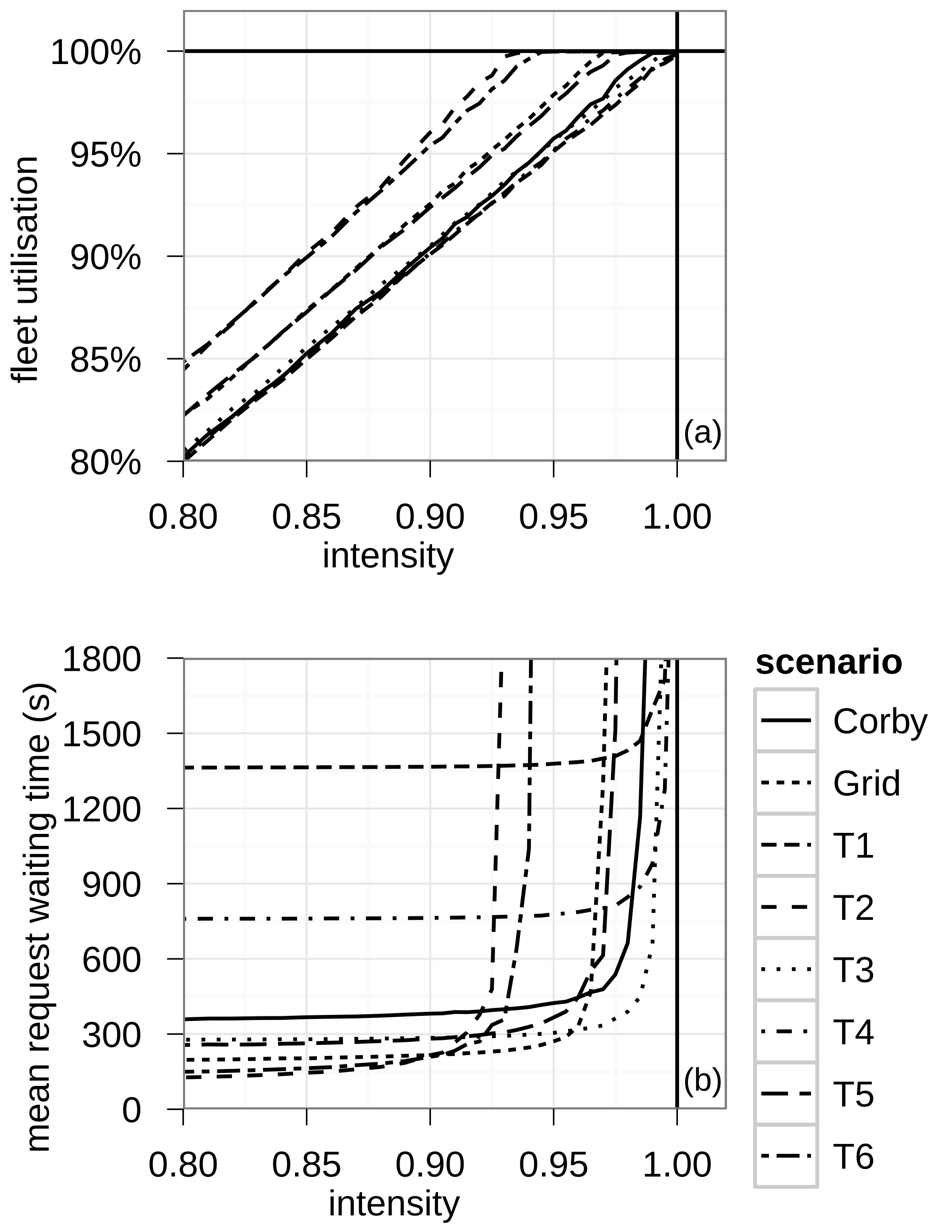


Figure 2.7: Fleet utilisation (a) and mean waiting times (b) for BWNN on the test scenarios at intensities above 80%. As on the training data, BWNN delivers high throughput; saturation intensities are consistently above 0.9. Here intensity is increased in increments of 0.01, and each point is the average of five runs with 10^5 requests per run.

network systems, it is often possible to route around congestion. However, the algorithms for accomplishing this are beyond the scope of this thesis.

The proprietary simulator has been configured to use simple synchronous control (Irving et al., 1978, pp. 92–94). As described in Section 1.1.4, synchronous control means that a central computer schedules each vehicle's whole trip before it leaves its origin station, ensuring that the chosen path does not conflict with that of any previously scheduled trip; vehicles are responsible only for staying on their allocated schedules. Intelligent routing has been disabled, so vehicles are restricted to the path with the smallest free-flow time. This gives a pessimistic estimate of the line throughput that is realistically achievable

The proprietary EVR algorithm has been configured to closely (but not exactly) match the LWPF algorithm. The curves in Figures 2.8a and 2.8b are very similar to those in Figures 2.6c and 2.6d when the minimum headway is 1s; in particular, the saturation intensities are roughly the same. When the minimum headway is increased, the line capacity is decreased, so delays due to line congestion become more likely; these delays contribute to the overall trip times (effectively increasing the t_{ij}), which causes the number of waiting passengers to diverge at lower intensity. In this case, reducing line capacity by a factor of 2 (or more, in the Grid network) produces only small changes in Figure 2.8. Of course, this might not be the case with larger fleet sizes or smaller travel times.

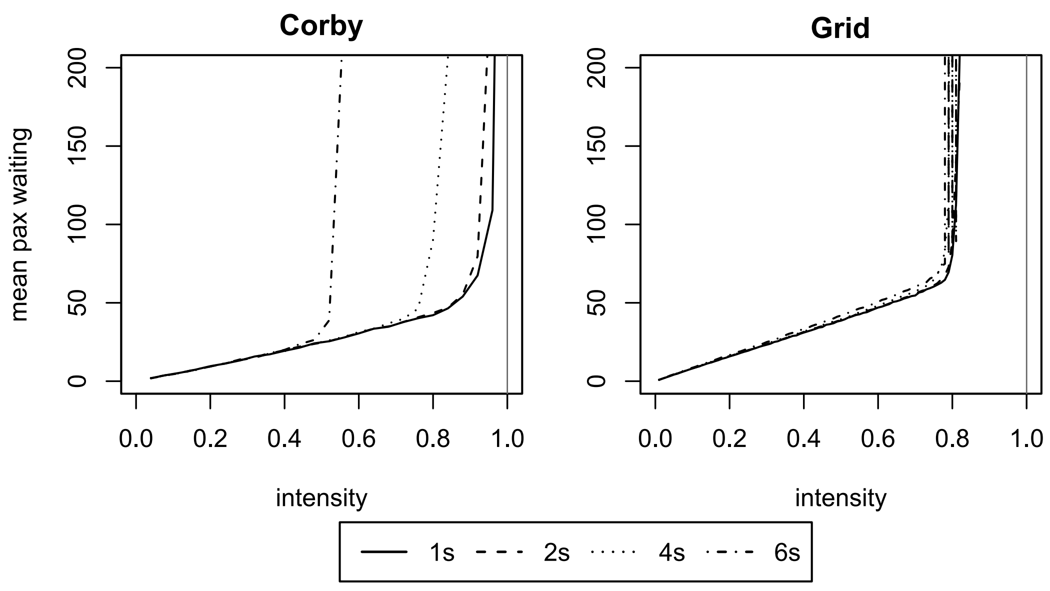


Figure 2.8: Effect of line congestion on mean queue length for varying minimum headway (Section 1.1.4). Queues diverge at the same intensities as in Figure 2.6 on page 56, validating model assumptions. The Grid network is operating far below maximum line capacity with 200 vehicles; more could be added. The Corby system is closer to line capacity: if the minimum headway exceeds 2s, delays due to line congestion reduce throughput.

Chapter 3

Benchmarks for Passenger Waiting Time

The motivating question for this chapter is, 'what is the lowest mean passenger waiting time possible for a given scenario?' We will develop two approaches that give partial answers to this question. Each provides a benchmark against which the mean waiting times obtained by particular EVR algorithms can be compared, in order to estimate the potential for further improvement.

The first approach (Section 3.1) combines the results of the fluid limit analysis in Chapter 2 with a standard queueing model called an M/G/s queue. In particular, the queueing model is used to estimate the passenger waiting time distribution from the demand intensity (2.8). This queueing approximation is exact (under the assumptions in Section 1.2) when all requests originate from a single station. When there are multiple origin stations, the queueing approximation tends to be optimistic; that is, it underestimates the achievable passenger waiting times.

The second approach (Section 3.2) is to consider a *static* version of the EVR problem. The actual EVR problem is *dynamic*, meaning that new requests are received as the system operates, and *stochastic*, because there is uncertainty about where and when future requests will be received. In the static problem, we instead consider a fixed set of future requests, each with a known origin, destination and time of receipt, and we aim to minimise the total waiting time for these requests. An optimal solution for an instance of the static problem provides a benchmark for waiting times; in particular, no algorithm for the dynamic EVR problem, in which information about the future is available only at a statistical level, can do better on that instance. Comparison of the performance of dynamic and static problems is known as competitive analysis (Mitrovic-Minic et al., 2004; Berbeglia et al., 2010). The static EVR problem is closely related to several well-studied problems in combinatorial optimisation and vehicle routing. It will be seen that the static EVR problem is NP-hard, but exact solutions can be obtained for small instances. For larger instances, a simple heuristic, here called Static Nearest Neighbours (SNN) is given; SNN is structurally similar to the BWNN heuristic developed in Section 2.2.1. It will be seen that SNN can often find solutions with zero waiting time (which are clearly optimal) when the intensity of the demand is low or moderate (around 0.8 on the training scenarios). (The SNN heuristic is also used later as a part of the Sampling and Voting algorithm introduced in Chapter 4.)

The queueing model in Section 3.1 has not been published. The use of the static EVR problem and the SNN heuristic for benchmarking has been published in Lees-Miller and Wilson (2011). Some of the details of the static EVR problem (in particular Section 3.2.1) have been described in Lees-Miller and Wilson (2012).

3.1 An M/G/s Queueing Model

Queueing models are used to model a wide variety of systems. An illustrative and historically important example is a telephone call centre. The call centre has a given number of agents that serve incoming calls. If a call is received and there is an available agent, the agent immediately answers the call; otherwise, the call is queued. Calls in the queue are answered in the order in which they are received, as agents become available. The time at which each call is received and the time needed to serve the call, once an agent has answered it, are both random. We are interested mainly in the time that a call must wait in the queue before it is answered, which is also random; its distribution is determined by (i) how quickly calls are being received, (ii) how quickly each call can be served once it has been answered, and (iii) the number of agents.

One way to apply such a queueing model to a PRT system is to treat passenger requests as calls and vehicles as agents. The rate at which requests are received is set by the demand matrix. The service time for a request consists of an occupied trip time and an empty trip time. The distribution of the occupied trip time is determined by the demand matrix and the network travel times, and a distribution for the empty trip time can be computed from an optimal solution to the linear program (2.6), as described below.

Consider a scenario with trip times t_{ij}, fleet size n_K, and a fixed demand with demand matrix entries d_{ij}, as in Section 1.2. Let x^*_{ij} denote the empty vehicle flow from station i to station j in an optimal solution to the linear program (2.6). To simplify notation, let

$$d_{i\sigma} = \sum_{\substack{j\in S\\ i\neq j}} d_{ij}, \quad d_{\sigma j} = \sum_{\substack{i\in S\\ i\neq j}} d_{ij}, \quad \text{and} \quad d_{\sigma\sigma} = \sum_{\substack{i,j\in S\\ i\neq j}} d_{ij} \tag{3.1}$$

denote the row, column and total sums of the demand matrix, and define $x^*_{i\sigma}$, $x^*_{\sigma j}$, and $x^*_{\sigma\sigma}$ similarly.

In what follows, we will exclude stations that have no demand; this is because a station with no demand also has no empty vehicle flow in any optimal solution to the linear program (2.6), so a vehicle that moves according to this optimal solution will never stop there, so it has no effect. More formally, if $d_{i\sigma} = d_{\sigma i} = 0$ for some station i, then no empty vehicle flow is required to satisfy flow conservation (2.1), and the fact that the trip times satisfy the strict triangle inequality (1.1) guarantees that $x^*_{i\sigma} = x^*_{\sigma i} = 0$ in any optimal solution.

3.1.1 Random Walk Model for Empty Vehicle Trips

Each request requires one occupied vehicle trip, with origin and destination as determined by the request, and zero or more empty vehicle trips to the origin of the next request that happens to be assigned to the same vehicle.

The sequence of empty trips can be viewed as a random walk through the network. However, we will see that if an empty vehicle moves according to the flows x^*_{ij}, then this random walk consists of at most one trip. The distribution of the total (occupied plus empty) time required to serve a single request can then be computed.

The probabilities that govern the random walk are as follows. Consider a vehicle that has just completed an occupied trip to station $j \in S$; it may now make an empty trip to another station, or it may stay at j. Define

$$p_{jh} = \frac{x^*_{jh}}{d_{j\sigma} + x^*_{j\sigma}} \tag{3.2}$$

as the probability that the vehicle's next trip is to station $h \in S$, when $h \neq j$. The denominator in (3.2) is the total (occupied plus empty) flow out of station j; it is guaranteed to be positive, because we have excluded stations with no demand. The probability that the vehicle stays at j is then set to

$$p_{jj} = 1 - \sum_{\substack{h \in S \\ j \neq h}} p_{jh} \tag{3.3}$$

so that the p_{jh} sum to one.

To establish that the empty vehicle's random walk will consist of at most one trip, we use the following properties of the flows.

Lemma 3.1 If the trip times t_{ij} satisfy the strict triangle inequality (1.1), then any station with empty vehicle flow in has no empty flow out; that is,

$$x^*_{\sigma j} > 0 \implies x^*_{j\sigma} = 0 \tag{3.4}$$

for each station $j \in S$.

Proof Suppose that both $x^*_{\sigma j} > 0$ and $x^*_{j\sigma} > 0$. In this case, there must exist stations $i \in S$, with $i \neq j$ and $x^*_{ij} > 0$, and $h \in S$, with $h \neq j$ and $x^*_{jh} > 0$. Let $\epsilon = \min\{x^*_{ij}, x^*_{jh}\}$ denote the flow from i through j to h. The contribution of this flow to the objective function $\mathbf{t}'\mathbf{x}$ of the linear program

(2.6) is then $\epsilon(t_{ij} + t_{jh})$. However, the strict triangle inequality implies that

$$\epsilon(t_{ij} + t_{jh}) > \epsilon t_{ih}$$

so the objective value would be reduced if the flow went directly from i to h, without going through j. This contradicts the premise that we have an optimal solution. ■

The number of trips in the vehicle's random walk is then as follows. There are zero trips if $x^*_{jh} = 0$ for all $h \neq j$, since this implies $p_{jh} = 0$ for all $h \neq j$, and $p_{jj} = 1$. Otherwise, we have $x^*_{\sigma h} > 0$, and Lemma 3.1 implies that $x^*_{h\sigma} = 0$, so there is no empty flow out of h, and the vehicle's random walk ends there, after one trip.

3.1.2 The Service Time Distribution

We have now established that each passenger request requires one occupied vehicle trip and at most one empty trip. Let the random variables I and J denote the origin station and destination station of the request, respectively, and let the random variable H denote the destination of the empty vehicle trip (which may be J). The service time of the request is then a random variable

$$\tau = t_{IJ} + t_{JH} \tag{3.5}$$

with distribution defined by the joint distribution of I, J and H.

The demand matrix defines the joint probability distribution of I and J, namely

$$\Pr(I = i, J = j) = \frac{d_{ij}}{d_{\sigma\sigma}} \tag{3.6}$$

for $i \in S$ and $j \in S$. The distribution of the empty trip's destination, H, conditional on the occupied trip's destination, J, is

$$\Pr(H = h | J = j) = p_{jh} \tag{3.7}$$

for p_{jh} as defined in (3.2) and (3.3). The full joint distribution is then

$$\begin{aligned}\Pr(I=i,J=j,H=h) &= \Pr(H=h|I=i,J=j)\Pr(I=i,J=j)\\ &= \Pr(H=h|J=j)\Pr(I=i,J=j) \qquad (3.8)\\ &= \frac{d_{ij}p_{jh}}{d_{\sigma\sigma}}.\end{aligned}$$

In the next section, we will be interested mainly in the expected service time, $\mathrm{E}[\tau]$. By the linearity of expectation, we have $\mathrm{E}[\tau] = \mathrm{E}[t_{IJ}] + \mathrm{E}[t_{JH}]$. For the expected occupied trip time,

$$\begin{aligned}\mathrm{E}[t_{IJ}] &= \sum_{i\in S}\sum_{j\in S} t_{ij}\Pr(I=i,J=j) \qquad (3.9)\\ &= \sum_{i\in S}\sum_{j\in S} t_{ij}(d_{ij}/d_{\sigma\sigma})\\ &= \frac{\mathbf{t}'\mathbf{d}}{d_{\sigma\sigma}}.\end{aligned}$$

For the expected empty trip time, we need the joint distribution of J and H, which is

$$\begin{aligned}\Pr(J=j,H=h) &= \Pr(H=h|J=j)\Pr(J=j)\\ &= \Pr(H=h|J=j)\left(\sum_{i\in S}\Pr(I=i,J=j)\right)\\ &= p_{jh}\left(\sum_{i\in S}\frac{d_{ij}}{d_{\sigma\sigma}}\right) = \frac{d_{\sigma j}p_{jh}}{d_{\sigma\sigma}}\end{aligned}$$

where the second equality comes from summing out the possible origin stations, I. The expected empty trip time is then

$$\begin{aligned}
\mathrm{E}[t_{JH}] &= \sum_{j\in S}\sum_{h\in S} t_{jh}\frac{d_{\sigma j}p_{jh}}{d_{\sigma\sigma}} \qquad (3.10)\\
&= {d_{\sigma\sigma}}^{-1}\sum_{j\in S} d_{\sigma j}\left(\sum_{\substack{h\in S\\ h\neq j}} t_{jh}p_{jh} + t_{jj}p_{jj}\right)\\
&= {d_{\sigma\sigma}}^{-1}\sum_{j\in S} d_{\sigma j}\sum_{\substack{h\in S\\ h\neq j}} t_{jh}\left(\frac{x_{jh}}{d_{j\sigma}+x^*_{j\sigma}}\right)\\
&= {d_{\sigma\sigma}}^{-1}\sum_{j\in S}\left(\frac{d_{\sigma j}}{d_{j\sigma}+x^*_{j\sigma}}\right)\sum_{\substack{h\in S\\ h\neq j}} t_{jh}x_{jh}\\
&= \frac{\mathbf{t}'\mathbf{x}^*_{\mathbf{d}}}{d_{\sigma\sigma}}
\end{aligned}$$

where the last equality holds because for each j, either $x^*_{\sigma j} = 0$, in which case

$$\frac{d_{\sigma j}}{d_{j\sigma}+x^*_{j\sigma}} = \frac{d_{\sigma j}+x^*_{\sigma j}}{d_{j\sigma}+x^*_{j\sigma}} = 1$$

by flow conservation (2.1), or $x^*_{\sigma j} > 0$, in which case $x_{jh} = 0$ for all h, by Lemma 3.1. In summary, the expected service time is

$$\mathrm{E}[\tau] = \frac{\mathbf{t}'(\mathbf{d}+\mathbf{x}^*_{\mathbf{d}})}{d_{\sigma\sigma}}. \qquad (3.11)$$

3.1.3 The Queueing Model

More formally, the model that we will consider here is an M/G/s queue, in Kendall notation (Adan and Resing, 2002). The ‘M’ (for Markovian) means that requests are received according to a Poisson process. The ‘G’ refers to a general service time distribution; in particular, we will use the distribution of the service time τ (3.5). The ‘s’ denotes the number of servers, which in our case is also the number of vehicles, n_K.

Let λ be the mean arrival rate, and let μ be the mean service rate (the reciprocal of the mean service time). The stability condition for the M/G/s queue with n_K servers is $\lambda/\mu < n_K$. When $\lambda = d_{\sigma\sigma}$ and $\mu = \tau^{-1}$, this condition is

$$\mathbf{t}'(\mathbf{d} + \mathbf{x}^*_{\mathbf{d}}) < n_K$$

by (3.11), which matches the capacity region condition (2.7).

No exact analytical results are known for the mean waiting time in an M/G/s queue when $s > 1$. When $s = 1$, the Pollaczek-Khinchine formula gives a closed-form expression for the mean waiting time, but no such closed form is known for the multi-server case (Adan and Resing, 2002). A large literature is available on approximations for the M/G/s queue; see Boxma et al. (1979) and, for a recent survey, Gupta et al. (2010). Closed form results are known for the M/M/s queue (Adan and Resing, 2002), in which the service time distribution is Markovian. However, Markovian service times with a sufficiently large mean tend to be over-dispersed relative to the distributions obtained from (3.8) for τ. There are efficient numerical schemes for the M/D/s queue (Tijms, 2006), in which service times are deterministic. In what follows, we will simply use simulations to estimate the performance measures.

3.1.4 Results

Figure 3.1 shows the mean passenger waiting time estimated using the M/G/s queueing model on the 4-station star network from Example 2.2 with varying fleet sizes. For all fleet sizes, waiting times are near zero when the intensity is near zero, and they diverge as the intensity approaches one. However, the fleet size significantly affects the shape of the curve in between; for larger fleet sizes, the queueing model predicts a flatter curve at low intensity and a steeper asymptote at intensity one. In practical terms, this means that the queueing model predicts that a PRT system with many vehicles can run at high utilisation and still provide low passenger waiting times. For example, at intensity 0.8 with five-minute one-way trip times, the mean waiting time with six vehicles is 150s, but it is near zero with sixty vehicles. Doubling the trip times increases these figures slightly, as shown in Figure 3.1b.

Figure 3.2 compares the mean waiting time predicted by the queueing model with that obtained by the SV algorithm that will be described later in Section 4.1; SV gave the lowest mean passenger waiting times so far achieved for this scenario. When the demand is from the hub to the spokes (Figure 3.2a), the two curves match closely, as expected, because the queueing model is exact when there is one origin. When the demand is from the spokes to the hub (Figure 3.2b), the queueing model predicts lower waiting times than those that are achieved by SV. However, the gap diminishes as the fleet size increases, so in this case low waiting times can indeed be obtained even at high utilisation, when the fleet size is large.

These results illustrate the main simplification made in the M/G/s model: it assumes that the assigned vehicle is idle at the request's origin station. In practice, there may be idle vehicles, but they may be located at the wrong station, and in this case the passenger incurs an extra wait that is not included in the queueing model. Proactive empty vehicle movement, using algorithms such as SV, can make this less likely to occur, particularly when the fleet size is large. These results also show that the benchmark provided by the M/G/s model is sometimes attainable. The next section develops another benchmark based on a different principle.

3.2 The Static EVR Problem

This section formally describes the static version of the EVR problem, in which all of the requests to be served are known in advance. Section 3.2.1 constructs a *vehicle-request graph* that will allow us to formulate the problem succinctly in Section 3.2.2. Related problems in the combinatorial optimisation and vehicle routing literature are discussed in Section 3.2.3; the main conclusion is that the static EVR problem is NP-hard. Section 3.2.4 derives a Mixed Integer Linear Programming (MILP) formulation that was used to solve small instances for validation purposes. Finally, Section 3.2.5 describes a simple constructive heuristic that provides good solutions for very large instances.

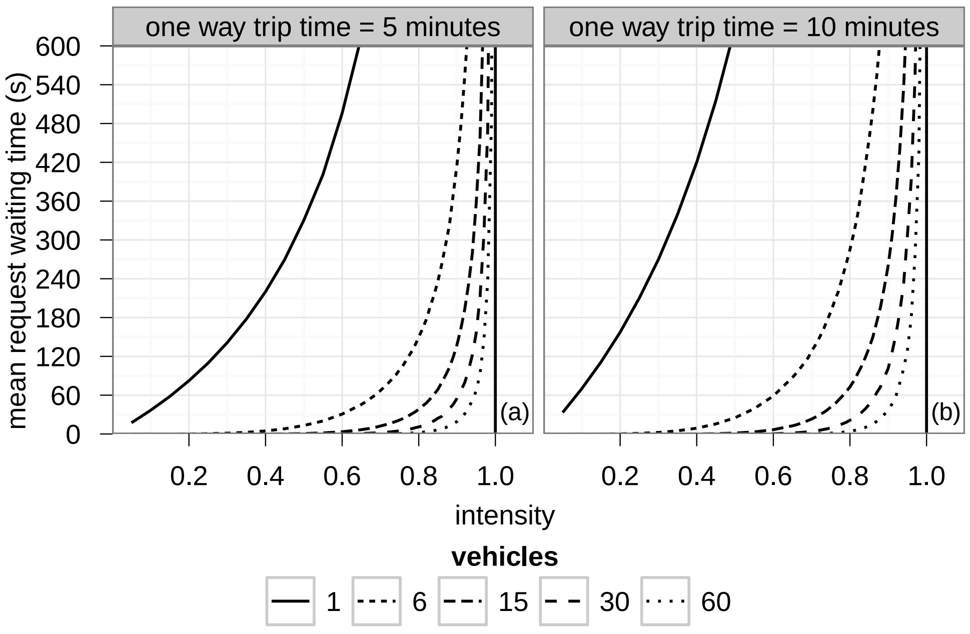


Figure 3.1: Mean waiting times for the M/G/s model on the 4-station star from Figure 2.2c on page 41 for varying fleet sizes, with five-minute (a) and ten-minute (b) one-way trip times. When the fleet size is large, the mean waiting time curve is flatter at low and moderate intensities, and it has a sharper asymptote at intensity one. These curves are computed using simulation; each point is the average of 5 runs of 0.5 million requests.

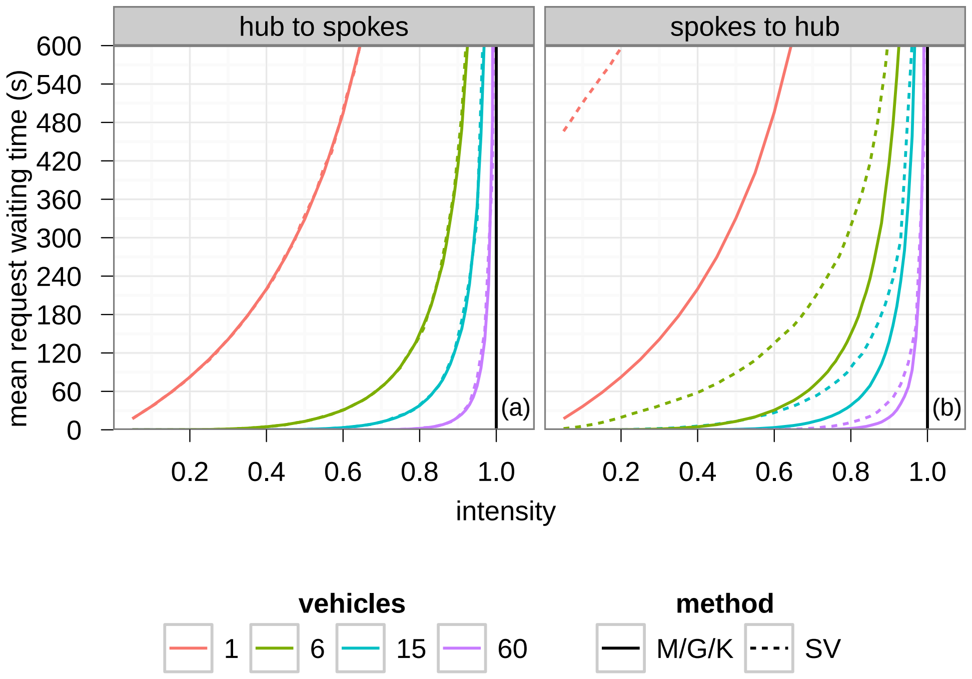


Figure 3.2: Comparison of mean waiting times on the four-station star for the M/G/s model and the SV algorithm. SV gives the lowest waiting times achieved in practice for these scenarios. It will be described in Section 4.1. Mean waiting times for SV match those for the M/G/s model when the demand is tidal from the hub to the spokes (that is, when there is a single origin). When the demand is tidal from the spokes to the hub, waiting times for the M/G/s model are lower than those obtained by SV, but the difference diminishes as the number of vehicles increases. One-way trip times are set to five minutes. The M/G/s curve is computed using simulation; each point is the average of five runs of 0.5 million requests. For the SV curve, $n_E = 30$, $n_R = 100$, and each point is the average of five runs of fifty thousand requests each.

3.2.1 The Vehicle Passenger Graph

In Chapter 2, the EVR problem was described in terms of vehicles moving between stations. In what follows, however, it will be more convenient to think about vehicles moving between requests. Let the vehicle-request graph, G, be a directed, weighted graph with node set $K \cup R \cup \{s\}$, where R is a set of request nodes, K is a set of vehicle nodes, and s is a special *vehicle sink node*. Each vehicle begins at its vehicle node, visits zero or more request nodes, and then terminates at the sink node. The sink node is just a technicality; it does not correspond to anything physical. The edge set is

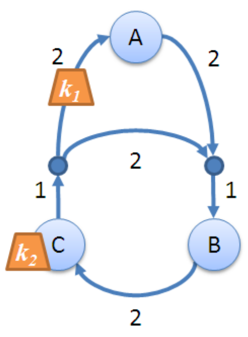
$$E = \{(u, v) : u \in K \cup R, v \in R \cup \{s\}, u \neq v\}$$

and the edge weights are delays, defined to encode the structure of the original PRT network, as follows. Following the notation from Section 1.2, let $t(i, j)$ be the quickest-path travel time from station i to station j, with the convention that $t(i, j) = 0$ if $i = j$. For each request $u \in R$, let i_u, j_u and e_u denote its origin station, destination station and time of receipt, respectively. We will assume that the first request is received at time 0. The vehicles may initially be idle at stations, or they may still be serving passengers that arrived in the past, before the set of requests being considered now. For each vehicle $u \in K$, define constants j_u and e_u so that vehicle u first becomes idle at station j_u at time e_u; if the vehicle is initially idle, then $e_u = 0$. The edge weights

$$w_{uv} = \begin{cases} e_u + t(j_u, i_v) & u \in K,\ v \in R \\ t(i_u, j_u) + t(j_u, i_v) & u, v \in R,\ u \neq v \\ 0 & u \in K \cup R,\ v = s \end{cases}$$

are defined to include both occupied travel time (e_u if $u \in K$ or $t(i_u, j_u)$ if $u \in R$) and empty travel time. Note that if $j_u = i_v$, then there is no empty vehicle trip required, and $t(j_u, i_v) = 0$. Also, since the $t(i, j)$ satisfy the triangle inequality (1.1), so do the edge weights. An example construction is given in Figure 3.3.

(a) The PRT network and initial vehicle positions.



(b) The requests to be served.

r_0	A → B at 0.0min (now)
r_1	C → B at 0.3min
r_2	C → A at 1.2min

(c) The resulting vehicle-request graph.

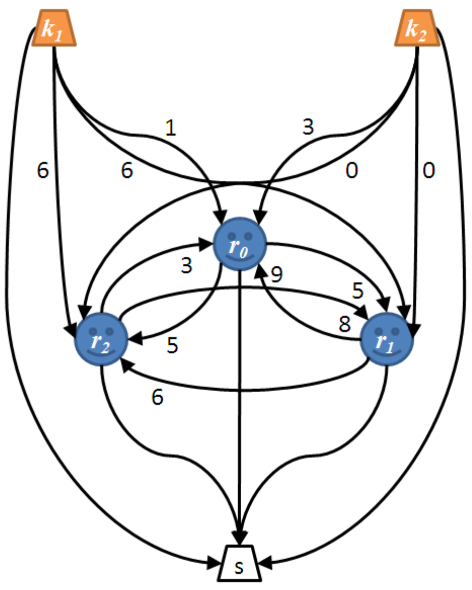


Figure 3.3: An example of a vehicle-request graph. The PRT network (a) includes stations, merges and diverges, but only quickest-path times are important for the vehicle-request graph. The edge weights are travel times, in minutes. There are two vehicles, k_1 and k_2; k_1 will arrive in station A in one minute, and k_2 is idle at station C. A complete list of requests (b) is provided, including times of receipt, origins and destinations; here there are three requests. The resulting vehicle-request graph (c) has a node for each vehicle and each request, and a sink node, s. Edge weights correspond to vehicle travel times. Suppose that, for example, vehicle k_1 serves requests r_0, r_1 and r_2, in that order. First, it finishes its current trip to station A, which takes 1 minute; then it arrives at A, which is r_0's origin. So, the edge (k_1, r_0) has weight 1. It then serves r_0 by moving from A to B, which takes 3 minutes, and moves empty for 2 minutes from r_0's destination, B, to r_1's origin, C; so, edge (r_0, r_1) has weight $2+3=5$. This repeats for r_2, and then the vehicle terminates at the sink node.

3.2.2 Arc Flow Formulation

For each edge $(u,v) \in E$, let x_{uv} be a binary variable; $x_{uv} = 1$ signifies that a vehicle traverses edge (u,v), and $x_{uv} = 0$ signifies that no vehicle traverses that edge. For example, if u and v are request nodes, then $x_{uv} = 1$ means that a vehicle serves request u and then proceeds to pick up request v. For each request u, let the variable t_u be the time at which a vehicle picks up that request; this is the time at which the vehicle departs from the request's origin station. It is also convenient to define constants $t_u = 0$ for $u \in K$. The Static EVR problem can then be defined as follows.

$$\min \sum_{u \in R} t_u \tag{3.12}$$

$$\text{s.t.} \quad \sum_{v:(u,v)\in E} x_{uv} = 1 \qquad \forall\, u \in K \cup R \tag{3.13}$$

$$\sum_{u:(u,v)\in E} x_{uv} = 1 \qquad \forall\, v \in R \tag{3.14}$$

$$\sum_{u:(u,v)\in E} x_{us} = |K| \tag{3.15}$$

$$x_{uv} = 1 \implies t_u + w_{uv} \le t_v \qquad \forall\, (u,v) \in E, v \ne s \tag{3.16}$$

$$e_u \le t_u \qquad \forall\, u \in R \tag{3.17}$$

$$t_u \le t_v \qquad \forall\, (u,v) \in E_{\text{FCFS}} \tag{3.18}$$

$$x_{uv} \in \{0,1\} \qquad \forall\, (u,v) \in E \tag{3.19}$$

Request u's waiting time is $t_u - e_u$, so the objective (3.12) is equivalent to minimising the total waiting time, since the e_u are constants, and hence the average waiting time, since all requests are served. Constraints (3.13), (3.14) and (3.15) ensure that each request is served by exactly one vehicle, and that each vehicle terminates at the sink node. (Constraint (3.15) is actually redundant.) The vehicle trips can be determined by starting from each vehicle node and following the arcs (u,v) for which $x_{uv} = 1$, through zero or requests, to the sink node. Constraints (3.16) ensure that pickup times are updated according to the vehicle flows. These are conditional constraints: if a vehicle serves request u and then request v (that is, if $x_{uv} = 1$), the vehicle must serve u and then move from his destination to v's origin before picking up v. Constraints (3.17) are *one-sided time window constraints* that ensure that request u is not picked up before it is received, at e_u. Constraints (3.18) are *precedence constraints* that ensure that requests with the same origin station are served in first-come-first-served order, with

$$E_{\text{FCFS}} = \{(u, v) : u \in R,\ v \in R,\ i_u = i_v,\ e_u < e_v\}\,.$$

3.2.3 Related Problems

The Static EVR problem is related to several well-known problems. It is helpful to begin with the Travelling Salesman Problem (TSP) and the Travelling Repairman Problem (TRP). In the TSP, a salesman wishes to visit his potential customers with the least possible time spent travelling between them; in the TRP, a repairman wishes instead to minimise the time that his customers spend waiting for service. In the Static EVR problem, the salesman (or repairman) is a vehicle, and the customers are requests. This section explores these connections in more detail; this gives insights into how the problem may be solved, and its complexity.

The Travelling Salesman Problem

The TSP, and many variations upon it, have been studied extensively; see Applegate et al. (1998) for a recent general survey. In the classical TSP, there is a single vehicle, all requests are received at time 0, and the travel times between requests are metric (they satisfy the triangle inequality) and symmetric ($w_{uv} = w_{vu}$). The vehicle's route must be a cycle that includes all of the requests; such a cycle is called a *tour*. The objective is to find a tour that minimises total vehicle travel time; this is equivalent to minimising empty vehicle travel time, because occupied vehicle travel times are fixed. Also, there are no time window or precedence constraints (3.17) or (3.18). When w_{uv} need not equal w_{vu}, the problem is known as an Asymmetric TSP (ATSP). Under these assumptions, and adopting the convention that $x_{uv} = 0$

and $w_{uv} = 0$ if $u = v$, the following is a formulation of the ATSP.

$$\min \sum_{u,v \in R} w_{uv} x_{uv} \tag{3.20}$$

$$\text{s.t.} \sum_{v \in R} x_{uv} = 1 \qquad \forall\, u \in R \tag{3.21}$$

$$\sum_{u \in R} x_{uv} = 1 \qquad \forall\, v \in R \tag{3.22}$$

$$\sum_{u \in C} \sum_{v \in C} x_{uv} \leq |C| - 1 \qquad \forall\, C \subset R, C \neq \emptyset \tag{3.23}$$

$$x_{uv} \in \{0, 1\} \qquad \forall\, u, v \in R \tag{3.24}$$

This is known as the Dantzig-Fulkerson-Johnson formulation (Dantzig et al., 1954), which has been used to obtain exact solutions to very large ATSP (and TSP) instances (Miller and Pekny, 1991; Applegate et al., 2003). The objective (3.20) and constraints (3.21) and (3.22) comprise an Assignment Problem (AP) of requests to requests; that is, each request is assigned to the next request to be served. The AP is a special case of the minimum cost network flow problem, and it can be solved efficiently using standard techniques (Bertsimas and Tsitsiklis, 1997), but the selected edges may form cycles that do not include all of the requests in R; these cycles are called *subtours*. Constraints (3.23) are *subtour elimination constraints* (SECs). They assert that any proper subset of n passengers must be connected by at most $n-1$ edges; adding another edge would create a subtour. There are exponentially many SECs in this formulation, so it is impractical to consider them all at once. Instead, constraints of this type are generated only when needed, using a *branch-and-cut* procedure. A recent survey and comparative analysis of ATSP formulations is given in Oncan et al. (2009).

Some formulations eliminate subtours using only polynomially many constraints but also extra variables. One of the oldest of these is the Miller-Tucker-Zemlin formulation (Miller et al., 1960), in which constraints (3.23) are replaced by

$$p_u - p_v + |R| x_{uv} \leq |R| - 1 \qquad \forall\, u, v \in R$$

where the continuous variables p_u are called *potentials*. These constraints eliminate subtours because the potentials increase along any path: when $x_{uv} = 1$, these constraints require that $p_v \geq p_u + 1$. This formulation requires $O(|R|)$ extra continuous variables and $O(|R|^2)$ constraints. The conditional constraints (3.16) on the pickup times in the Static EVR problem eliminate subtours using the same principle, because they force the passenger pickup times to increase along any path.

The TSP and the ATSP are both NP-hard, and much work has been done on approximation algorithms. Whereas constant-factor polynomial time approximations for the metric symmetric TSP are known, the best known approximations for the metric ATSP have only an $O(\log n)$ guarantee (Kaplan et al., 2005); in this sense, the ATSP is harder. The classical TSP requires that the vehicle return to its starting position at the end of its tour; if this requirement is relaxed, the problem is known as the Hamiltonian Path Problem, which is also NP-hard. In the asymmetric case, the problem is often called the Asymmetric Travelling Salesman Path Problem (ATSPP), which has an $O(\log n)$ approximation algorithm (Chekuri and Pál, 2006).

The Travelling Repairman Problem

The classical TRP is just like the TSP, except that the objective is to minimise the sum of the requests' pickup times, instead of the vehicle's travel time. In this sense, the TRP is customer-oriented, and the TSP is server-oriented. The TRP is also known as the Delivery Man Problem (Fischetti et al., 1993), the Travelling Deliveryman Problem (Mendez-Diaz et al., 2008), and the Minimum Latency Problem (Blum et al., 1994). Like the TSP, the MLP is NP-hard, and much work has been done on approximations (Archer et al., 2008). Exact approaches have been much less successful for the MLP than for the TSP; random instances with 30–40 customers are considered large (Mendez-Diaz et al., 2008; Sarubbi and Luna, 2007). Variants in which the costs are asymmetric and the server is not required to return to the start have also been studied (Friggstad et al., 2010).

Vehicle Routing Problems

Vehicle Routing Problems (VRPs) are variants of the TSP that often feature multiple vehicles, vehicle capacity constraints and time windows. The classical VRP features a fleet of vehicles (say, tanker trucks) that deliver a single commodity (gasoline) from a single depot (a refinery) to several customers (gas stations) (Dantzig and Ramser, 1959). When the vehicle has a fixed capacity (volume or weight limit), the problem is said to be *capacitated.* See Toth and Vigo (2002) and Golden et al. (2008) for recent surveys.

Pickup and Delivery Problems (PDPs) are VRPs in which the goods (or passengers) to be transported have particular origins and destinations. A typical example is an urban courier service, in which one or more vans pick up and deliver parcels. Each parcel must be picked up from its origin and delivered to its destination by the same van, and each van usually has a finite capacity. See Berbeglia et al. (2007, 2010) for recent surveys. A similar problem arises in paratransit, where the vehicle is a bus and the parcels are replaced by passengers; this is called the Dial-A-Ride Problem (DARP). An important feature of the DARP is that each vehicle can carry several passengers at once, each of whom may have a different destination. We have assumed that each vehicle serves only one request at a time; problems of this kind are sometimes called *truckload* problems (Powell, 1996).

It is often the case that requests can only be picked up within set *time windows.* The window is defined by an *early time* and a *late time*; if only one is specified, the time window said to be *one-sided.* The early time is usually interpreted as a hard constraint, but the late time may be a soft constraint; that is, the objective function may includes terms that penalise solutions in which requests are picked up after their late time. This is one way in which latency can be incorporated into the objective function of a VRP.

Summary

In TSP/VRP terms, the salient features of the Static EVR problem are as follows.

1. The travel times are *asymmetric* (w_{uv} may not equal w_{vu}) and *metric* (they satisfy the triangle inequality).

2. There are multiple vehicles, which must collectively serve all of the requests; this is common in vehicle routing problems, and there is also a literature on multi-vehicle TSPs (Bektas, 2006).

3. Each vehicle may have a different starting point. When there are multiple vehicles, it is often assumed that all vehicles begin at a single *depot*; the Static EVR problem can be seen as a *multi-depot* problem.

4. Vehicles do not have to return to their starting points at the end of their tour. That is, a path is required, rather than a cycle. This is a minor difference when travel times are asymmetric, because zero-time edges can be added from each request to any starting point.

5. Requests have one-sided time window constraints (3.17), because request u cannot be picked up before it is received, at e_u; if a vehicle reaches a request node u before e_u then the vehicle will wait for u to arrive.

6. Requests received at the same origin are subject to precedence constraints (3.18).

7. Latency is part of the objective function. Here we have defined the static EVR problem in terms of latency only, but terms for (empty) vehicle travel time could also be added.

The static EVR problem is NP-hard, because it generalises the minimum latency version of the ATSPP, which is NP-hard (Nagarajan and Ravi, 2008). In particular, the static EVR problem is a minimum latency ATSPP when the following three conditions hold.

1. There is one vehicle.

2. All requests are received at the start; that is, $e_u = 0$ for all requests, so the time window constraints (3.17) are not active. This is approximately true when the demand intensity is very high.

3. Every request is from a different station, so the precedence constraints (3.18) are not active. This is approximately true when the underlying PRT system has a large number of stations.

This means that there are some very hard instances of the static EVR problem, but it does not imply that all instances are hard. In particular, when the demand intensity is low and the number of stations is small, constraints (3.17) and (3.18) largely prescribe the order in which requests should be served. It may be possible to exploit these constraints to obtain provably optimal solutions for usefully large instances, particularly at low-to-moderate demand intensity. So far, only very small instances have been solved exactly using standard techniques (mixed integer linear programming), as described in the next section.

3.2.4 Exact Solution as a Mixed Integer LP

To solve (3.12) with standard mixed integer linear programming methods, the conditional constraints (3.16) must be linearised. The standard way to do this is the "big-M trick" (Williams, 1999): replace constraints (3.16) with

$$t_u + w_{uv} - t_v + M_{uv} x_{uv} \leq M_{uv} \qquad \forall\, u \in R, v \in R, u \neq v \quad (3.25)$$

where the constant M_{uv} is an upper bound on $t_u + w_{uv} - t_v$. When $x_{uv} = 1$, this constraint is equivalent to the original constraint; when $x_{uv} = 0$, the constraint is trivially satisfied, because M_{uv} was chosen to be an upper bound on the remaining terms. Here, $M_{uv} = t^{\max} - e_v$ where $t^{\max}$ is an upper bound on $t_u + w_{uv}$, namely

$$t^{\max} = \max_{u \in R} e_u + \max_{u \in K, v \in R} w_{uv} + \sum_{u \in R} \left(\max_{v:(u,v) \in E} w_{uv} \right). \quad (3.26)$$

This is obtained by assuming that one vehicle serves all of the passengers, that it waits until all passengers have arrived before moving, and that it chooses the worst link on each stage of its journey. When $u \in K$, the constraints can be linearised without big-Ms, as

$$w_{uv} x_{uv} \leq t_v \qquad \forall\, u \in K,\ v \in R, \tag{3.27}$$

because these are trivial constraints on t_v when $x_{uv} = 0$.

It is well-known that LP relaxations of MILPs with big-M constraints tend to give weak lower bounds, which requires that more nodes be examined in the branch-and-bound search. It is possible to reformulate the problem to avoid the big-M constraints by using a formulation for the Travelling Repairman Problem that was proposed by van Eijl (1995). It uses more continuous variables, but the big-M constraints are not required. For each request u, we replace the pickup time variable t_u with new variables τ_{uv} for each v with $(u, v) \in E$. It is also convenient to define constants $\tau_{uv} = 0$ for $u \in K$. Constraints (3.16) can then be replaced with

$$\sum_{u:(u,v)\in E} (\tau_{uv} + w_{uv} x_{uv}) \leq \sum_{w:(v,w)\in E} \tau_{vw} \qquad \forall\, v \in R \tag{3.28}$$

$$0 \leq \tau_{uv} \leq t^{\max} x_{uv} \qquad \forall\, u, v \in R \tag{3.29}$$

These constraints, together with the integrality constraints (3.19) on the x_{uv}, ensure that τ_{uv} is the pickup time at node u if $x_{uv} = 1$, and that $\tau_{uv} = 0$ if $x_{uv} = 0$. In other words, the new variables are related to the old variables by the identity $\tau_{uv} = t_u x_{uv}$. Constraints (3.17) and (3.18) and the objective (3.12) can then be rewritten in terms of the new variables using the identity

$$t_u = t_u \sum_{v:(u,v)\in E} x_{uv} = \sum_{v:(u,v)\in E} t_u x_{uv} = \sum_{v:(u,v)\in E} \tau_{uv}, \tag{3.30}$$

where the first equality follows from constraints (3.13).

It is worth noting that constraints (3.28) could also have been written without the summation on the left hand side, by instead creating a separate constraint for each u, like in (3.16). When the x_{uv} are restricted to $\{0, 1\}$, there is no difference, but when the x_{uv} are continuous, the form (3.28) with the summation is stronger.

A further refinement is to remove some arcs from the vehicle-request graph by using the precedence constraints. In particular, if requests u and v have the same origin station, and u arrives before v, a vehicle clearly cannot move from v to u. So, the arcs

$$\{(v, u) : u \in R,\ v \in R,\ i_u = i_v,\ e_u \leq e_v\} \tag{3.31}$$

can be removed, which is significant when the number of stations with outgoing demand is small. Note that removing these arcs does not make the FCFS constraints (3.18) redundant, unless there is only one vehicle. Time window constraints often rule out some routes in advance (Dumas et al., 1991, 1995), but with multiple vehicles and only one-sided time windows (3.17), this does not appear to be the case.

This MILP formulation was used to exactly solve very small instances (less than ten requests) with the open source GNU Linear Programming Kit (GLPK) solver (version 4.43) for validation purposes. However, tests with somewhat larger instances (forty requests) failed to terminate after several hours, even when the heuristic in the next section was used to give a reasonable initial solution. A commercial MILP solver from the Numerical Algorithms Group (NAG) was also used, but its performance was not significantly different from that of GLPK. Better results might have been obtained from the industry-standard iLOG CPLEX solver; this was not attempted. However, to obtain meaningful average waiting times for a given scenario, instances with thousands of requests would have to be solved. It may be possible to obtain exact solutions for such instances, but it appears that a more sophisticated approach is required. Here we will use the heuristic described in the next section, rather than attempting to obtain exact solutions.

3.2.5 Approximate Solution: Static Nearest Neighbours (SNN)

The following heuristic, which we call Static Nearest Neighbours (SNN), is a novel modification to the BWNN heuristic (Section 2.2.1) that takes advantage of known future requests. For each request $r \in R$, in ascending order by e_r, SNN chooses the vehicle

$$k^* = \underset{k}{\operatorname{argmin}} \left[\max(0, a_k + t(d_k, i_r) - e_r)\right] \tag{3.32}$$

that minimises the waiting time for request r. For vehicles with $a_k \geq e_r$, the objective values (2.15) for BWNN and (3.32) for SNN are the same. However, if a vehicle has $a_k < e_r$, SNN allows the vehicle to start its empty trip before e_r, whereas BWNN does not. In this sense, SNN moves empty vehicles retroactively, which can substantially reduce waiting times.

Ties in (3.32) are common, because there are often several vehicles that could reach i_r before the request at e_r, and all such vehicles would give zero waiting time. To break ties, we first try to select the vehicle with the minimum empty travel time $t(d_k, i_r)$. However, if (for example) there is more than one vehicle inbound to station i_r, they will all have zero empty time and so a further tie-breaking rule is required. To this end, we select the vehicle with latest arrival time $a_k + t(d_k, i_r)$ at i_r. The idea is that this choice allows better moves as one progresses through the list of requests, because the vehicles that arrive earlier at i_r have more flexibility to serve requests further down the list. Finally, if these measures are equal, we break ties by choosing the minimum such vehicle index k.

3.3 Discussion

In this chapter, we have developed two benchmarks for passenger waiting time: (i) the M/G/s queueing model, and (ii) the approximate solution of static EVR problems using SNN. Figure 3.4 compares these two benchmarks and the BWNN algorithm (Section 2.2.1) on the Corby and Grid scenarios. Both benchmarks predict near-zero mean waiting time for intensity less than roughly 0.8. Above intensity 0.8, the SNN benchmark predicts higher waiting times than the M/G/s benchmark. This may be due to the suboptimality of the SNN heuristic, or it may be that the mean waiting times predicted by the M/G/s queueing model are not achievable in practice; this is not currently

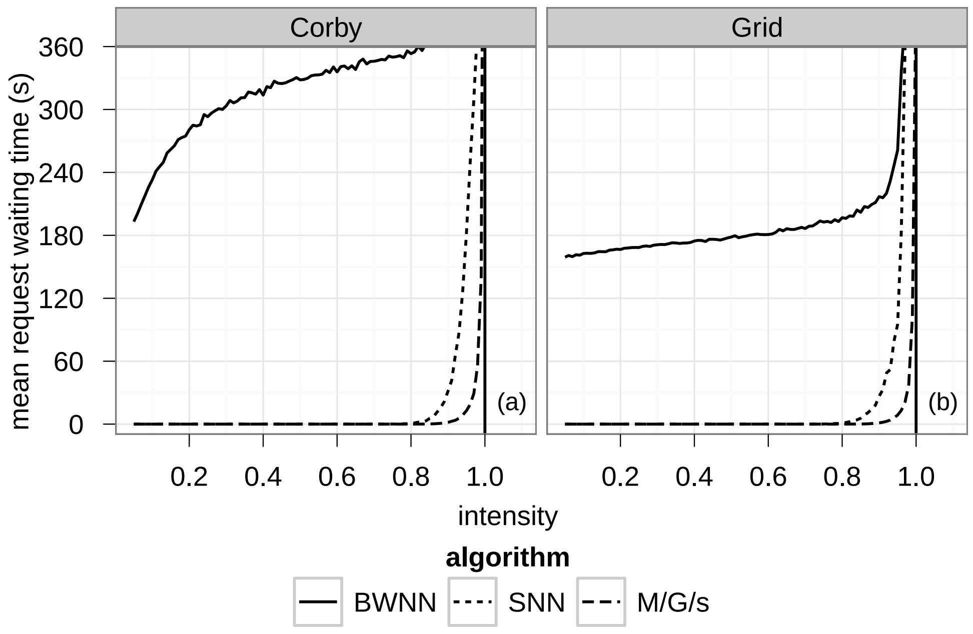


Figure 3.4: Baseline results for the Corby and Grid scenarios. The gap between the mean waiting times obtained by the reactive BWNN algorithm and the benchmarks indicates the potential for reducing mean waiting times by moving vehicles proactively. Each point on the M/G/s curve is the average of five runs of 0.5 million requests each. Each point on the SNN and BWNN curve is the average of five runs of fifty thousand requests each.

known.

Figure 3.4 also shows that there is a large gap between the mean waiting times obtained by the BWNN algorithm and the benchmarks, except at very high intensities. For example, the mean waiting time obtained by the BWNN algorithm on the Corby network at intensity 0.8 is roughly six minutes, but both benchmarks predict zero waiting time. This gap quantifies the possible reduction in waiting times that might be achieved by moving vehicles proactively, as the algorithms developed in the next chapter will do. For simplicity, we will compare the results only to the SNN benchmark, because no mean waiting times below this benchmark have been obtained in practice.

Chapter 4

New Proactive EVR Algorithms

This chapter introduces two new EVR algorithms, here called Sampling and Voting (SV) and Dynamic Transportation Problem (DTP), that move idle vehicles proactively, in anticipation of future requests. These algorithms are formulated as extensions to the BWNN algorithm (Section 2.2.1); BWNN is itself a reactive algorithm, meaning that it moves vehicles only in response to requests that have already been received. The main challenge for proactive EVR algorithms is that the details of individual future requests are not known exactly; only the statistics of future requests are known. In particular, here it is assumed that the mean request rates are known for each pair of stations (that is, that the demand matrix is known).

The SV algorithm works by directly generating an ensemble of possible sequences of future requests from the demand matrix over a given finite horizon. For each sequence, together with the current state of the system, the problem of finding a plan of vehicle movements that minimises total passenger waiting time is an instance of the static EVR problem (Section 3.2). In SV, each such problem instance is solved approximately using the SNN heuristic (Section 3.2.5), and each solution suggests a plan of empty vehicle movements. Features of these plans that are common across the ensemble suggest which empty vehicles should actually be moved. SV identifies these features using a voting system, in which each solution casts one 'vote' on which action to actually take.

The SV algorithm is motivated by similar approaches for dynamic vehicle routing problems (dynamic VRPs). The majority of vehicle routing literature is on static problems (Berbeglia et al., 2007), in which perfect information is available about all of the requests to be served. The usual context is that customers call in requests one or more days in advance, and the problem is to plan routes each evening for the following day; this means that all of the requests are known when planning. However, there are several recent studies (Cordeau et al., 2006; Berbeglia et al., 2010) on *dynamic and stochastic* problems, in which some fraction of the requests are for 'same day' service, and statistical distributions for these same day requests are available from historical data. Sampling approaches similar to that used here have been successfully applied to several such dynamic VRP variants related to the EVR problem (Yang et al., 2004; Bent and Van Hentenryck, 2004; Hvattum et al., 2006); these problems are partially dynamic (many requests are known in advance), or some advance notice is given for same day requests. Other heuristics for fully dynamic problems have also been proposed (Bertsimas and Levi, 1996; Swihart and Papastavrou, 1999). However, none of these problems is an exact match for the EVR problem.

Whereas the SV algorithm is based on ideas from the vehicle routing literature, the DTP algorithm builds on ideas from the PRT literature. It works by setting a *target* for the number of vehicles to be idle at, or inbound to, each station. Vehicles at stations with surpluses of vehicles (relative to their targets) can then be sent to stations with deficits. The problem of satisfying the targets with minimum empty vehicle movement is a classical transportation problem (Bertsimas and Tsitsiklis, 1997, ch. 7). The idea of maintaining a target number of vehicles at each station is common in the PRT literature (Andréasson, 1998; Anderson, 1998). The decisions on which vehicles to move are typically made according to rules that can be viewed as approximation algorithms for the transportation problem. For example, when a station has a deficit, it may call the nearest idle vehicle at a station with a deficit not greater than its own (Andréasson, 1998). The transportation problem and related minimum-cost network flow problems appear in many related contexts; for example, Andréasson (2003) formulates a different transportation problem that can be used to reroute PRT vehicles while they are moving, and Powell (1987) formulates a network flow problem for the dynamic allocation of trucks.

The principle of proactive empty vehicle movement also applies to conventional taxis. Most existing work on taxi dispatch focuses on reactive algorithms (Horn, 2002a,b; Bell and Wong, 2005; Seow et al., 2010) and related operational challenges such as travel time estimation (Lee et al., 2004) and the handling of both advanced bookings and immediate requests (Horn, 2002a; Wang et al., 2009). Approaches to proactive movement include random roaming (Lee et al., 2004), the go to hotspot heuristics of (Li, 2006), and the rank homing heuristics of (Horn, 2002a). A heuristic similar to that of Horn (2002a) is included in the simulation tests in Section 4.3. First, sections 4.1 and 4.2 describe the SV and DTP algorithms in more detail.

The SV algorithm was introduced in Lees-Miller and Wilson (2011). The DTP algorithm was introduced in Lees-Miller and Wilson (2012). The evaluation of SV and DTP in this chapter includes additional simulation results from a separate set of test scenarios (see Section 2.3); evaluation on a separate test set increases confidence in the obtained conclusions.

4.1 Sampling and Voting (SV)

The SV algorithm is defined as follows. When a new request is received, a vehicle is immediately assigned using BWNN (2.15). Immediately after a vehicle has been assigned to serve the request, SV may then move idle vehicles proactively. To decide which idle vehicles to move, an ensemble of n_E possible sequences of n_R future requests each is generated from the demand matrix. Each sequence in the ensemble, together with the current state of the system, defines an instance of the static EVR problem. Each of these instances is solved approximately using the SNN algorithm, and each resulting solution prescribes a sequence of empty vehicle trips, which constitutes 'advice' on which idle vehicles the system should actually move.

However, because each solution is for a different sequence of requests, they may offer conflicting advice.

To determine which action should actually be taken, a voting system is used. The system adopted here is that at most one idle vehicle at each station may be moved. So, each solution casts one vote on the best destination (as defined below) for an idle vehicle at each station i with idle vehicles; note that it may vote for i as the best destination, which means that it votes not to move any idle vehicles from i at this decision point. If the destination with the most votes is not i, an idle vehicle at i is selected and moved.

The destinations to vote for are determined as follows. The solution for each static EVR instance maps each input request to an empty vehicle trip, so each solution yields an ordered sequence of exactly n_R empty trips. Empty trip $p \in \{1, \ldots, n_R\}$ is described by the tuple (i_p, j_p, k_p), where $k_p \in K$ is the index of the vehicle used for trip p, and $i_p \in S$ and $j_p \in S$ are the stations at which the empty trip begins and ends, respectively. Trips with $i_p = j_p$ are trivial, in that no actual empty vehicle movement is required. For each station i with idle vehicles, let $K_i = \{k \in K : d_k = i \text{ and } a_k \leq t\}$ be the set of vehicles that are currently idle at i, and hence eligible to be moved at the current decision point. The following rules then determine the destination to vote for.

(i) vote for i if all vehicles in K_i were used for trips p with $j_p = i$, or

(ii) vote for j_p for the first trip p with $k_p \in K_i$ and $j_p \neq i$, if one exists, or

(iii) vote for j_p for the first trip p with $i_p = i$ and $j_p \neq i$, if one exists, or

(iv) vote for i.

Rule (i) votes to leave all idle vehicles at i if they were all needed there. If an idle vehicle at i was moved to another station, rule (ii) votes to perform this trip. If none of the idle vehicles were moved, as is common when demand is light, rule (iii) looks at all trips for a hint at where it should send one of these idle vehicles. If there were no trips from station i, rule (iv) leaves them where they are.

The concept of planning with an ensemble is very general, and many variants on the SV algorithm are possible, such as (a) using algorithms other than BWNN and SNN to assign vehicles to requests; (b) running the ensemble generation and voting system at different decision points; (c) using different voting systems; (d) using different rules to choose destinations from each sequence in the ensemble. The SV algorithm described here gave the lowest mean passenger waiting times among several such variants that were evaluated on the training scenarios.

4.2 Dynamic Transportation Problem (DTP)

The DTP algorithm is defined as follows. For each station i, let θ_i be the target number of inbound vehicles at station i; the θ_i are parameters that must be set, either manually or using an algorithm. At a given time t, let b_i be the number of vehicles that are inbound to i (that is, with $d_k = i$), and let l_i be the number of vehicles that are idle at i (that is, with $d_k = i$ and $a_k \leq t$); note that $b_i \geq l_i \geq 0$. Define $u_i = \min\{b_i - \theta_i, l_i\}$ as the *surplus* of vehicles at station i. If $u_i > 0$, station i has a surplus of inbound vehicles, but only l_i of these are currently idle, so at most l_i vehicles can be moved now. If $u_i < 0$, station i has a *deficit* of inbound vehicles. In general, the surpluses and deficits need not balance, so we introduce an extra dummy node q with $u_q = -\sum_i u_i$. Let $S' = S \cup \{q\}$ and partition S' into sets $S_t^+ = \{i \in S' : u_i \geq 0\}$ and $S_t^- = \{i \in S' : u_i < 0\}$. Note that this partition may change over time.

For each $i \in S_t^+$ and each $j \in S_t^-$, let x_{ij} be the number of vehicles to send from node i to node j, which is to be solved for. If i and j are both stations, then $x_{ij} > 0$ means that x_{ij} vehicles which are currently idle at i are to move to station j. If either $i = q$ or $j = q$, then no vehicles are actually moved, regardless of the value of x_{ij}; in other words, a decision to move idle vehicles from or to the dummy node means that they should be left where they are until the next decision point. The costs for sending vehicles to or from the dummy node are zero (define $t_{iq} = t_{qi} = 0$ for all $i \in S$), because these vehicles do not actually move. The transportation problem to be solved is then

$$\min \sum_{i \in S_t^+} \sum_{j \in S_t^-} t_{ij} x_{ij} \tag{4.1}$$

$$\text{s.t.} \sum_{j \in S_t^-} x_{ij} = u_i \qquad \forall\, i \in S_t^+ \tag{4.2}$$

$$\sum_{i \in S_t^+} x_{ij} = -u_j \qquad \forall\, j \in S_t^- \tag{4.3}$$

$$x_{ij} \geq 0 \qquad \forall\, i \in S_t^+,\ j \in S_t^- \tag{4.4}$$

Constraints (4.2) and (4.3) are flow conservation constraints; they ensure that the surpluses match the deficits. Constraints (4.4) prevent negative flows. When the targets θ_i are integers, there is an optimal solution in which all of the x_{ij} are integers, because the transportation problem is a special case of the minimum cost network flow problem and the u_i are integers (Bertsimas and Tsitsiklis, 1997). The problem (4.1) is solved exactly with the integer RELAX IV code (Bertsekas and Tseng, 1988) every time a request is received, and also every time a vehicle becomes idle.

The targets θ_i must be chosen to suit the network, demand matrix and fleet size. In what follows, we let $\boldsymbol{\theta}$ be the vector of targets θ_i for each $i \in S$, listed in order. The aim is to find a vector $\boldsymbol{\theta}$ that minimises the mean passenger waiting time. We use two metaheuristics for this purpose, namely simulated annealing and the cross-entropy method.

4.2.1 Setting Targets with Simulated Annealing

Simulated annealing is a well-known local search method (Laarhoven and Aarts, 1987). Here we use the implementation provided by the GNU Scientific Library (Galassi et al., 2010). The method is iterative. In each iteration, it replaces the current solution (target vector) with a randomly chosen *neighbour* solution, with an *acceptance probability* that depends on the resulting decrease (or increase) in the *energy* function (objective function). The acceptance probability also depends on a global 'temperature' parameter that decreases as the search proceeds, according to a *cooling schedule*. This terminology is motivated by connections with the physical process of annealing metals.

An initial estimate for the target vector is obtained from the fluid limit calculations in Chapter 2, namely

$$\hat{\theta}_i = \text{round}\left(\left(\sum_j t_{ji}(d_{ji} + x^*_{ji})\right)\left(\frac{\sum_j d_{ij}}{\sum_j (d_{ij} + x^*_{ij})}\right)\right) \tag{4.5}$$

where the x^*_{ij} are obtained from the optimal solution to (2.6), and the function round$(\cdot)$ denotes rounding to the nearest integer. The first factor is the number of vehicles that are expected to be inbound to station i on average, and the second factor is the fraction of vehicles that leave station i occupied. The rationale for the second factor is that if most of the vehicles leaving station i are empty, on average, then the station should not attempt to retain many idle vehicles.

Neighbouring solutions are generated by adding -1, 0 or 1 with equal probability to each target, and each target is constrained to remain in $[0, n_K]$. The energy function, $E(\boldsymbol{\theta})$, which is to be minimised, is the average passenger waiting time as estimated from a simulation with a given number of passengers. The cooling schedule and acceptance probabilities are set in the usual way for an 'inhomogeneous' simulated annealing scheme, as follows. The input parameters are the initial and final temperatures, t_0 and t_1, the Boltzmann constant, k, and the temperature decay factor μ_T. The temperature is reduced from T to T/μ_T in each iteration. The probability of moving from targets $\boldsymbol{\theta}$ to $\boldsymbol{\theta}'$ is 1 if $E(\boldsymbol{\theta}') < E(\boldsymbol{\theta})$ and

$$\exp\left(-\frac{E(\boldsymbol{\theta}') - E(\boldsymbol{\theta})}{kT}\right)$$

otherwise. The algorithm stops when $T < t_1$, and the returned target vector $\boldsymbol{\theta}^*$ is the one with the lowest energy out of all those evaluated in the course of the search (not necessarily the last one considered).

4.2.2 Setting Targets with the Cross-Entropy Method

The treatment here follows de Boer et al. (2005) and Alon et al. (2005). Let $\theta_{\max}$ be a positive integer that is an upper bound on the value of any component of the target vector, such as the fleet size. For each station $i \in S$, let p_{ix} be the probability that $\theta_i = x - 1$, for $x = 1, \ldots, \theta_{\max} + 1$, and let $\mathbf{P}$ be the matrix with entries p_{ix}. We require that the rows of $\mathbf{P}$ sum to one (that is, $\sum_x p_{ix} = 1$). The p_{ix} are all set to $(\theta_{\max} + 1)^{-1}$ initially.

In each iteration of the cross-entropy method, a fixed number, M, of target vectors $\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(M)}$ are drawn from the distribution defined by $\mathbf{P}$. The mean passenger waiting time score $E(\boldsymbol{\theta}^{(l)})$ for each vector $\boldsymbol{\theta}^{(l)}$ is then estimated by running a simulation of DTP with the corresponding targets. A fixed proportion, η, of the best target vectors are then used to update the probabilities for future iterations; in particular, let $\hat{\gamma}$ be the η-quantile of the scores in the sample (that is, if $\eta = 0.1$, at least 10% of the scores are less than $\hat{\gamma}$). Based on the sample, the best estimates of the probabilities that should be used for the next iteration are

$$\hat{p}_{ix} = \frac{\sum_{l=1}^{M} I(E(\boldsymbol{\theta}^{(l)}) \le \hat{\gamma}) I(\theta_i^{(l)} = x - 1)}{\sum_{l=1}^{M} I(E(\boldsymbol{\theta}^{(l)}) \le \hat{\gamma})}$$

where $I(\cdot)$ is an indicator function that is 1 if the given condition is satisfied and 0 otherwise. The update rule has a fairly simple interpretation: count how many target vectors gave low mean passenger waiting times relative to the other target vectors in the sample (as measured by η), and of those count how many target vectors had their ith component set to $x - 1$. Dividing the latter by the former gives the estimate. Because the $\hat{p}_{ix}$ are only estimates, it is common to update the probabilities for the next iteration using a smoothing factor α, according to

$$p_{ix} \leftarrow \alpha \hat{p}_{ix} + (1 - \alpha) p_{ix}.$$

The algorithm stops after a fixed number of iterations, or when the vector of most likely targets,

$$\boldsymbol{\theta}^* = \left(\left(\operatorname*{argmax}_x p_{ix}\right) - 1\right)_i ,$$

has remained the same for a given number of iterations. This vector of most likely targets is the output from the cross-entropy search.

4.3 Results

In this section, the proposed algorithms are evaluated in simulation. The simulations are conducted in the same way as those for BWNN, as described in Chapter 2.

For comparison, an existing EVR algorithm, here called the Surplus / Deficit (SD) algorithm, is also evaluated. It is an algorithm for the dynamic EVR problem that moves vehicles proactively. The general approach in SD is similar to several other published EVR algorithms (Anderson, 1998); it is most similar to that of Andréasson (1998). Several variants of SD were evaluated on the training scenarios; here we describe the one that gave the lowest mean waiting times. In the SD algorithm, each station i has an associated call time τ_i, which is the cumulative average of all previous empty vehicle trip times to that station. The surplus of vehicles at station i is the number of inbound vehicles that will arrive within the call time, minus the expected number of requests over the call time, namely $\tau_i \sum_j d_{ij}$. Like DTP, SD works by maintaining a target number of vehicles idle at, or inbound to each station; the difference is that the targets for SD are defined implicitly by the call times, rather than explicitly as parameters. When a new request is received, a vehicle is assigned using BWNN. Immediately afterward, SD may move idle vehicles proactively, as follows. For each station i with idle vehicles, in descending order by number of idle vehicles, if the surplus of vehicles at i is greater than or equal to one, an idle vehicle at i is sent to the nearest station with surplus less than zero (if any). Additionally, when a vehicle becomes idle at station i, the above actions are taken for station i only.

Figure 4.1a compares the mean waiting times observed for the five heuristics on the Corby system. An important observation is that waiting times increase rapidly as intensity approaches one, regardless of which EVR algorithm is used, as is expected based on the definition of intensity (2.8). In practice, we are most interested in the system's performance at intensities from around 0.7 to around 0.9, because in this range the system is well-utilised, but acceptably low passenger waiting times may still be obtained. At intensity 0.8, for example, mean waiting times are 355s for BWNN, 41s for SD, 20s for DTP, and 15s for SV. By moving vehicles proactively, SV reduces mean waiting times by 96% from the BWNN baseline. The relative reduction decreases as intensity increases, however, and in fact the SV, DTP and SD algorithms become increasingly similar to the BWNN algorithm at higher intensities, because there are fewer idle vehicles to redistribute. Figure 4.1c shows that the reduction in passenger waiting times comes from a modest increase in the average number of moving empty vehicles, or equivalently in empty vehicle travel time. The largest increase occurs at intensity 0.91, and this is from 47 concurrently moving empty vehicles with BWNN to 51 with SV (out of 200 vehicles). With perfect information about future arrivals, SNN finds routes with average waiting times less than those for the dynamic case, as expected; at intensity 0.8, the mean waiting time for SNN is 3s. Only mean waiting times are reported, but the ranking of the five algorithms is the same at the 90th percentile of the waiting time distribution; at intensity 0.8, 90% of passengers wait less than 51s with SV, 76s with DTP, and 106s with SD. Results for the Grid network are qualitatively similar, as shown in Figures 4.1b and 4.1d.

The parameters used to choose the targets for the DTP algorithm are given in Table 4.1 on page 98. The DTP results shown in Figure 4.1 are those with the minimum mean waiting time found by either the cross-entropy method or simulated annealing with the 'fast' cooling scheme defined in Table 4.1. The mean waiting time for each solution is estimated with a simulation of 20000 requests. Five independent runs of each search algorithm are performed for each DTP point in Figure 4.1.

Figure 4.2 compares the mean waiting times obtained by simulated annealing with the 'fast' cooling schedule to those obtained with a 'slow' schedule, as defined in Table 4.1. The parameters for the 'slow' schedule are chosen so that the simulated annealing search performs roughly the same number of simulations as the cross-entropy method, namely about 85 thousand simulations per run. The results are mixed; in the Grid scenario, the slow schedule produces lower mean waiting times than the fast schedule, but the opposite is true in the Corby scenario. Overall, the lowest mean waiting times (the solid line in Figure 4.2) were sometimes produced by simulated annealing but more often by the cross-entropy method. However, many other cooling schedules are possible, some of which are more sophisticated (Laarhoven and Aarts, 1987) than those used here, and these might produce better results.

Computation times for the BWNN, SNN and SD algorithms are negligible. Computation times for SV are larger, but SV is still fast enough for real time use with the case study networks: the mean computing time per passenger request for the SV results in Figure 4.1 is 0.04s with the hardware and SV implementation used for testing (user plus system time on a 2.0GHz Intel Xeon E5405 with g++ 4.1.2 and -O3). Execution times may of course vary depending on the hardware used and the implementation of the algorithm. The largest system so far tested with SV has 60 stations and 600 vehicles. When $n_E = 50$ sequences and $n_R = 750$ generated requests per sequence, SV produces lower waiting times than the SD and BWNN methods on an AM peak demand matrix for this system, and it uses 1.2s of computing time per passenger request. For this scenario, the demand at intensity one is 5050 requests/hour, or 0.7s per request, so the sequential SV algorithm used for testing is not fast enough for real time use at high intensity, in this case. However, SV is easy to parallelise, because each of the n_E sequences can be generated and processed independently; for example, real elapsed time per request could in this case be reduced to near 0.6s if two processors were used. The main limit on the scalability of the SV algorithm is thus the time to process a single sequence. The SNN minimisation (3.32) takes $O(n_K)$ time, so processing a single sequence takes $O(n_K n_R)$ time.

The computation times for DTP are small once the targets ($\boldsymbol{\theta}$) are set (4×10^{-5}s per request on the training scenarios). However, a potentially

Parameter \ Cooling Schedule	fast	slow
initial temperature (t_0)	10	20
temperature decay factor (μ)	1.01	1.001
final temperature (t_1)	0.1	0.004
moves per iteration [1]	10	10
Boltzmann constant (k)	1	1
simulations per run	4.6×10^3	8.5×10^4

(a) simulated annealing

Parameter \ Scenario	Corby	Grid
maximum target ($\theta_{\max}$)	50	50
sample size (M)	1500	2400
rarity parameter (η)	0.1	0.1
smoothing parameter (α)	0.5	0.5
maximum number of iterations	50	50
convergence test iterations [2]	5	5
mean simulations per run	8.4×10^4	

(b) cross-entropy method

Table 4.1: Parameters for the search for DTP targets with (a) simulated annealing and (b) the cross-entropy method on the training scenarios. The same simulated annealing parameters are used for both the Grid and Corby scenarios, but two different cooling schedules are used ('fast' and 'slow').

[1] Moves to this many neighbouring solutions are considered at each fixed temperature.
[2] The cross-entropy search terminates if the maximum likelihood solution does not change for this number of iterations.

large amount of off-line computation is required to find targets that give low waiting times. The off-line computation times for the DTP results in Figure 4.1 are around 100h per point (including both simulated annealing with the 'fast' cooling schedule and cross-entropy), which is impractically long. However, there are many possible improvements to the target search method used here. For example, rather than treating each demand matrix (that is, each point in Figure 4.1) separately, targets obtained for one demand matrix could be used as the initial targets for similar demand matrices. This is a topic for future work.

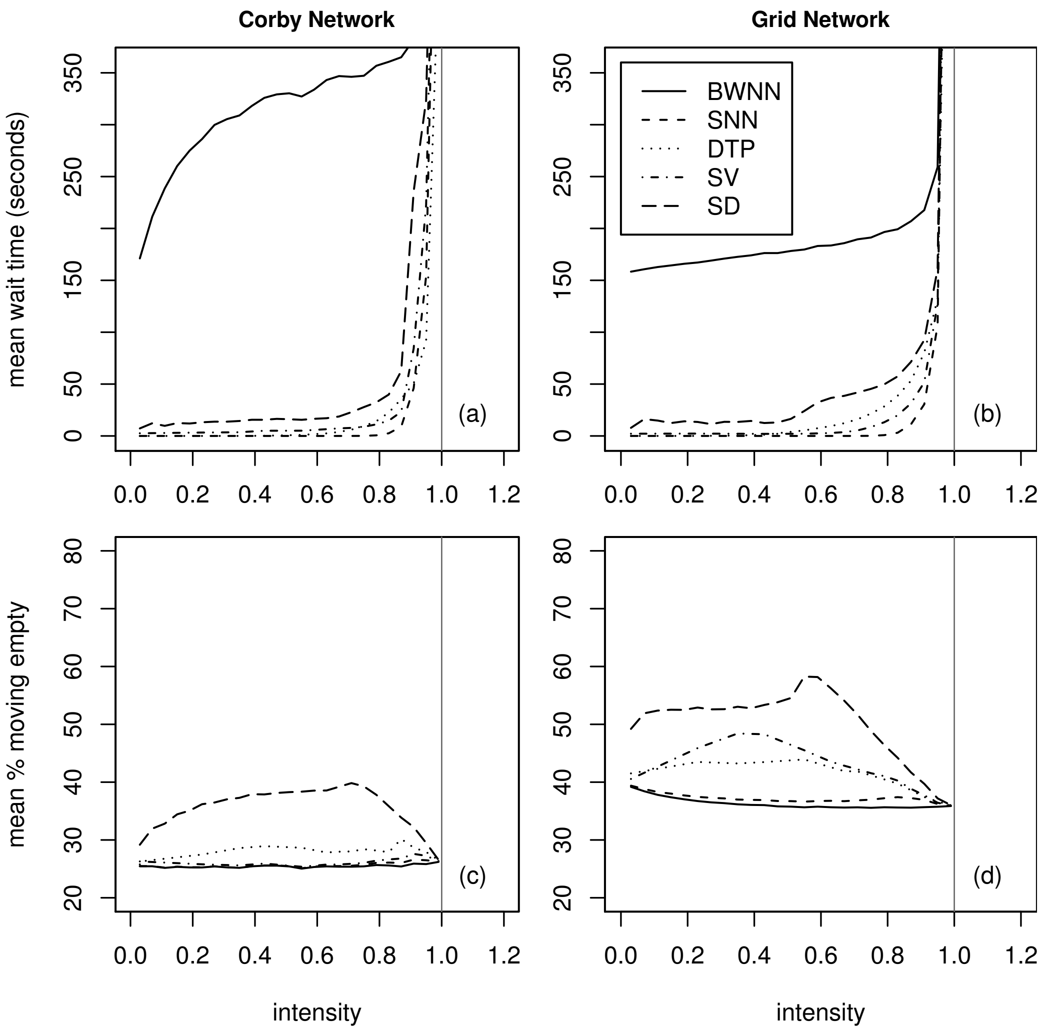


Figure 4.1: Mean passenger waiting times (a, b) and empty vehicle use (c, d) for the Grid and Corby scenarios for the five algorithms. The SV and DTP algorithms move idle vehicles in anticipation of future requests, which reduces waiting times significantly below the BWNN baseline; they also reduce waiting times below those obtained with the SD algorithm from the literature. The SNN algorithm operates with perfect information about future requests in order to estimate how much further waiting times might be reduced. Here there are $n_E = 50$ sequences, each with $n_R = 300$ requests for SV. The targets used for the DTP algorithm at each intensity are those that produced the lowest mean waiting time at that intensity, whether they were found by simulated annealing (with the 'fast' cooling schedule) or cross-entropy. Each point is averaged over 10 independent runs of 50000 simulated passengers each.

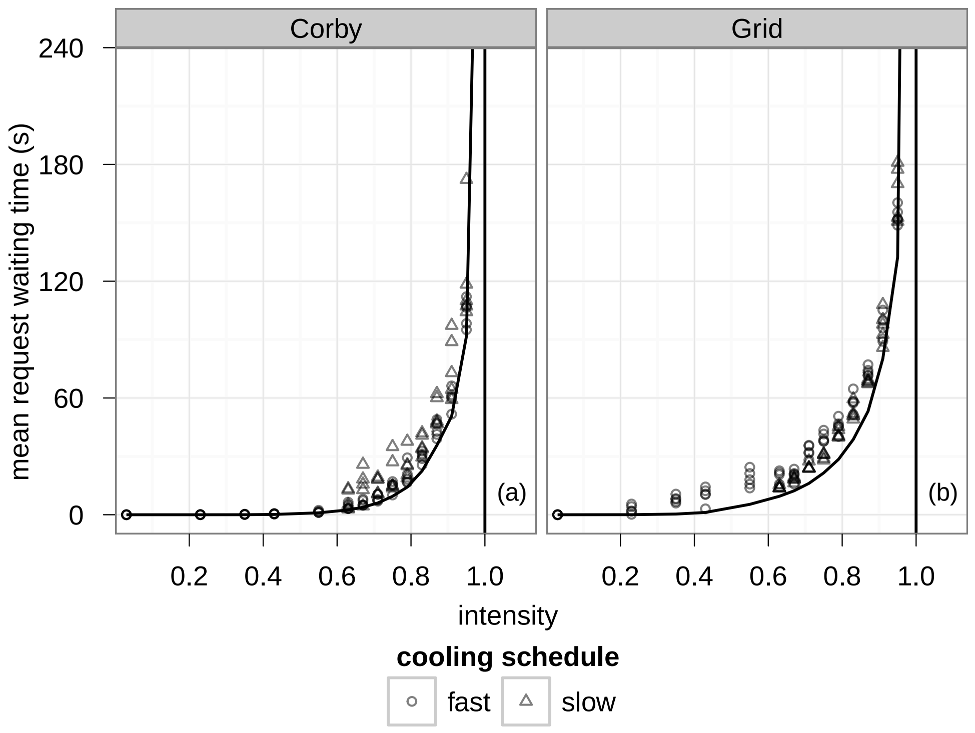


Figure 4.2: Mean waiting times for the DTP heuristic with targets obtained by simulated annealing (SA) on the Corby and Grid scenarios. Each point marks the lowest mean passenger waiting times obtained for one run of one of the SA cooling schedules defined in Table 4.1(a). The black lines mark the lowest mean passenger waiting times obtained by DTP using targets from any method (simulated annealing or cross entropy). The 'slow' cooling schedule was evaluated only for intensities above 0.6.

Scenario	n_E	n_R	$\theta_{\max}$	M
T1	50	1000	100	1500
T2	50	300	50	1800
T3	50	300	50	1800
T4	50	1000	50	1400
T5	50	300	50	1400
T6	50	300	50	1300

Table 4.2: Parameters for evaluation on the test scenarios. The n_E and n_R parameters are used in the SV algorithm. The $\theta_{\max}$ and M parameters are used in the cross-entropy search for DTP targets; also, the rarity parameter is 0.1; the probability update smoothing factor is 0.5; the maximum number of allowed iterations is 20 for intensities less than 0.5 and 40 for the rest; there are 10000 requests per evaluation.

4.3.1 Results on the Test Scenarios

The algorithms have also been evaluated on a separate set of test scenarios (see Section 2.3). Figures 4.3 and 4.4 show the mean passenger waiting times and empty vehicle use for the test scenarios. The parameters used are summarised in Table 4.2; the DTP targets are set using only the cross-entropy method. Again considering performance for intensities between 0.7 and 0.9, DTP gives equal or lower mean waiting times than SD in all six test scenarios, and SV gives equal or lower mean waiting times than SD in five out of six test scenarios; the exception is T1, for which SV produces lower waiting times only for very high intensities (above 0.85). Empty vehicle use is also reduced (or approximately equal) in five out of six scenarios for SV, and in all six scenarios for DTP. Overall, results from the test scenarios confirm those obtained on the training scenarios.

4.3.2 Effect of Sequence Length and Number on SV

The SV algorithm has two free parameters. Figure 4.5 shows the effects of varying the number of sequences n_E, and Figure 4.6 shows the effects of varying the number of requests, n_R, in each sequence. Figures 4.5a and 4.5b show that waiting times are not very sensitive to the number of sequences used. On the Corby system, a single sequence is sufficient to reduce waiting

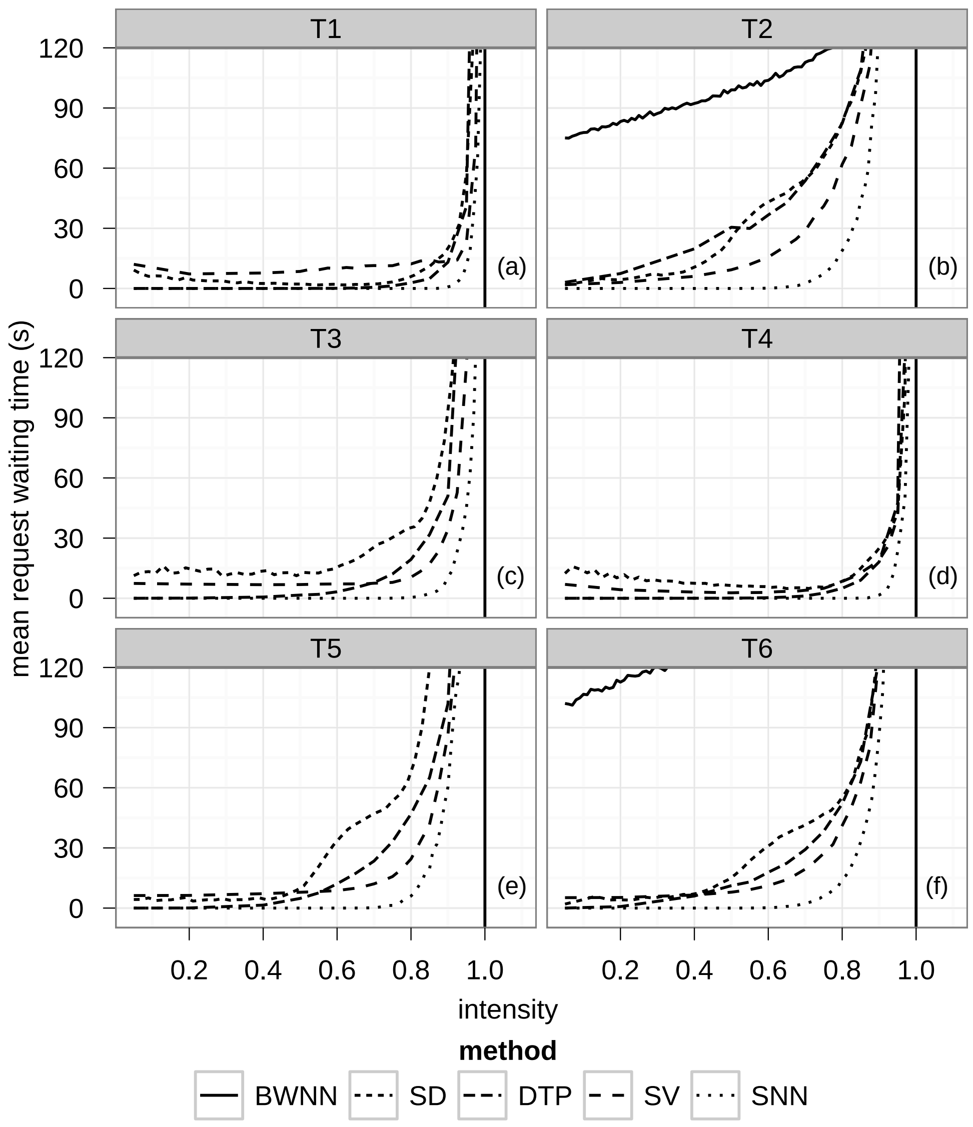


Figure 4.3: Mean passenger waiting times for proactive EVR algorithms on the test scenarios. Table 4.2 lists the parameters used. From intensity 0.7 to 0.9, which is the main region of interest, the SV algorithm gives lower mean waiting times than the SD algorithm from the literature in five out of the six test scenarios; DTP gives lower mean waiting times in all six scenarios.

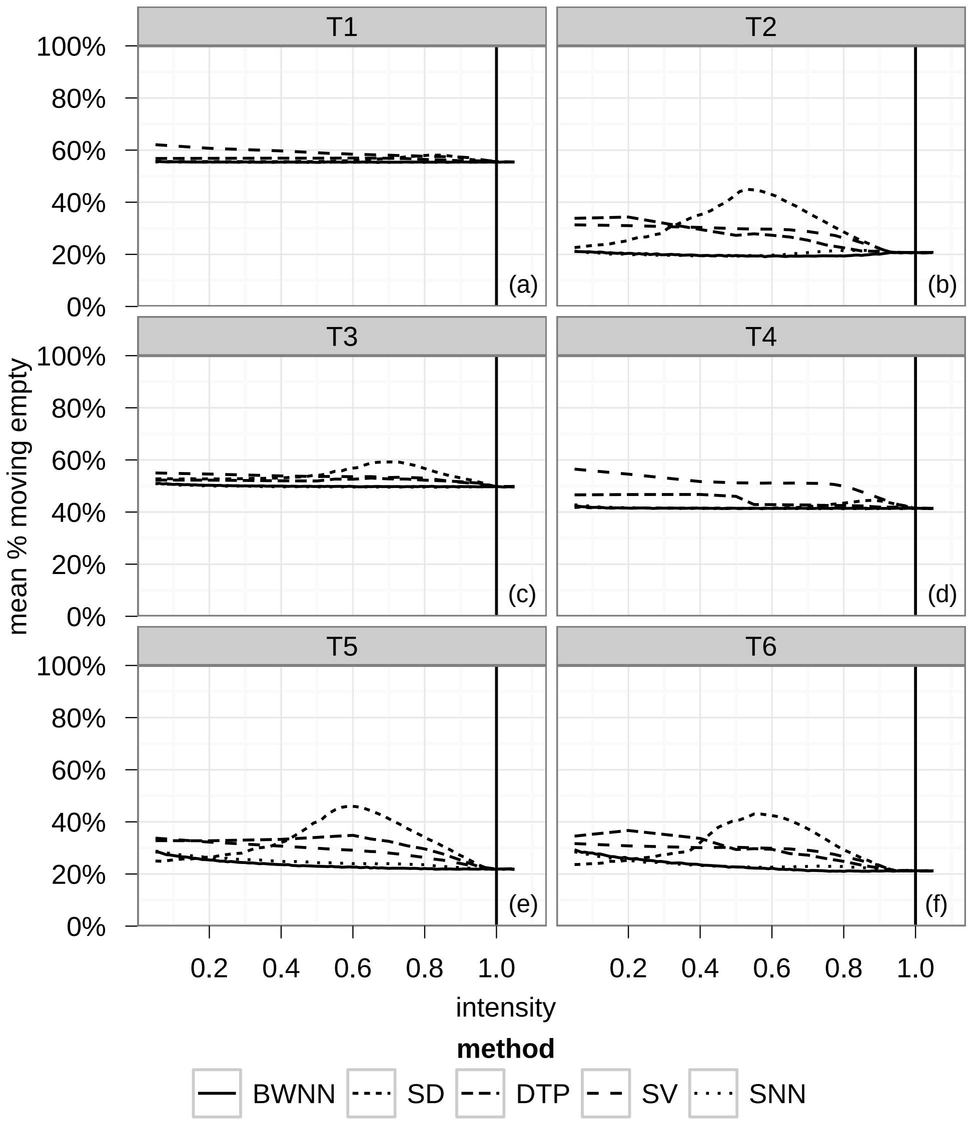


Figure 4.4: Empty vehicle use for proactive EVR algorithms on the test scenarios. Table 4.2 lists the parameters used. From intensity 0.7 to 0.9, the SV algorithm uses fewer empty vehicles than the SD heuristic from the literature for five out of the six test scenarios; the DTP algorithm uses fewer empty vehicles for all six test scenarios.

times well below the BWNN baseline. However, Figures 4.5c and 4.5d show that using more sequences produces some further reductions in waiting time and also significant reductions in empty vehicle travel. For example, on the Grid system at intensity 0.4 with a single sequence, 65% of moving vehicles are empty on average, but this falls to 45% when 50 sequences are used. This is because a small ensemble is less likely to be representative of the actual demand, and also because the actions recommended by a small ensemble are more likely to change at each decision point. So, while a small ensemble will sometimes make good recommendations that reduce passenger waiting times, it is also likely to make bad recommendations that result in wasted empty vehicle trips.

Figure 4.6 shows that waiting times decrease for all intensities as n_R increases. A sequence with more requests generates more occupied and empty vehicle trips, which makes it more likely that the sequence will vote on what to do with an idle vehicle, rather than just leaving it idle. This results in more empty vehicle movement, but also lower waiting times. For example, at intensity 0.8 on the Grid network, increasing n_R from 100 to 200 increases the fraction of moving vehicles that are empty from 36% to 40%, and it decreases mean passenger waiting times from 46s to 18s.

4.4 Discussion

The results show that the SV and DTP algorithms substantially reduce passenger waiting times by moving vehicles proactively. However, SV and DTP are heuristics, and the benchmarks developed in Chapter 3 do not rule out further improvements. This section describes several areas in which SV and DTP could be improved.

SV tends to give lower mean waiting times than the other heuristics at high intensities, but it sometimes performs poorly at low intensities. For example, SV provides the lowest mean waiting time for intensities larger than 0.55 in Figure 4.3e, but this is not the case at lower intensities. The main reason for this is that the SNN heuristic prefers vehicles with later arrival times (the second tie-breaking rule in Section 3.2.5), so it prefers to

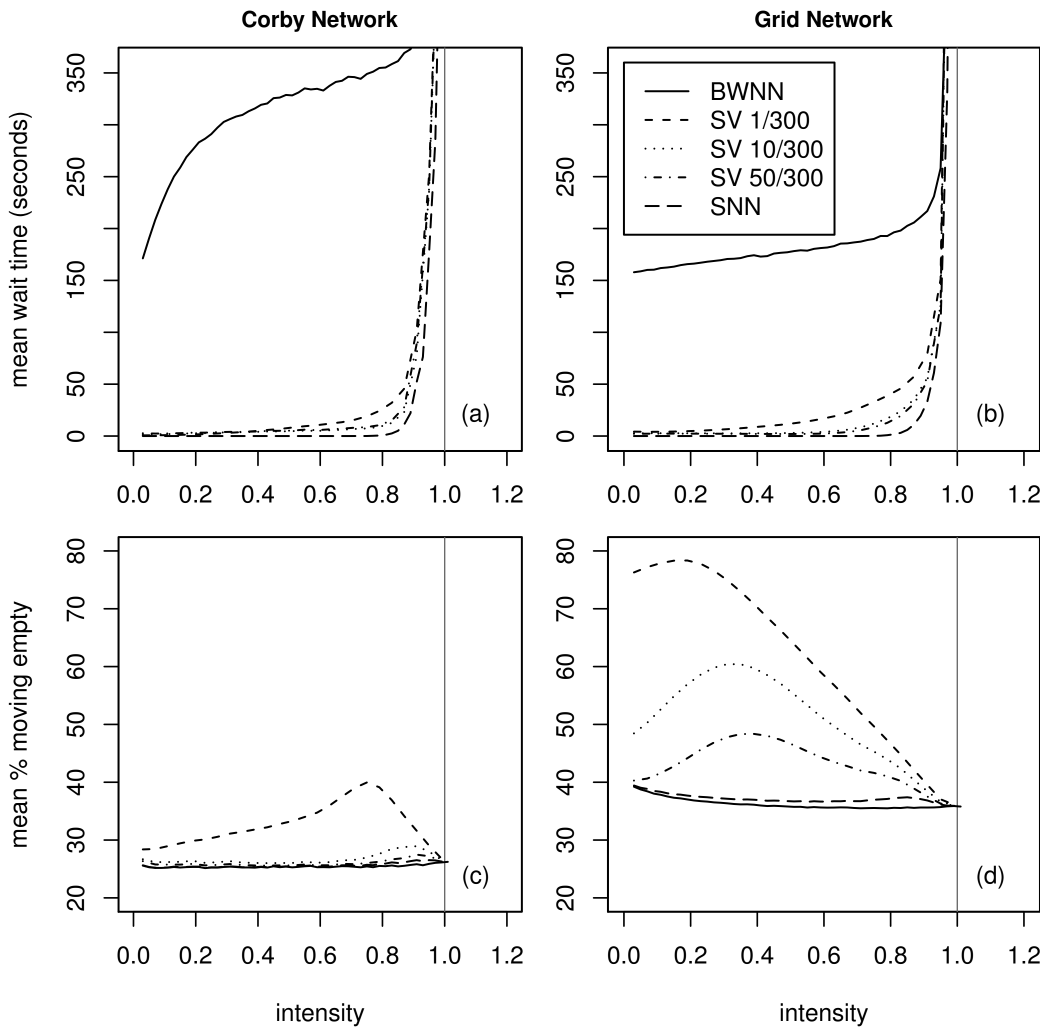


Figure 4.5: Effect of the number of sequences (n_E) on mean passenger waiting times (a, b) and the fraction of moving vehicles which are empty (c, d). Even a single sequence ($n_E = 1$) is sufficient to significantly reduce waiting times below the BWNN baseline, but using more sequences reduces the amount of empty vehicle travel required. Here $n_R = 300$ requests per sequence. Each point is averaged over 10 independent runs of 50000 simulated passengers each.

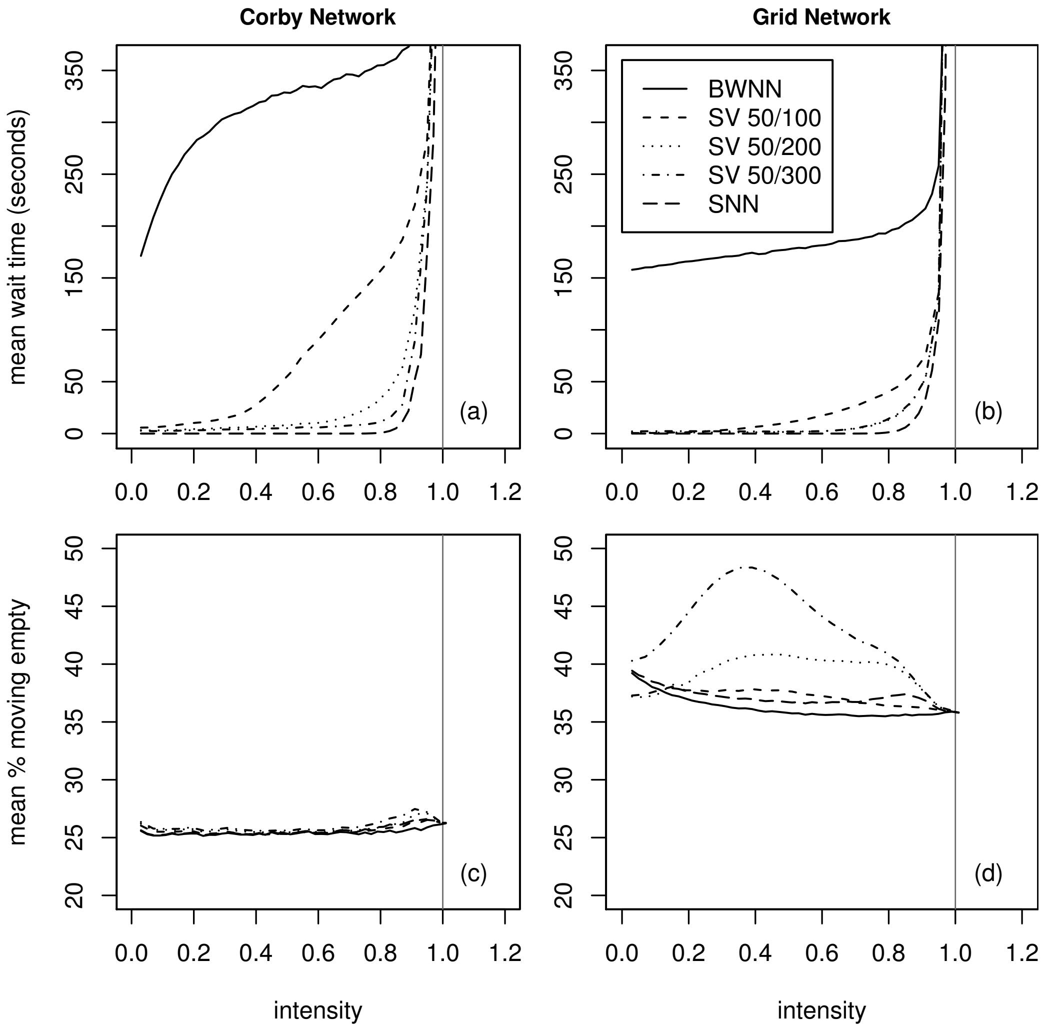


Figure 4.6: Effect of sequence length (n_R) on mean waiting times (a, b) and the fraction of moving vehicles which are empty (c, d). Longer sequences (with more requests) reduce passenger waiting times at the cost of modest increases in empty vehicle running. Here $n_E = 50$ sequences. Each point is averaged over 10 independent runs of 50000 simulated passengers each.

extend the routes of non-idle vehicles rather than moving idle vehicles. When the demand intensity is low, it can often use a small subset of the vehicles to serve all of the requests in a given sequence, so it tends to leave too many idle vehicles idle. The third voting rule in Section 4.1 mitigates this by counting empty trips with non-idle vehicles if some idle vehicles remain unused, but this is only a partial solution.

A more fundamental limitation of SV is that it uses a *scenario analysis* approach to optimisation under uncertainty. This term refers any method that generates and solves an ensemble of static (deterministic) problems (Kall and Wallace, 1997), as SV does. It is well-known that this approach is not sound in general; the reason is that the optimal action to take when there is uncertainty may not be optimal in any static version of the problem. This is because a static problem is solved with the benefit of hindsight, so it is never optimal to, for example, delay decisions until more information is available. The next chapter takes a different approach that is sound but so far intractable for practical problems.

A practical illustration of the above issue in the context of the EVR problem is that SV never uses empty vehicle *sidings*. An empty vehicle siding, or *car barn* (Anderson, 1998), is an off-line buffer where idle vehicles can be stored before they proceed to particular stations. For example, if there is a siding immediately upstream of several stations that might need an empty vehicle in the future, it could be best to send it to the siding first, and then send it to one of the stations later, when more information is available about requests at the stations themselves. SV never does this, because SNN never sends an idle vehicle to a siding; it only sends a vehicle directly to the origin station of a request, and there are no requests from sidings. (This is also true of any optimal solution to the static EVR problem, due to the strict triangle inequality (1.1) on the travel times.) The use of sidings may be increasingly important for larger networks, which have longer empty vehicle travel times.

It is notable that DTP may provide some support for using empty vehicle sidings. In particular, a siding could be modelled as a station with zero demand; this would allow DTP to have a non-zero target parameter (θ_i) for the siding, so some idle vehicles could be moved there proactively; the effectiveness of this approach has not yet been evaluated, however. This observation, and the fact that DTP tends to work better than SV at low demand intensities, suggests that it may be beneficial to combine DTP and SV into a hybrid heuristic; this is a topic for future research.

One potentially undesirable characteristic of DTP is that it does not prioritise larger deficits over smaller deficits. For example, it may occur that, at a particular decision point, there is a single surplus idle vehicle for the whole network, and several stations with deficits of vehicles. In this case, DTP moves the idle vehicle to the nearest station with any deficit, even if there is another station that is slightly further away but has a much larger deficit. A special case of this situation arises on the four-station star network from Example 2.2, in which all of the hub-to-spoke distances are the same, and DTP performs poorly in this case. There are several possible ways of mitigating this problem by adjusting the objective function of the transportation problem (4.1), but none of these have so far been evaluated.

Overall, the results show that SV and DTP are effective heuristics. They usually provide lower waiting times than other algorithms in the literature, albeit at the expense of more computation. In practice, the value of reductions in the passenger waiting times achievable with a given fleet size (or reductions in the fleet size needed to meet a given target for passenger waiting times) will likely offset the extra cost of these computations. The heuristics presented here may therefore be of considerable value to practitioners.

Chapter 5

Future Work and Conclusions

Section 5.1 presents preliminary work on a formal model of a PRT system (under the assumptions in Section 1.2) in a Markov Decision Process (MDP) framework (Puterman, 2005). This allows us to find empty vehicle redistribution strategies that are provably optimal for small systems. It may also allow us to apply standard techniques from the MDP literature to find good (but not provably optimal) strategies for larger systems. Here we will take some first steps in this direction by developing one such formal model and using it to find optimal strategies on small systems. Finally, Section 5.2 concludes the thesis.

5.1 A Markov Decision Process Formulation

MDPs are a formalism for modelling discrete time control problems in which the outcome is at least partially due to random chance. At each time step, the process is in a state s, and the decision maker takes an action a, that is valid in this state. The process then moves to a new state, s', for the next time step, with a given probability $\Pr(s, a, s')$. Each state has an associated reward, $R(s)$, which the decision maker receives immediately upon entering state s. The aim is to find a corresponding *policy* that the decision maker can follow to choose which action to take in each state; in its simplest form, a policy is just a table that lists the optimal action to take in each state. An *optimal* policy is one that maximises the (discounted) sum of the rewards that the decision maker receives over time.

Section 5.1.1 describes one way to define an MDP for a particular scenario — that is, for a given trip time matrix, demand matrix and fleet size. In particular, we will define the states, the actions that are valid in those states, the transition probabilities that relate states and actions, and the structure of the rewards. The rewards are defined so that maximum (discounted) total reward corresponds to minimum (discounted) total passenger waiting time. An optimal policy is thus an optimal empty vehicle redistribution strategy for the corresponding scenario.

A wide range of standard techniques are available to find good policies. For small MDPs, provably optimal policies can be obtained by dynamic programming (Puterman, 2005), as described in Section 5.1.2. Section 5.1.3 presents the optimal policies obtained for some small systems in this way. For larger systems, which generate larger MDPs on which dynamic programming is computationally infeasible, algorithms such as approximate dynamic programming (Bertsekas and Tsitsiklis, 1996; Powell, 2007), reinforcement learning (Sutton and Barto, 1999) and receding horizon control can produce high-quality policies. These approaches and other directions for future work are discussed in Section 5.1.4.

5.1.1 An MDP Model

There are many possible ways to model a PRT system as an MDP. The model used here is based on the delayed assignment model used in the LWPF algorithm in Section 2.2.2. Possible refinements will be discussed later.

We define the system state to be the number of queued passenger requests at each station, the assigned destination of each vehicle, and how long it will be before each vehicle reaches its assigned destination. A vehicle that has already arrived at its destination is said to be *idle*. The decision maker is a central dispatch system that can send idle vehicles on empty trips. The randomness in the system comes from the occupied vehicle trips, which occur in response to passenger requests. In each time step, the system receives zero or more passenger requests, each of which has an origin station and a destination station. If there is an idle vehicle at the origin, it serves the request by moving to the destination; otherwise, the request is added to the queue at the origin, where it waits to be served by a vehicle.

The decision maker's task is to move idle vehicles so as to minimise (discounted) total passenger waiting time. So, the reward associated with each state is chosen to be the negative sum of the queue lengths at all stations. If a passenger is left waiting for several time steps, the accumulated negative reward is the passenger's waiting time. It is also possible to penalise empty vehicle movements, but we will focus only on waiting times, for now.

Based on the notation from Section 1.2, let K be the fleet size, let S be number of stations, let t_{ij} be the trip time from station i to station j in whole time steps, and let d_{ij} be the demand from station i to station j in requests per time step. We will sometimes write t_{ij} and d_{ij} as $t(i,j)$ and $d(i,j)$, respectively, for readability, and we will define $t_{ij} = 0$ and $d_{ij} = 0$ when $i = j$. Let $d_{i\sigma}$ denote the total demand out of station i, $\sum_j d_{ij}$.

The system state at time t is written as a vector

$$s = (q_1, \ldots, q_S, d_1, \ldots, d_K, r_1, \ldots, r_K)$$

where q_i is the number of queued requests at station i ($q_i \geq 0$), d_k is the destination station of vehicle k ($1 \leq d_k \leq S$), and r_k is the number of time steps remaining until k reaches d_k ($0 \leq r_k \leq \max_{i,j} t_{ij}$). When $r_k = 0$, vehicle k is idle at d_k. For all valid states, the following also hold.

1. For each vehicle k, $r_k \leq \max_i t(i, d_k)$.
2. If there is no demand from a station, the queue is empty ($d_{i\sigma} = 0 \implies q_i = 0$).
3. If there are idle vehicles at a station then the queue is empty ($\{k : d_k = i \text{ and } r_k = 0\} \neq \emptyset \implies q_i = 0$).

We now turn to the description of all possible successor states reachable from state s. Let

$$s' = (q'_1, \ldots, q'_S, d'_1, \ldots, d'_K, r'_1, \ldots, r'_K)$$

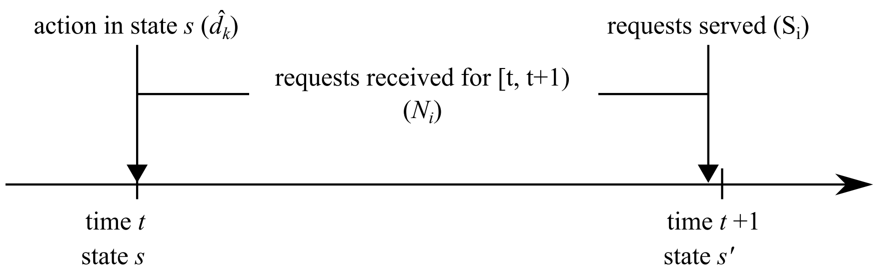


Figure 5.1: Time discretisation for the MDP model. If the action taken at time t is to move an idle vehicle from station i, it is not available for requests received at station i in $[t, t+1)$. All remaining idle vehicles at i, and vehicles that will become idle at time $t+1$, are available to serve requests.

denote a possible successor state; its components will be defined in terms of random variables, as described below; the transition process is also summarised in Figure 5.1.

1. The effects of the dispatcher's action are applied. An action for state s is written as a vector of new destinations

$$a = (\hat{d}_1, \ldots, \hat{d}_K)$$

that satisfies $\hat{d}_k = d_k$ if $r_k > 0$, because only idle vehicles can be sent on empty trips. If there are no idle vehicles in state s, then there is only one possible action, which is to allow vehicles to continue toward their assigned destinations; such a state represents a delay due to vehicle travel time.

2. New passenger requests are received at each station during the next time step. Let the random variable N_i denote the number of requests received with origin station i in the interval $[t, t+1)$. Here it is assumed that N_i follows a Poisson distribution with rate parameter $d_{i\sigma}$.

3. Queued or newly received requests at each station are assigned to the

available vehicles. Define

$$A_i = \left\{k : \hat{d}_k = i \text{ and } d_k = i \text{ and } r_k \leq 1\right\}$$

as the set of vehicles that are available to serve requests from station i in the current time step. The number of requests with origin i that can be served in the current time step is $S_i = \min\{q_i + N_i, |A_i|\}$, where $|A_i|$ is the number of vehicles in the set A_i. The number of queued (unserved) requests left over in s' is then $q'_i = q_i + N_i - S_i$. Each served request can have any destination (other than i). Let the random variable Δ_{il} be the destination station of the l^{th} request to be served $(1 \leq l \leq S_i)$; Δ_{il} takes value j with probability $d_{ij}/d_{i\sigma}$. Sort the vehicles in A_i into ascending order so that $k_1 < \cdots < k_{S_i} < \cdots < k_{|A_i|}$; for the first S_i of these, $k_1, \ldots, k_{S_i}$, set $d'_{k_l} = \Delta_{il}$. For all other vehicles, set $d'_k = \hat{d}_k$.

4. Finally, set

$$r'_k = \begin{cases} r_k - 1, & r_k > 1 \\ t(d_k, d'_k), & r_k \leq 1 \end{cases}$$

to move vehicles closer to their destinations.

In principle, each state in this MDP has an infinite number of possible successor states, because any number of requests could be received in the next time step (each N_i is unbounded above, because it follows a Poisson distribution). In practice, we make the state space finite by truncating each queue length q'_i to an arbitrary length, $q_{\max}$, beyond which additional requests are rejected.

The successor states s' are generated by enumerating all possible values of N_i that give queue lengths $q'_i \leq q_{\max}$, and, for each of these values, enumerating all possible permutations of the S_i destination variables Δ_{il}. The transition probability $\Pr(s, a, s')$ for each such s' is determined by the probability mass functions of the random variables N_i and Δ_{il}. Let n_i and δ_{il} be the particular values of N_i and Δ_{il} that yield the successor state s'. The random variables are all mutually independent, so

$$\Pr(s,a,s') = \prod_{i=1}^{S} f(n_i) \prod_{l=1}^{S_i} d(i,\delta_{il})/d_{i\sigma}$$

where

$$f(n_i) = \begin{cases} \Pr(N_i = n_i), & q_i' < q_{\max}, \\ \Pr(N_i \geq n_i), & q_i' = q_{\max}. \end{cases}$$

5.1.2 Solution Methods

The model described in section 5.1.1 is a finite MDP, and small instances can be solved exactly using standard techniques, such as value iteration (Puterman, 2005). The aim of value iteration is to assign every state s a *value*, $V(s)$, so that the Bellman optimality equations

$$V(s) = R(s) + \gamma \max_a \sum_{s'} \Pr(s,a,s')V(s') \tag{5.1}$$

are satisfied, where $\gamma \in (0,1)$ is a discount factor. That is, the value of state s is the immediate reward, $R(s)$, plus the discounted expected value of its successor states, assuming that we choose the action that maximises this quantity. Because the rewards are bounded (in $[-q_{\max}S, 0]$) and the state and action spaces are finite, equations (5.1) are guaranteed to have a unique solution, denoted $V^*(s)$ (Puterman, 2005). This solution is obtained by initialising $V(s)$ to $R(s)$ and then iterating until the value function converges to within a specified tolerance.

The optimal policy, $a^*(s)$, is obtained by being greedy with respect to the optimal value function; that is,

$$a^*(s) = \arg\max_a \sum_{s'} \Pr(s,a,s')V^*(s')$$

is the best action to take in state s.

5.1.3 Results

The smallest system of interest is a two station ring with a single vehicle. Let $S = 2$, $K = 1$,

$$T = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \qquad D = \begin{pmatrix} 0 & 0.2 \\ 0.3 & 0 \end{pmatrix} \tag{5.2}$$

and $q_{\max} = 1$. The states, rewards, optimal values and actions for the corresponding MDP are shown in Table 5.1. In states s_5–s_{12}, the vehicle is moving ($r_1 > 0$), so only one action is possible: the vehicle must continue on to its destination. States s_1–s_4 are more interesting, because there are two actions available for each state. For example, in s_1, the vehicle is idle at station 1 ($d_1 = 1$, $r_1 = 0$), and the allowed actions are to either keep it at station 1 ($\hat{d}_1 = 1$) or to move it to station 2 ($\hat{d}_1 = 2$). In this case, the optimal action in state s_1 is to keep the vehicle at station 1 ($a^*(s_1) = (1)$).

When the demand matrix D is changed, the optimal policy also changes. Figure 5.2 shows the optimal actions in states s_1–s_4 for demand

$$D = \begin{pmatrix} 0 & d_{12} \\ d_{21} & 0 \end{pmatrix} \tag{5.3}$$

as d_{12} and d_{21} each vary from 0 to 1 request per time step; this covers the system's capacity region, as derived in Example 2.1, and some of the space beyond it. When there are no queued requests ($q_1 = 0$ and $q_2 = 0$), the vehicle should (roughly speaking) either

1. go to station 1 (if not already there) and wait, when $d_{12} \gg d_{21}$,
2. go to station 2 (if not already there) and wait, when $d_{12} \ll d_{21}$, or
3. stay where it is, when d_{12} and d_{21} have similar magnitude or are both large.

When there is a queued request at station 1 ($q_1 > 0$ and $q_2 = 0$), for example, the vehicle should (roughly speaking) either

1. go to station 1 and serve the request, when d_{21} is small, or

	q_1	q_2	d_1	r_1	$R(s)$	$V^*(s)$	$a^*(s)$
s_1	0	0	1	0	0	-50.81	(1)
s_2	0	1	1	0	-1	-52.90	(2)
s_3	0	0	2	0	0	-50.57	(2)
s_4	1	0	2	0	-1	-52.88	(1)
s_5	0	0	1	1	0	-50.81	(1)
s_6	1	0	1	1	-1	-51.88	(1)
s_7	0	1	1	1	-1	-53.26	(1)
s_8	1	1	1	1	-2	-53.90	(1)
s_9	0	0	2	1	0	-50.57	(2)
s_{10}	1	0	2	1	-1	-53.17	(2)
s_{11}	0	1	2	1	-1	-52.10	(2)
s_{12}	1	1	2	1	-2	-53.88	(2)

Table 5.1: Solution to the problem (5.2) with discount factor $\gamma = 0.99$. The MDP has twelve states, s_1–s_{12}. The optimal values $V^*(s)$ and the optimal actions $a^*(s)$ are computed with value iteration. For example, in s_4, there are one or more passengers waiting at station 1 ($q_1 = 1$), the vehicle is idle at station 2 ($d_1 = 2$, $r_1 = 0$), and the optimal action is to move the vehicle to station 1 ($a^*(s_4) = (1)$). Note that there is more demand from station 2 to station 1 than in the other direction; this is why the value function for s_8 is more negative than that for s_{12}, for example.

2. wait at station 2 until a request from 2 to 1 is received, otherwise.

The particular shapes of the boundaries that separate these actions depend strongly on $q_{\max}$, especially when the demand is heavy. Figure 5.3 shows the optimal policies for $q_{\max} = 10$. The true value of $q_{\max}$ is probably larger (the queue length at which potential passengers give up and choose another mode), but the state space grows rapidly as $q_{\max}$, or the other parameters, increases, as shown in Table 5.2. When $q_{\max} = 10$, there are 264 states, as compared to 12, when $q_{\max} = 1$; as more stations and vehicles are added, or as $q_{\max}$ is increased, the number of states quickly reaches the millions.

Figure 5.4 shows the optimal policies when a second vehicle is added, which are roughly

1. send both vehicles to station 1 (go to 1), if they are not already there,
2. send both vehicles to station 2 (go to 2), if they are not already there,
3. leave both vehicles where they are (stay),
4. send the vehicles to opposite stations (balance), or
5. a mixture of 'balance' and either 'go to 1' or 'go to 2,' depending on the particular locations of the vehicles, as described in Table 5.3.

5.1.4 Discussion

The MDP as defined here could be refined in several ways, including the following.

1. Vehicles are interchangeable, but the current model does not exploit this: each vehicle has its own 'label'. The numbers of vehicles inbound to or idle at each station, and their remaining trip times are important, but knowing precisely which vehicles are where is not important. The state space and transition mechanism could be reformulated to exploit this.

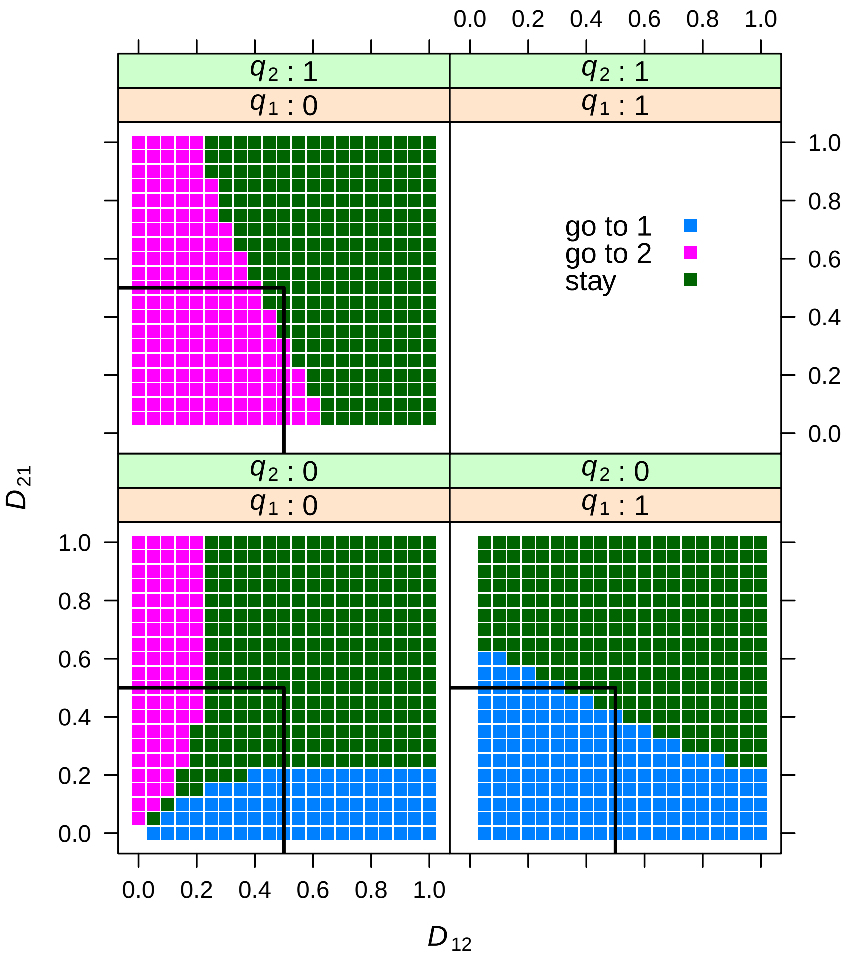


Figure 5.2: Optimal policy for a two station ring ($t_{12} = t_{21} = 1$) with one vehicle and $q_{\max} = 1$, as the demand matrix (5.3) is varied. For example, when $d_{12} = 0.8$, $d_{21} = 0.1$ and both q_1 and q_2 are zero (bottom left), the vehicle should go to station 1 (if it is not already there) and wait. If a request is then received at station 2, so $q_1 = 0$ and $q_2 = 1$ (top left), the vehicle should stay at 1, and leave the request at 2 waiting; this is because it anticipates that a request from 1 to 2 will soon be received, so it can serve the request at 1 on the way back. The policy when $q_1 = 1$ and $q_2 = 0$ (bottom right) is the mirror image of the policy when $q_1 = 0$ and $q_2 = 1$ (top left), because the network is symmetric. Note that when both $q_1 > 0$ and $q_2 > 0$ (empty space at top right), the system state is either s_8 or s_{12} (Table 5.1), so the vehicle is not idle, and there is only one valid action. The black rectangle marks the system's capacity region.

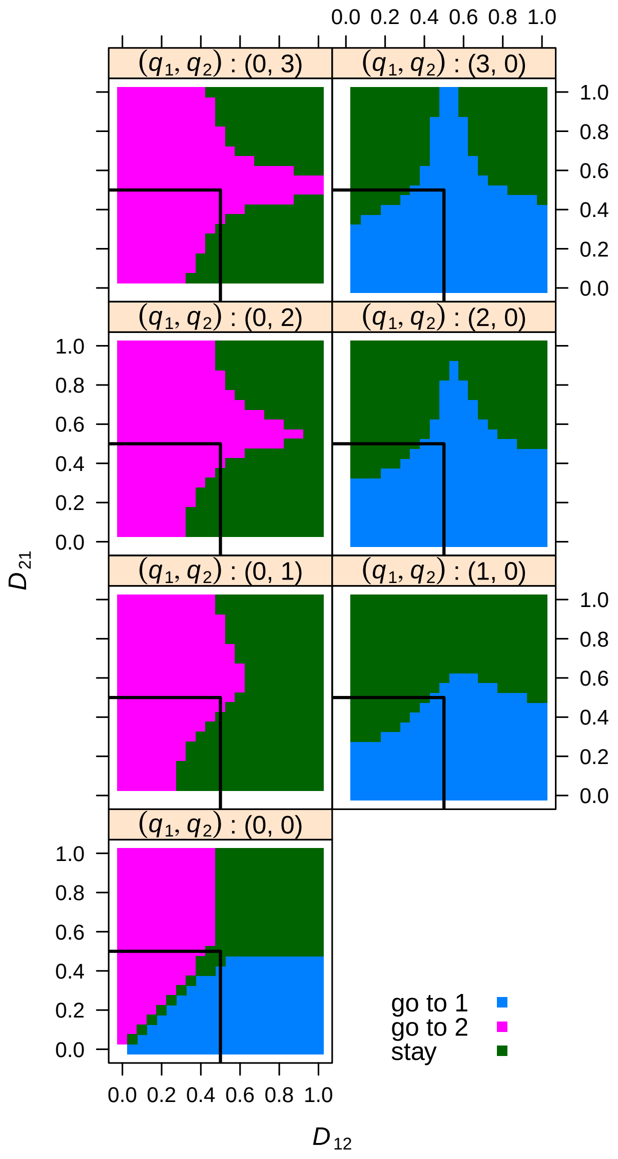


Figure 5.3: Optimal policy for a two station ring ($t_{12} = t_{21} = 1$) with one vehicle and $q_{\max} = 10$, as the demand matrix (5.3) is varied. Comparison with Figure 5.2 shows that $q_{\max}$ affects the optimal policy significantly when demand is high. It also contributes to the cusp in the boundary between optimal actions when there are queued requests. For example, when $q_1 = 0$, $q_2 = 1$ and $d_{12} = 0.6$, the vehicle should wait when $0.7 \leq d_{21} \leq 1.0$, because it is likely that the queue at station 2 will reach $q_{\max}$ and stay there, regardless of whether the vehicle serves the currently waiting request or not. The black rectangle marks the system's capacity region.

$K \setminus q_{\max}$	1	2	3	4	5	...	10
1	12	24	40	60	84		264
2	38	68	106	152	206		596
3	126	204	298	408	534		1,404
4	434	644	886	1,160	1,466		3,476
5	1,542	2,124	2,770	3,480	4,254		9,084
6	5,618	7,268	9,046	10,952	12,986		25,076
7	20,886	25,644	30,658	35,928	41,454		72,924
8	78,914	92,804	107,206	122,120	137,546		222,356

(a)

$K \setminus q_{\max}$	1	2	3	4	5
1	60	189	432	825	1,404
2	456	1,341	2,952	5,505	9,216
3	3,510	9,645	20,430	37,161	61,134
4	27,348	70,317	143,244	253,905	410,076
5	215,550	519,549	1,017,702	1,756,665	2,783,094

(b)

Table 5.2: State space size for (a) a two station ring with $t_{12} = t_{21} = 1$, and (b) a three station ring (b) with $T = \begin{pmatrix} 0 & 1 & 2 \\ 2 & 0 & 1 \\ 1 & 2 & 0 \end{pmatrix}$. Here, K is the fleet size and $q_{\max}$ is the artificial limit on the queue length at each station; it is assumed that $d_{ij} > 0$ if $i \neq j$.

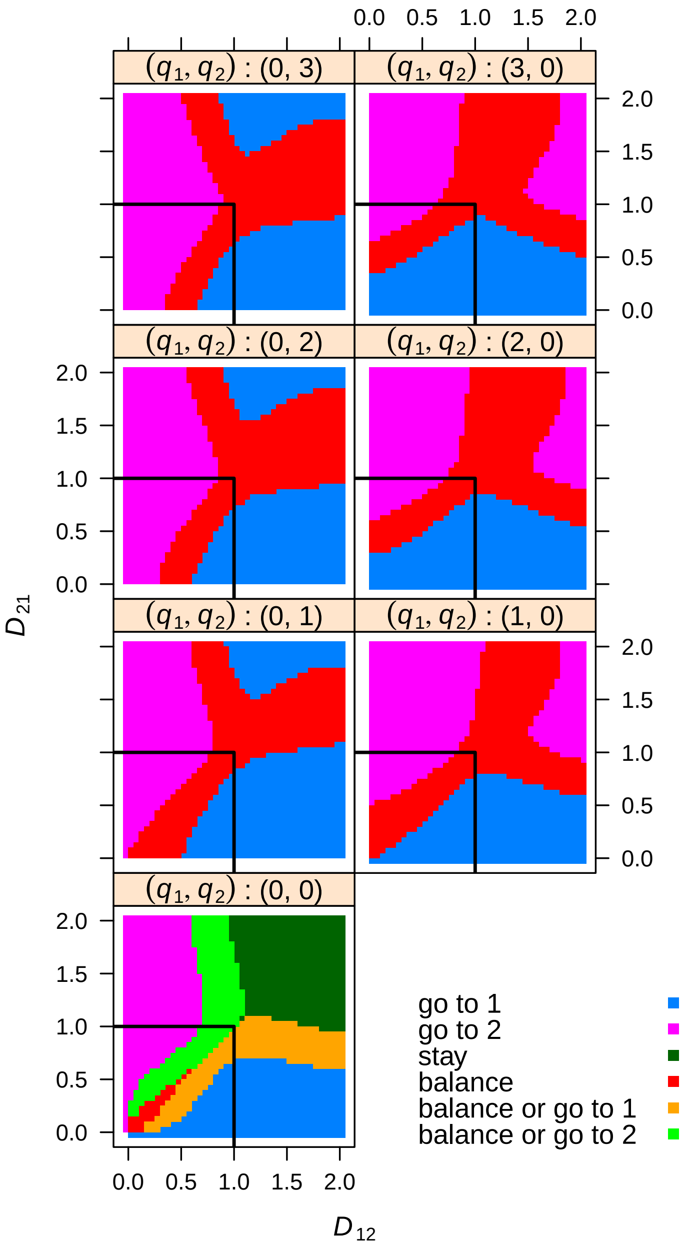


Figure 5.4: Optimal policy for a two station ring with two vehicles ($t_{12} = t_{21} = 1$, $K = 2$, $q_{\text{max}} = 10$) as the demand matrix (5.3) is varied. In this case 'go to 1' means that both vehicles should go to station 1, if they are not already there, and similarly for 'go to 2.' The 'balance or go to 1' and 'balance or go to 2' regions are described in more detail in Table 5.3. The black rectangle marks the system's capacity region.

	d_{21}	d_1	d_2	r_1	r_2	$a^*(s)$	Comments
1	0.15	1	1	0	0	$(1,1)$	leave both vehicles at station 1
2	0.15	2	2	0	0	$(1,1)$	move both vehicles to station 1
3	0.15	1	2	1	0	$(1,1)$	move both vehicles to station 1
4	0.20	1	1	0	0	$(1,1)$	leave both vehicles at station 1
5	0.20	2	2	0	0	$(1,2)$	move one vehicle to station 1, ...
6	0.20	1	2	1	0	$(1,1)$	... then move the other to station 1
7	0.25	1	1	0	0	$(1,1)$	leave both vehicles at station 1
8	0.25	2	2	0	0	$(1,2)$	move one vehicle to station 1, ...
9	0.25	1	2	1	0	$(1,2)$	... and leave the other at station 2

Table 5.3: Selected states and optimal actions near the boundary between the 'go to 1' and 'balance or go to 1' regions in Figure 5.4 for $d_{12} = 0.5$, $q_1 = 0$ and $q_2 = 0$. For $d_{21} \lesssim 0.15$, the optimal policy is always to send both vehicles to station 1 (go to 1). On the boundary, for $d_{21} \approx 0.2$, the optimal policy is to go to 1, except when both vehicles are idle at station 2. In this case, there is a delay between sending the first vehicle and, if no requests are received, sending the second vehicle. (The state on row 6 is the successor of the state on row 5, if no requests are received.) Finally, when $d_{21} \gtrsim 0.25$, the optimal action is to balance the vehicles between the two stations, except when both vehicles are idle at station 1. In this case, they should both be left idle at station 1. These last two cases are interesting, because they are examples of optimal policies that cannot be described by setting a target for the number of vehicles to be idle at or inbound to each station.

2. The current state includes only the number of queued requests at each station. This might be appropriate for a system in which passengers select their destination in the vehicle. If there are destination selection panels (see Section 1.1.2), then destinations will be known for at least some of the queued requests. The state space could be expanded to include the destinations of some or all waiting passengers (up to $q_{\max}$). This would permit a wider range of policies, and waiting times might be reduced.

The rapid growth of the state space with $q_{\max}$ limits the applicability of exact methods, because a large $q_{\max}$ is required in order to study the behaviour of the system at moderate-to-high demand intensities, which are the most interesting. A formulation that exploited vehicle interchangeability (point 1 above) might reduce the size of the state space and yield solutions for somewhat larger systems.

Beyond this, approximate dynamic programming (ADP) methods may be used (Bertsekas and Tsitsiklis, 1996; Powell, 2007). Whereas (exact) dynamic programming updates the value of every state in every iteration, ADP methods update the values of states along a single sample path at a time. The idea is to avoid spending time to refine the value estimates for states (or state-action pairs) that are very far from optimal. It is also noteworthy that ADP methods do not require a finite state space, and so do not require a $q_{\max}$ parameter.

The MDP formalism may permit the development of practical EVR algorithms with a more rigorous foundation than those in the Chapter 4. Whereas the approach taken so far has been to solve approximations to the actual problem, the MDP approach is to solve the actual problem approximately. The MDP approach has produced good results in related problem domains, such as the universal elevator controller in Wesselowski and Cassandras (2006), and it is a promising direction for future work on the EVR problem.

5.2 Conclusions

This thesis has developed two successful new EVR algorithms and several theoretical tools for evaluating them. Chapter 2 used a fluid limit to derive a benchmark for maximum attainable throughput. Using this, it was shown that BWNN, a simple nearest neighbour heuristic, achieves high throughput in practice. However, BWNN also caused long passenger waiting times when the demand was light, because it only moved idle vehicles reactively, in response to requests that had already been received. Chapter 3 then derived two benchmarks for waiting time, in order to quantify the reduction in waiting time that might be achieved by moving idle vehicles proactively, in anticipation of future requests. One of these benchmarks was based on combining the fluid limit analysis with a queueing model, and the other was based on approximately solving a static version of the EVR problem, in which the details of future requests were assumed to be known.

Chapter 4 described and evaluated two new EVR algorithms, SV and DTP, that moved idle vehicles proactively; the results showed that these algorithms substantially reduced passenger waiting times below those obtained by the reactive BWNN algorithm. SV and DTP also compared favourably with other proactive EVR algorithms in the literature, and they may be of significant value to practitioners. However, the benchmarks derived in Chapters 2 and 3 do not rule out further improvements over the methods developed here, particularly when the demand is heavy.

Finally, Section 5.1 presented early work on a Markov Decision Process (MDP) formulation of the EVR problem. Provably optimal policies were obtained for small systems, and this approach may also yield practical EVR algorithms in the future.

A common theme was the dependence of the results on how ‘heavy’ the demand was relative to what the system could theoretically serve. This was formalised using the demand intensity (2.8). At intensities near zero, there were many idle vehicles, and near-zero waiting times were obtained; as intensity approached one, however, waiting times diverged as the demand approached the system’s theoretical capacity. The EVR algorithm strongly affected the shape of the waiting time curve at intensities between zero and one. Proactive movement of empty vehicles flattened the curve at lower intensities and increased the steepness of the asymptote at intensity one. In other words, proactive EVR increased the range of demands for which the system could provide low waiting times. This effect was stronger for systems with larger fleet sizes.

The main focus was on passenger waiting times. In practice, there would be costs associated with empty vehicle operation. However, the fact that PRT vehicles would be guided by computers makes these costs considerably lower than those for conventional taxis. Proactive movement resulted in a modest increase in the amount of empty vehicle travel required, but it greatly reduced passenger waiting times.

Several topics were left for future research, including:

1. The effect of line congestion was ignored. Additional empty vehicle movement due to proactive EVR would contribute to line congestion; this could significantly increase trip times on systems that operate near their congested limit.

2. The demand matrix used to plan proactive movements was the same demand matrix that was used to generate the requests in simulation. In practice, there would be error in the estimated demand, which could make it difficult or impossible to move vehicles proactively.

3. The demand matrix was assumed to be stationary. In practice, the demand would be time-varying, and in general it could change rapidly due to public events, for example.

4. The demand intensity may often exceed one during peak periods; that is, the system could be required to operate outside of its capacity region for a limited time. A proactive EVR algorithm should ideally be able to prepare the system for these large transient demands.

5. The demand was assumed to be Poisson. A PRT station directly adjacent to a train station could experience a more bursty traffic pattern.

6. Vehicles were assumed to be interchangeable. This would not be the case if, for example, vehicles used batteries that sometimes ran out of charge.

Despite several simplifying assumptions, no model or solution for the EVR problem was obtained that was both accurate and tractable. The fluid limit model was tractable, but it did not prescribe a way of making decisions at the level of individual vehicles. The M/G/s queueing model was shown to be an accurate model of a PRT system with a single origin station, but its mean performance measures could only be obtained by simulation. The static EVR problem, in which the details of all requests were assumed to be known in advance, was shown to be NP-hard. An accurate MDP model was formulated, but the state and action spaces were found to be large enough to make exact solution infeasible. However, two new algorithms were developed and shown to perform well in practice, and several new and useful theoretical results were derived.

References

2getthere (2011a). Masdar operations. Accessed 11 May, 2011. Available from: http://www.2getthere.eu/?p=128.

2getthere (2011b). Personal rapid transit. Accessed 11 May, 2011. Available from: http://www.2getthere.eu/?page_id=58.

Adan, I. and Resing, J. (2002). Queueing theory. Course Notes, Eindhoven University of Technology. Available from: http://www.win.tue.nl/~iadan/queueing.pdf.

Alon, G., Kroese, D., Raviv, T., and Rubinstein, R. (2005). Application of the Cross-Entropy method to the buffer allocation problem in a Simulation-Based environment. *Annals of Operations Research*, 134(1):137–151. Available from: http://dx.doi.org/10.1007/s10479-005-5728-8.

Anderson, J. E. (1978). *Transit systems theory*. Lexington Books. Available from: http://www.worldcat.org/isbn/9780669019025.

Anderson, J. E. (1996). Some lessons from the history of personal rapid transit. Available from: http://faculty.washington.edu/jbs/itrans/history.htm.

Anderson, J. E. (1998). Control of personal rapid transit systems. *Journal of Advanced Transportation*, 32(1):57–74.

Anderson, J. E. (2003). Control of personal rapid transit systems. *Telektronikk*, 99(1):108–116.

Anderson, J. E. (2005). The SkyWeb express personal rapid transit system. *Urban Transport Xi*.

Anderson, J. E. (2007). Some history of PRT simulation programs. Accessed 10 May, 2011. Available from:

http://prtnz.com/publications-mainmenu-37/doc_details/24-some-history-of-prt-simulation-programs.

Andréasson, I. (1994). Vehicle distribution in large personal rapid transit systems. In *Transportation Research Record: Journal of the Transportation Research Board*, volume 1451, pages 95–99, Washington, D.C. Transportation Research Board of the National Academies.

Andréasson, I. (1998). Quasi-Optimum redistribution of empty PRT vehicles. In Sproule, W. J., Neumann, E. S., and Lynch, S. W., editors, *Automated People Movers VI: Creative Access for Major Activity Centers*, pages 541–550, Reston, VA. American Society of Civil Engineers.

Andréasson, I. (2003). Reallocation of empty personal rapid transit vehicles en route. *Transportation Research Record*, 1838:36–41.

Applegate, D., Bixby, R., Chvátal, V., and Cook, W. (2003). Implementing the Dantzig-Fulkerson-johnson algorithm for large traveling salesman problems. *Mathematical Programming*, 97(1):91–153. Available from: http://dx.doi.org/10.1007/s10107-003-0440-4.

Applegate, D., Bixby, R., Cook, W., and Chvátal, V. (1998). *On the solution of traveling salesman problems.* Rheinische Friedrich-Wilhelms-Universität Bonn. Available from: http://www.worldcat.org/oclc/535371278.

Archer, A., Levin, A., and Williamson, D. P. (2008). A faster, better approximation algorithm for the minimum latency problem. *SIAM Journal on Computing*, 37(5):1472–1498. Available from: http://link.aip.org/link/?SMJ/37/1472.

Becker, K. (1976). Cabintaxi: Technical level, market situation, and targets. In Gary, D. A., Gary, M. J., Kornhauser, A. L., and Garrard, W. L., editors, *Personal Rapid Transit III*. University of Minnesota, Audio Visual Library Service.

Bektas, T. (2006). The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega*, 34(3):209–219. Available from: http://dx.doi.org/10.1016/j.omega.2004.10.004.

Bell, M. G. H. and Wong, K. I. (2005). A rolling horizon approach to the optimal dispatching of taxis. In Mahmassani, H. S., editor, *Transportation and traffic theory : flow, dynamics and human interaction*

: proceedings of the 16th International Symposium on Transportation and Traffic Theory, pages 629–648, Amsterdam. Elsevier.

Bender, J. G. (1991). An overview of systems studies of automated highway systems. *Vehicular Technology, IEEE Transactions on*, 40(1):82–99. Available from: http://dx.doi.org/10.1109/25.69977.

Bent, R. W. and Van Hentenryck, P. (2004). Scenario-Based planning for partially dynamic vehicle routing with stochastic customers. *Operations Research*, 52(6):977–987. Available from: http://dx.doi.org/10.2307/30036646.

Berbeglia, G., Cordeau, J.-F., Gribkovskaia, I., and Laporte, G. (2007). Static pickup and delivery problems: a classification scheme and survey. *TOP*, 15(1):1–31. Available from: http://dx.doi.org/10.1007/s11750-007-0009-0.

Berbeglia, G., Cordeau, J.-F., and Laporte, G. (2010). Dynamic pickup and delivery problems. *European Journal of Operational Research*, 202(1):8–15. Available from: http://dx.doi.org/10.1016/j.ejor.2009.04.024.

Bertsekas, D. P. and Tseng, P. (1988). Relaxation methods for minimum cost ordinary and generalized network flow problems. *Operations Research*, 36(1):93–114. Available from: http://dx.doi.org/10.2307/171381.

Bertsekas, D. P. and Tsitsiklis, J. N. (1996). *Neuro-dynamic programming*. Athena Scientific. Available from: http://www.worldcat.org/isbn/9781886529106.

Bertsimas, D. and Tsitsiklis, J. (1997). *Introduction to Linear Optimization*. Athena Scientific. Available from: http://portal.acm.org/citation.cfm?id=548834.

Bertsimas, D. J. and Levi, D. S. (1996). A new generation of vehicle routing research: Robust algorithms, addressing uncertainty. *Operations Research*, 44(2):286–304. Available from: http://dx.doi.org/10.2307/171796.

Blum, A., Chalasani, P., Coppersmith, D., Pulleyblank, B., Raghavan, P., and Sudan, M. (1994). The minimum latency problem. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, STOC

'94, pages 163–171, New York, NY, USA. ACM. Available from: http://dx.doi.org/10.1145/195058.195125.

Bly, P. H. and Teychenne, P. (2005). Three financial and Socio-Economic assessments of a personal rapid transit system. In *Proceedings of the Tenth International Conference on Automated People Movers*, pages 39+. American Society of Civil Engineers. Available from: http://link.aip.org/link/?ASC/174/39.

Boxma, O. J., Cohen, J. W., and Huffels, N. (1979). Approximations of the mean waiting time in an M/G/s queueing system. *Operations Research*, 27(6). Available from: http://dx.doi.org/10.2307/172087.

Caudill, R. J., Kornhauser, A. L., and Wroble, J. R. (1979). Hierarchical vehicle management concept for automated guideway transportation systems. *Vehicular Technology, IEEE Transactions on*, 28(1):11–21. Available from: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1622606.

Chakroborty, P. and Das, A. (2003). *Principles of transportation engineering*. Prentice Hall of India. Available from: http://www.worldcat.org/isbn/9788120320840.

Chekuri, C. and Pál, M. (2006). An o(logn) approximation ratio for the asymmetric traveling salesman path problem. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 95–103. Springer. Available from: http://dx.doi.org/10.1007/11830924_11.

Cordeau, J.-F., Laporte, G., Potvin, J.-Y., and Savelsbergh, M. W. P. (2006). Transportation on demand. In Barnhart, C. and Laporte, G., editors, *Transportation*, Handbooks in operations research and management science, v. 14, chapter 7. North Holland.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2001). *Introduction to Algorithms*. MIT Press, second edition. Available from: http://www.worldcat.org/isbn/9780262032933.

Cottrell, W. D. (2005). Critical review of the personal rapid transit literature. In *Proceedings of the 10th International Conference on Automated People Movers*, volume 174, pages 40+. ASCE. Available from: http://dx.doi.org/10.1061/40766(174)40.

Dantzig, G., Fulkerson, R., and Johnson, S. (1954). Solution of a Large-Scale Traveling-Salesman problem. *Journal of the Operations Research Society of America*, 2(4):393–410. Available from: http://dx.doi.org/10.2307/166695.

Dantzig, G. B. and Ramser, J. H. (1959). The truck dispatching problem. *Management Science*, 6(1):80–91. Available from: http://dx.doi.org/10.2307/2627477.

de Boer, P.-T., Kroese, D., Mannor, S., and Rubinstein, R. (2005). A tutorial on the Cross-Entropy method. *Annals of Operations Research*, 134(1):19–67. Available from: http://dx.doi.org/10.1007/s10479-005-5724-z.

Delle Site, P., Filippi, F., and Usami, D. (2005). Design of operations of personal rapid transit systems. In *Proceedings of the 10th Meeting of the Euro Working Group on Transportation*. Available from: http://www.iasi.cnr.it/ewgt/16conferencePROC.html.

Dumas, Y., Desrosiers, J., Gelinas, E., and Solomon, M. M. (1995). An optimal algorithm for the traveling salesman problem with time windows. *Operations Research*, 43(2):367–371. Available from: http://dx.doi.org/10.2307/171843.

Dumas, Y., Desrosiers, J., and Soumis, F. (1991). The pickup and delivery problem with time windows. *European Journal of Operational Research*, 54(1):7–22. Available from: http://dx.doi.org/10.1016/0377-2217(91)90319-Q.

Featherstone, C. T. (2005). *Automatic Transport System and Vehicle Control Design for Passenger Comfort and Safety*. PhD thesis, University of Bristol, Department of Aerospace Engineering.

Fichter, D. (1964). *Individualized Automatic Transit and the City*. B. H. Sikes, 1430 East 60th Place, Chicago, Illinois 60637. Available from: http://www.worldcat.org/oclc/4872209.

Fischetti, M., Laporte, G., and Martello, S. (1993). The delivery man problem and cumulative matroids. *Operations Research*, 41(6):1055–1064. Available from: http://dx.doi.org/10.2307/171600.

Ford, B. M., Roesler, W. J., and Waddell, M. C. (1972). Vehicle management for PRT systems. In Anderson, J. E., editor, *Personal*

Rapid Transit, pages 411–434. University of Minnesota, Audio Visual Library Services.

Friggstad, Z., Salavatipour, M. R., and Svitkina, Z. (2010). Asymmetric traveling salesman path and directed latency problems. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '10, pages 419–428, Philadelphia, PA, USA. Society for Industrial and Applied Mathematics. Available from: http://portal.acm.org/citation.cfm?id=1873636.

Galassi, M., Davies, J., Theiler, J., Gough, B., Jungman, G., Alken, P., Booth, M., and Rossi, F. (2010). *GNU Scientific Library Reference Manual*. Network Theory Ltd, 1.14 edition.

Golden, B. L., Raghavan, S., and Wasil, E. A. (2008). *The vehicle routing problem latest advances and new challenges*. Springer. Available from: http://www.worldcat.org/isbn/9780387777788.

Gupta, V., Harchol-Balter, M., Dai, J., and Zwart, B. (2010). On the inapproximability of M/G/k: whytwomoments of job size distribution arenotenough. *Queueing Systems*, 64(1):5–48. Available from: http://dx.doi.org/10.1007/s11134-009-9133-x.

Homerick, D. J. (2010). PRT-SIM: An Open-Source microsimulator for personal rapid transit systems. Master's thesis, University of California Santa Cruz.

Horn, M. (2002a). Fleet scheduling and dispatching for demand-responsive passenger services. *Transportation Research Part C: Emerging Technologies*, 10(1):35–63. Available from: http://dx.doi.org/10.1016/S0968-090X(01)00003-1.

Horn, M. (2002b). Multi-modal and demand-responsive passenger transport systems: a modelling framework with embedded control systems. *Transportation Research Part A: Policy and Practice*, 36(2):167–188. Available from: http://dx.doi.org/10.1016/S0965-8564(00)00043-4.

Hvattum, L. M., Løkketangen, A., and Laporte, G. (2006). Solving a dynamic and stochastic vehicle routing problem with a sample scenario hedging heuristic. *Transportation Science*, 40(4):421–438. Available from: http://dx.doi.org/10.1287/trsc.1060.0166.

Irving, J. H., Bernstein, H., Olson, C. L., and Buyan, J. (1978). *Fundamentals of Personal Rapid Transit*. Lexington Books, D.C. Heath and Company.

Ishii, T., Iguchi, M., and Koshi, M. (1976). CVS: Computer-Controlled vehicle system. In Gary, D. A., Gary, M. J., Kornhauser, A. L., and Garrard, W. L., editors, *Personal Rapid Transit III*. University of Minnesota, Audio Visual Library Services.

Johnson, R. E., Walter, H. T., and Wild, W. A. (1976). Analysis and simulation of automated vehicle stations. In Gary, D. A., Girrard, W. L., and Kornhauser, A. L., editors, *Personal Rapid Transit III*, pages 269–281. University of Minnesota, Minneapolis.

Kall, P. and Wallace, S. W. (1997). *Stochastic programming*. Wiley. Available from: http://www.worldcat.org/isbn/9780471951087.

Kaplan, H., Lewenstein, M., Shafrir, N., and Sviridenko, M. (2005). Approximation algorithms for asymmetric TSP by decomposing directed regular multigraphs. *J. ACM*, 52:602–626. Available from: http://dx.doi.org/10.1145/1082036.1082041.

Laarhoven, P. J. M. and Aarts, E. H. L. (1987). *Simulated annealing: theory and applications*. Kluwer Academic Publishers, Norwell, MA, USA. Available from: http://portal.acm.org/citation.cfm?id=59580.

Latour, B. (1996). *Aramis, or, The love of technology*. Harvard University Press. Available from: http://www.worldcat.org/isbn/9780674043237.

Lee, D.-H., Wang, H., Cheu, R., and Teo, S. (2004). Taxi dispatch system based on current demands and Real-Time traffic conditions. *Transportation Research Record: Journal of the Transportation Research Board*, 1882:193–200. Available from: http://dx.doi.org/10.3141/1882-23.

Lees-Miller, J. D., Hammersley, J., and Davenport, N. (2009). Ride sharing in personal rapid transit capacity planning. In Griebenow, R. R., editor, *Automated People Movers 2009*, pages 321–332, Reston, VA. American Society of Civil Engineers.

Lees-Miller, J. D., Hammersley, J. C., and Wilson, R. E. (2010). Theoretical maximum capacity as benchmark for empty vehicle redistribution in personal rapid transit. *Transportation Research Record:*

Journal of the Transportation Research Board, 2146:76–83. Available from: http://dx.doi.org/10.3141/2146-10.

Lees-Miller, J. D. and Wilson, R. E. (2011). Sampling for personal rapid transit empty vehicle redistribution. *Transportation Research Record: Journal of the Transportation Research Board*.

Lees-Miller, J. D. and Wilson, R. E. (2012). Proactive empty vehicle redistribution for personal rapid transit and taxis. *Transportation Planning and Technology*.

Levy, G. (1976). The french aramis system. In Gary, D. A., Gary, M. J., Kornhauser, A. L., and Garrard, W. L., editors, *Personal Rapid Transit III*. University of Minnesota, Audio Visual Library Service.

Li, S. (2006). Multi-attribute taxi logistics optimization. Master's thesis, Massachusetts Institute of Technology. Available from: http://dspace.mit.edu/handle/1721.1/35112.

Lowson, M. (2003). New approach to effective and sustainable urban transport. *Transportation Research Record*, 1838:42–49.

Lowson, M. (2004). Idealised models for public transport systems. *International Journal of Transport Management*, 2(3-4):135–147. Available from: http://dx.doi.org/10.1016/j.ijtm.2005.05.001.

Lowson, M. V. (1999). Personal public transport. *Proceedings of the Institution of Civil Engineers-Transport*, 135(3).

Mendez-Diaz, I., Zabala, P., and Lucena, A. (2008). A new formulation for the traveling deliveryman problem. *Discrete Applied Mathematics*, 156(17):3223–3237. Available from: http://dx.doi.org/10.1016/j.dam.2008.05.009.

Miller, C. E., Tucker, A. W., and Zemlin, R. A. (1960). Integer programming formulation of traveling salesman problems. *J. ACM*, 7(4):326–329. Available from: http://dx.doi.org/10.1145/321043.321046.

Miller, D. L. and Pekny, J. F. (1991). Exact solution of large asymmetric traveling salesman problems. *Science*, 251(4995):754–761. Available from: http://dx.doi.org/10.2307/2875169.

Mitrovic-Minic, S., Krishnamurti, R., and Laporte, G. (2004). Double-horizon based heuristics for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B: Methodological*, 38(8):669–685. Available from: http://dx.doi.org/10.1016/j.trb.2003.09.001.

Munson, A. V. (1972). Quasi-synchronous control of High-Capacity PRT networks. In Anderson, J. E., Dais, J. L., Garrard, W. L., and Kornhauser, A. L., editors, *Personal Rapid Transit*. University of Minnesota, Audio Visual Library Services.

Nagarajan, V. and Ravi, R. (2008). The directed minimum latency problem. In Goel, A., Jansen, K., Rolim, J. D. P., and Rubinfeld, R., editors, *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques.*, volume 5171 of *Lecture Notes in Computer Science*, pages 193–206. Springer. Available from: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.140.7480.

Oncan, T., Altinel, I., and Laporte, G. (2009). A comparative analysis of several asymmetric traveling salesman problem formulations. *Computers & Operations Research*, 36(3):637–654. Available from: http://dx.doi.org/10.1016/j.cor.2007.11.008.

Powell, W. B. (1987). An operational planning model for the dynamic vehicle allocation problem with uncertain demands. *Transportation Research Part B: Methodological*, 21(3):217–232. Available from: http://dx.doi.org/10.1016/0191-2615(87)90005-1.

Powell, W. B. (1996). A stochastic formulation of the dynamic assignment problem, with an application to truckload motor carriers. *Transportation Science*, 30(3):195–219. Available from: http://dx.doi.org/10.1287/trsc.30.3.195.

Powell, W. B. (2007). *Approximate dynamic programming : solving the curses of dimensionality*. Wiley-Interscience. Available from: http://www.worldcat.org/isbn/9780470171554.

Psaraftis, H. N. (1995). Dynamic vehicle routing: Status and prospects. *Annals of Operations Research*, 61(1):143–164. Available from: http://dx.doi.org/10.1007/BF02098286.

Puterman, M. L. (2005). *Markov decision processes : discrete stochastic dynamic programming*. Wiley-Interscience. Available from: http://www.worldcat.org/isbn/9780471727828.

Rydell, E. W. F. (2001). *The transportation Renaissance : the personal transit solution*. Distributed by Xlibris]. Available from: http://www.worldcat.org/isbn/9781401043643.

Sarubbi, J. F. M. and Luna, H. P. L. (2007). A new ow formulation for the minimum latency problem. In *International Network Optimization Conference*. Available from: http://www.euro-online.org/enog/inoc2007/Papers/author.31/paper/paper.31.pdf.

Schweizer, J., Danesi, A., Mantecchini, L., Traversi, E., and Caprara, A. (2010). Towards a PRT capacity manual. In *PRT@LHR Conference Proceedings*.

Schweizer, J. and Mantecchini, L. (2007). Performance analysis of large scale PRT networks: theoretical capacity and micro-simulations. In *Automated People Movers 2007*, pages 1–11.

Seow, K. T., Dang, N. H., and Lee, D.-H. (2010). A collaborative multiagent Taxi-Dispatch system. *IEEE Transactions on Automation Science and Engineering*, 7(3):607–616. Available from: http://dx.doi.org/10.1109/TASE.2009.2028577.

Sirbu, M. A. (1974). Station configuration, network operating strategy and station performance. In Anderson, J. E. and Romig, S., editors, *Personal Rapid Transit II*, pages 461–478. University of Minnesota, Audio Visual Library Services.

Strakosch, G. R. (1998). *The Vertical Transportation Handbook*. Wiley.

Sutton, R. S. and Barto, A. G. (1999). *Reinforcement learning : an introduction*. MIT Press. Available from: http://www.worldcat.org/isbn/9780262193986.

Swihart, M. R. and Papastavrou, J. D. (1999). A stochastic and dynamic model for the single-vehicle pick-up and delivery problem. *European Journal of Operational Research*, 114(3):447–464. Available from: http://dx.doi.org/10.1016/S0377-2217(98)00260-4.

Szillat, M. T. (2001). *A Low-level PRT Microsimulation*. PhD thesis, University of Bristol. Available from: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.112.1156.

Tijms, H. (2006). New and old results for the M/D/c queue. *AEU - International Journal of Electronics and Communications*, 60(2):125–130. Available from: http://dx.doi.org/10.1016/j.aeue.2005.11.008.

Toth, P. and Vigo, D. (2002). *The vehicle routing problem.* SIAM monographs on discrete mathematics and applications. Society for Industrial and Applied Mathematics. Available from: http://www.worldcat.org/isbn/9780898714982.

ULTra PRT (2008). ATS/CityMobil simulation installation guide and tutorial. Accessed 5 May, 2011. Available from: http://www.citymobil-project.eu/site/en/documenten.php.

ULTra PRT (2010). ULTra at london heathrow airport. Accessed July 31. Available from: http://www.ultraprt.com/applications/existing-systems/heathrow/.

van Eijl, C. A. (1995). A polyhedral approach to the delivery man problem. Technical Report COSOR 95-19, Eindhoven University of Technology.

Waddell, M. C., Williams, M. B., and Ford, B. M. (1973). *Disposition of empty vehicles in a personal rapid transportation system : interim report.* Johns Hopkins University, Applied Physics Laboratory. Available from: http://www.worldcat.org/oclc/13294904.

Wang, H., Lee, D.-H., and Cheu, R. (2009). PDPTW based taxi dispatch modeling for booking service. In *Fifth International Conference on Natural Computation*, pages 242–247. IEEE. Available from: http://dx.doi.org/10.1109/ICNC.2009.676.

Wesselowski, K. and Cassandras, C. G. (2006). The elevator dispatching problem: Hybrid system modeling and receding horizon control. In Cassandras, C., Giua, A., Seatzu, C., and Zaytoon, J., editors, *Analysis and Design of Hybrid Systems 2006: a Proceedings volume from the 2nd IFAC Conference*, pages 136–141. Elsevier.

Williams, H. P. (1999). *Model Building in Mathematical Programming.* John Wiley & Sons Ltd, fourth edition.

Won, J.-M., Choe, H., and Karray, F. (2006). Optimal design of personal rapid transit. In *Intelligent Transportation Systems Conference*, pages 1489–1494. Available from: http://dx.doi.org/10.1109/ITSC.2006.1707434.

Xithalis, C. (2011). Hermes network simulator. Available from: http://students.ceid.upatras.gr/~xithalis/simulation_en.html.

Yang, H., Wong, S., and Wong, K. (2002). Demand-supply equilibrium of taxi services in a network under competition and regulation. *Transportation Research Part B: Methodological*, 36(9):799–819. Available from: http://dx.doi.org/10.1016/S0191-2615(01)00031-5.

Yang, J., Jaillet, P., and Mahmassani, H. (2004). Real-Time multivehicle truckload pickup and delivery problems. *Transportation Science*, 38(2):135–148. Available from: http://dx.doi.org/10.1287/trsc.1030.0068.

York, H. L. (1974). The simulation of a PRT system operating under Quasi-Synchronous control. In Anderson, J. E. and Romig, S., editors, *Personal Rapid Transit II*. University of Minnesota, Audio Visual Library Services.