Heuristics for New Problems Arising in the Transport of People and Goods

Alan Lee

A thesis submitted for the degree of
Doctor of Philosophy at
The Australian National University

May 2017

© Alan Lee 2015

Although I worked with my supervisors and members of my supervisory panel on the work presented, the methods described in the thesis were my own invention.

Alan Lee

May 27, 2017

In memory of Clarence Easton Brown, 1924-2014.

Acknowledgements

Like all monumental undertakings, there are a large number of people without whom, this would never have been possible. First, I must give my sincerest thanks to my two primary supervisors, Martin Savelsbergh and Philip Kilby. Martin, you took in a wide eyed student and helped shape him into something resembling a real researcher. Finishing two papers within 16 months of starting requires a huge amount of dedication and skill, which I will always be grateful for. Phil, taking in someone else's graduate student halfway through their Ph.D. is an incredibly generous thing to do, and the level of insight you showed during the second half of my thesis was truly amazing. I must also give thanks to the other members of my supervisory panels; to Natashia Boland and Alex Mendes at the University of Newcastle, and to Pascal van Hentenryck at the Australian National University. I must also acknowledge the various academics, reviewers and students who have provided valuable feedback and advice throughout my studies. Finally, a thank you to the supervisor of my Honors thesis, Andrea Raith, for sparking my interest in research projects and showing me how much fun they can be.

I also need to thank my family for their unconditional love and support. You encouraged me to pursue my dreams overseas, even though it meant missing several precious family events, and I will always be grateful for that.

I need to thank my New Zealand based friends, Aaron Smith and Mostafa Biglari-Abhari, for your friendship and support. Thank you to my office mates in Newcastle; Simranjt Kaur, Reena Kapoor and Fabian Rigterink – you helped ground and center me during the first part of my thesis and gave me friends in every continent. Thanks also to my fellow students in Canberra; Fazlul Siddiqui, Terrence Mak, Cody Christopher and Hadi Afshar – you welcomed me into the group and made the transition as smooth as possible. I also need to acknowledge the researchers who offered their friendship and support, particularly Menkes van den Briel, Hassan Hijazi, Alban Grastien, Michael Norrish, Patrik Haslum, Scott Sanner and Justin Domke. I would like to thank my Canberra flatmates; Ade Prastyani, Jeremy Hall, Lucy Ellen and Chiara Edwards – I've lived in multiple flats throughout my studies, but ours was certainly my favorite.

Finally, thank you to the University of Newcastle, the Australian National University and NICTA for accepting my applications and offering me scholarships to complete my studies.

Abstract

The Vehicle Routing Problem (VRP) and its numerous variants are amongst the most widely studied in the entire Operations Research literature, with applications in fields including supply chain management, journey planning and vehicle scheduling. In this thesis, we focus on three problems from two fields with a wide reach; the design of public transport systems and the robust routing of delivery vehicles. Each chapter investigates a new setting, formulates an optimization problem, introduces various solution methods and presents computational experiments highlighting salient points.

The first problem involves commuters who use a flexible shuttle service to travel to a main transit hub, where they catch a fixed route public transport service to their true destination. In our variant, passengers must forgo some of the choices they had in previous versions; the service provider chooses the specific hub passengers are taken to (provided all relevant timing constraints are satisfied). This introduces both complexities and opportunities not seen in other VRP variants, so we present two solution methods tailored for this problem. An extensive computational study over a range of networks shows this flexibility allows significant cost savings with little impact on the quality of service received.

The second problem involves dynamic ridesharing schemes and one of their most persistent drawbacks: the requirement to attract a large number of users during the start up phase. Although this is influenced by many factors, a significant consideration is the perceived uncertainty around finding a match. To address this, the service provider may wish to employ a small number of their own private drivers, to serve riders who would otherwise remain unmatched. We explore how this could be formulated as an optimization problem and discuss the objectives and constraints the service provider may have. We then describe a special structure inherent to the problem and present three different solution methods which exploit this. Finally, a broad computational study demonstrates the potential benefits of these dedicated drivers and identifies environments in which they are most useful.

The third problem comes from the field of logistics and involves a large delivery firm serving an uncertain customer set. The firm wishes to build low cost delivery routes that remain efficient after the appearance and removal of some customers. We formulate this problem and present a heuristic which is both computationally cheaper and more versatile than comparative exact methods. A wide computational study illustrates our heuristic's predictive power and its efficacy compared to natural alternative strategies.

Glossary

- **VRP:** Vehicle Routing Problem
- **CVRP:** Capacitated Vehicle Routing Problem
- **VRPTW:** Vehicle Routing Problem with Time Windows
- **MDVRP:** Multiple Depot Vehicle Routing Problem
- **VRPM:** Vehicle Routing Problem with Multiple Use of Vehicles
- **VRPMTW:** Vehicle Routing Problem with Multiple Time Windows
- **SVRP:** Selective Vehicle Routing Problem
- **DVRP:** Dynamic Vehicle Routing Problem
- **SVRP:** Stochastic Vehicle Routing Problem
- **MOVRP:** Multi-Objective Vehicle Routing Problem
- **DARP:** Dial A Ride Problem
- **TWAVRP:** Time Window Assignment Vehicle Routing Problem
- **ConVRP:** Consistent Vehicle Routing Problem
- **FSMVRP:** The Fleet Size and Mix Vehicle Routing Problem
- **SA:** Simulated Annealing
- **TS:** Tabu Search
- **VNS:** Variable Neighborhood Search
- **LNS:** Large Neighborhood Search
- **GA:** Genetic Algorithm
- **CP:** Constraint Programming
- **DRC:** Demand Responsive Connector
- **EDRC:** Extended Demand Responsive Connector
- **DRS:** Dynamic Ride Sharing
- **DRSDD:** Dynamic Ride Sharing with Dedicated Drivers
- **VRPFI:** Vehicle Routing Problem with Future Insertions

List of Publications

Work from this thesis has already appeared in the following publications:

- Lee, A., & Savelsbergh, M. (2014). An extended demand responsive connector. EURO Journal on Transportation and Logistics, 1-26.
- Lee, A., & Savelsbergh, M. (2015). Dynamic ridesharing: Is there a role for dedicated drivers?. Transportation Research Part B: Methodological, 81, 483-497.
- Lee, A., & Kilby, P. (2015). Estimating the capability for future insertions in logistics schedules. Odysseus 2015: Sixth International Workshop on Freight Transportation and Logistics, France, May 2015.

Contents

Acknowledgements **vii**

Abstract **ix**

Glossary **xi**

List of Publications **xiii**

1 Introduction **1**

1.1 Background and Motivations 1

1.2 Research Objectives and Contributions 2

1.3 Thesis Outline 4

2 Background **5**

2.1 The Vehicle Routing Problem 5

2.2 Related Problems 10

2.3 Solution Methods 12

2.4 Applications 34

2.5 Summary 48

3 The Extended Demand Responsive Connector **49**

3.1 Introduction 49

3.2 The Extended Demand Responsive Connector (DRC) 49

3.3 An Integer Programming Formulation 53

3.4 A Heuristic Algorithm 57

3.5 Instance Generation 64

3.6 Computational Study 66

3.7 Conclusion 72

4 Dynamic Ride Sharing with Dedicated Drivers 73

4.1 Introduction 73

4.2 Motivation 73

4.3 Reformulation as a Transshipment Problem 76

4.4 DRS with Dedicated Drivers (DRSDD) 78

4.5 Exact Methods for the DRSDD 86

4.6 A Heuristic Algorithm 94

4.7 Instance Generation 104

4.8 Computational Study 106

4.9 Conclusion 115

5 Estimating the Robustness of Logistic Schedules to Future Insertions 117

5.1 Introduction 117

5.2 Motivation 117

5.3 The Vehicle Routing Problem with Future Insertions (VRPFI) 118

5.4 A Heuristic Approach 120

5.5 Instance generation 124

5.6 Computational Study 126

5.7 Conclusion 131

6 Conclusion **133**

6.1 Research Summary 133

6.2 EDRC Conclusions and Future Work 133

List of Figures

2.1 Example of a Vehicle Routing Problem (VRP) 7

2.2 Different branching strategies 13

2.3 Example branch and bound trees 14

2.4 A subtour inside a VRP solution 15

2.5 Illustration of Clarke and Wright's savings algorithm 24

2.6 Relocate and Swap moves 26

2.7 2–opt moves 27

2.8 Operational layout for a DRC 36

2.9 The time line of a single passenger 38

2.10 Potential matching between a driver and a rider. 41

2.11 Example matching between ride sharing participants 45

3.1 Network reduction arising from the EDRC 50

3.2 Departure time for passengers in different locations 51

3.3 The time line of a single passenger 52

3.4 Multiple equivalent solutions 56

3.5 Alternative network representation 56

3.6 Box obtained from locations l^1 and l^2 and used to define the RCL 59

3.7 Moving passengers within a route 61

3.8 Moving passengers between routes 62

3.9 Example of how an EDRC operates 64

3.10 Mapping arrival times to train services 65

3.11 Optimal and heuristic solution for instance with biggest optimality gap . . . 67

4.1 Network that enforces return trip constraints 77

4.2 Small network to illustrate difference between operational and net cost . . . 82

4.3 Example network to compare objectives 83

4.4 Network reduction techniques used 87

4.5 Counter example for Rule Two 102

4.6 Moving passengers within a route 103

4.7 Diagram of networks with hubs 105

5.1 Area in which a new customer can be inserted for a given slack time 121

5.2 Score calculation for a complete route 122

5.3 Areas in which insertions are profitable 123

5.4 Change in area of ellipse per extra unit of slack time 124

5.5 Correlation between EAS and simulation score for a single instance 128

List of Tables

3.1 Parameters defining classes of instances 65

3.2 Difference between optimal and heuristic solution with five passengers . . . 66

3.3 Difference between optimal and heuristic solution with ten passengers 68

3.4 Improvement resulting from the use of pseudo–schedules 69

3.5 Solution statistics 69

3.6 Benefits of the EDRC over the traditional variant 70

3.7 Indicators of service quality for passengers 71

3.8 Indicators of interest to the service provider 71

3.9 Savings made by increasing the number of stations under the EDRC 72

4.1 Construction parameters used 96

4.2 Shaking parameters used 99

4.3 Parameters defining classes of instances 106

4.4 Values of heuristic parameters trialed 107

4.5 Comparison of heuristic solutions and optimal solutions 108

4.6 Average number of pairs of participants removed from the problem 109

4.7 Fraction of riders that can be served by ad hoc drivers 110

4.8 Minimum number of riders that must be served 110

4.9 Performance statistics of dedicated drivers 111

4.10 Number of instances for which no dedicated drivers were needed 112

4.11 Average distance between a rider's origin and destination 113

4.12 Decrease in total cost from lowering the required service level 113

4.13 Effect of lowering $\hat{s}$ on the average per-rider cost 114

4.14 Effect of lowering $\hat{s}$ on the average OD distance of riders served 114

4.15 Percentage reduction resulting from greater flexibility 115

5.1 Differences compared to the best known objective value 127

5.2 Difference from best known objective values with minimum vehicles 127

5.3 Ratio of best to worst simulation score across the instance classes 128

5.4 Correlation coefficient between EAS and simulation score 128

5.5 Comparing simulation score of other strategies against greatest EAS 130

5.6 Comparing cost of alternative strategies against greatest EAS 130

5.7 Comparing solutions with maximum EAS and maximum slack time 131

Chapter 1

Introduction

1.1 Background and Motivations

Of the countless problems studied by Operations Research (OR) practitioners, those involving the efficient routing of vehicles are perhaps the easiest to observe in practice. Every time we travel anywhere, no matter the length or mode, we implicitly rely on a whole suite of optimization tools. If we travel by car, we might use a GPS to find the fastest route (updated using real time information) or pass through optimized traffic signals. Alternatively, those using public transport benefit from optimized schedules and catchment areas, as well as carefully designed routes and staffing schedules. Although less obvious to the casual observer, the movement of goods has also been carefully scrutinized by optimization practitioners seeking to minimize cost and maximize throughput. They study every part of the production cycle, from the transport of raw materials to customer delivery. The ubiquitous nature of these problems means even a small gain or improvement in the knowledge surrounding Vehicle Routing Problems (VRPs) can have significant impact, whether through faster journeys for travelers, or by reducing the costs associated with delivering goods.

1.1.1 The Transport of People

Key transport infrastructure has always had a large impact on the location and design of cities worldwide. Historically, settlements were often centered around large waterways (to allow easy movement by boat); in more modern times, the availability of private automobiles led to the development of large roading networks. The increased mobility and freedom offered by vehicles certainly improved the living standard of many people and could be considered one of the defining features of the era. However, as the combination of falling vehicle prices and rising population levels increased vehicle ownership rates, demand for these networks has risen beyond what could reasonably be catered for, leading to the well known problem of vehicular congestion. In response, cities started to offer mass transit systems, with mixed success. Oftentimes, the city simply wasn't sufficiently populated to justify a service with adequate frequencies or coverage to entice commuters out of their vehicles. This form of transport "no–man's land" motivates the development of systems which offer some of the flexibility of personal transit, while still alleviating congestion by removing cars from the road. Such approaches may increase the flexibility of traditional public transport services by allowing small deviations from fixed routes (Bruun and Marx, 2006), or encourage full utilization of private vehicles (e.g., ride sharing, car sharing, taxi sharing). However, these schemes often incur an additional cost, which can

become prohibitive if the scheme is poorly designed, or if the main factors driving the cost are not well understood. We identify ways to extend existing schemes to achieve the correct balance between user convenience and cost efficiency and we believe our variants offer effective yet viable public transit alternatives for mid–sized cities.

1.1.2 The Movement of Freight

Facilitating the efficient movement of goods is an important concern for governments and businesses (wanting to facilitate economic productivity) and consumers (who want their product as soon as possible). The size, scope and potential impact of work in this area led to the development of several academic fields studying related problems, ranging from the design of transport networks to the scheduling of workers at port terminals and container yards. One core problem faced by delivery businesses on a daily basis is that of designing the specific routes taken by service vehicles. Most firms aim to minimize routing cost, but others may aim to minimize vehicular emissions or to evenly distribute work amongst drivers. Unfortunately, regardless of the objective chosen, creating optimal delivery schedules remains a very difficult task. For this reason, those faced with industrial scale problems typically use *heuristics*, approximate methods which often find a good solution, but offer no guarantee as to their quality (although empirically, we may have reason to believe they perform well). The main advantage is computational speed; exact methods are simply too slow for problems of a realistic size. This increased speed allows researchers to focus on the additional constraints and issues that separate real world problems from pure academic ones. One such issue is that of dynamic customers, where the delivery firm must design routes knowing that the set of customers will change over time; after a decision has been made, there is little opportunity to alter it. Once new customers are revealed, they may only be served if doing so does not violate previous commitments, and if an existing customer leaves, remaining customers cannot be reshuffled to fill the gap. This means there is a strong incentive to design schedules that are both low cost and robust to future changes. Research in this area allows delivery firms to offer certainty of service to their original customers, while still growing their business by serving new customers.

1.2 Research Objectives and Contributions

Our contributions are three–fold; we introduce two novel public transport systems and address the problem of unknown customers in logistic settings with fixed delivery schedules. The first problem is that of the Extended Demand Responsive Connector (EDRC), a shuttle van which transports passengers from private residences to public transport hubs. Previous variants had overly restrictive rules regarding the location vehicles are dispatched from; we propose a variant where these are relaxed. This makes the problem more complex while also providing the potential for greater efficiency. The complexity arises because the time window constraints governing a passengers arrival at a station are a function of both the passenger and the station – this linked dependence is not seen in other problems and we propose novel solution methods to overcome this. We also illustrate that our variant offers lower operational costs and investigate factors which affect the size of

potential savings. Finally, we give insights into the trade–off between operational costs and the quality of service received.

The second problem relates to ride sharing systems, where participants traveling along similar routes are matched and travel together. However, when the system is still becoming established, low participation rates may mean there is a small probability of finding a match and this uncertainty could discourage some participants from joining the scheme. To alleviate this, some operators allow riders to specify they only want to be matched with a driver if they are guaranteed a return journey; previously, this was thought to require a general integer program, but we show that under minimal additional assumptions, we can use much simpler methods. We then propose a variant where the service provider employs a pool of drivers to meet requests that would otherwise go unsatisfied. We explore some of the equity issues surrounding this and discuss the merits of various funding mechanisms from both a practical and political point of view. We describe three alternative solution methods and discuss practical ways to reduce their solve times. We present computational experiments that show the need for such drivers depends on certain network characteristics; specifically, the number of participants in a given area, the flexibility of participants with respect to time windows, and the spread of origins and destinations.

The third problem concerns the design of delivery schedules, knowing the set of customers will change. We discuss ways to frame this as an optimization problem and explore what might be considered an "optimal solution". We propose a novel, geometric heuristic that can predict the robustness of a route via a simple numerical score and investigate its accuracy with computational experiments. We then explain how this score can be combined with traditional meta–heuristic solvers to construct and improve routes. Finally, we show that choosing robust routes can lead to significant increases in profitability and that our heuristic can do this with greater accuracy than natural alternative strategies.

Intrinsically, all three problems share commonalities along three main vectors. First, the routing nature of these problems mean they share many intrinsic similarities; at its core, the problem of transporting people isn't all that different from transporting goods. Both involve customers with specific time windows, require efficient routing decisions to be made, and have some inherent tradeoff between operational cost and customer satisfaction. In practice, all problems suffer from unpredictable, real world effects, e.g., stochastic traffic delays, last minute customer cancellations and existing legislative limitations. Additionally, all problems are designed for passenger/customer densities below a certain limit; if densities exceed this threshold, then the benefit from using our alternative models are reduced. In Chapter 4, we extensively discuss the concept of a critical participation threshold; if this is exceeded, then traditional formulations perform reasonably well. Similar arguments are made for the other problems. Finally, all the problems are often solved using similar techniques. Within the context of this thesis, we utilized similar neighborhood search meta heuristics for all three problems; within the broader Operations Researcher literature, all three problems have been studied using a shared pool of meta–heuristics and integer programming techniques.

1.3 Thesis Outline

This thesis is organized as follows:

- In Chapter 1 we introduce the thesis and give an overview.
- In Chapter 2 we give the required background knowledge and review the surrounding literature.
- In Chapter 3 we introduce the Extended Demand Responsive Connector and present our findings.
- In Chapter 4 we present the concept of Dynamic Ride Sharing with dedicated drivers and present our results.
- In Chapter 5 we introduce our work on designing robust logistics schedules and present our findings.
- In Chapter 6 we summarize our main contributions and present future research directions.

Chapter 2

Background

In this chapter we formally present the Vehicle Routing Problem (VRP) and survey the related literature. We will detail its numerous variants, focusing on those studied later in this thesis. We also give the common solution methods, again with a specific focus on those used in subsequent chapters.

2.1 The Vehicle Routing Problem

The Vehicle Routing Problem (VRP) is one the most widely studied problems in the field of Operations Research (OR) and has been adapted to countless real world settings. In its classical form, the VRP consists of a set of customers, denoted by $C = \{1, 2, \ldots, n\}$, surrounding a single depot containing both goods and a fleet of vehicles, $K = \{1, 2, \ldots, m\}$. A *solution* to the VRP is an ordered list of customers to whom each vehicle will deliver goods; for a single vehicle, such a list is referred to as a *route* or a *schedule*. To illustrate this, an example of a VRP and a possible solution is given in Figure 2.1. We also have some objective function, f, which takes a solution and returns a numerical score indicating some solution characteristic. The standard objective is to minimize the total cost incurred across all vehciles, where "cost" is some context specific combination of the number of vehicles used, the total distance traveled and the total time spent. Without loss of generality, we seek the solution with the smallest objective value, called the *optimal solution*, (in settings where larger values are preferable, we simply use the negative of the objective function). The VRP is well-known as a NP-hard problem even under very basic conditions (Karp, 1972), so unless the famous "P = NP" conjecture is proven true (Cook, 1971), finding an exact solution is computationally very difficult for problems of a non–trivial size. For context, current state of the art methods can solve VRP instances with up to 150 customers (Subramanian, 2012), but this assumes an advanced implementation, large amounts of computational time and no complicating side constraints. In the special case when the VRP contains a single vehicle, the problem is referred to as the *Traveling Salesman Problem* (TSP); similarly, the VRP is also known as the *Multiple Traveling Salesman Problem* (MTSP).

Formally, the VRP can be represented on a graph $G(\mathbf{N}, \mathbf{A})$ with a cost matrix F, where:

- $\mathbf{N} = \{n_0, n_1, \ldots, n_n\}$ is the node set, where node n_0 represents the depot and nodes n_1 to n_n represent customers.
- $\mathbf{A} = \{(n_i, n_j) \mid i, j \in \mathbf{N}, i \neq j\}$ is the set of arcs between nodes, and each arc has an associated cost.
- F is a cost matrix, where element $c_{ij} \in F, i, j \in \mathbf{N}$ gives the cost of traveling from node i to node j.

A route can be represented as a vector of nodes from $\mathbf{N}$, which contains n_0 in both the first and last position. The order of intermediate nodes give the order in which customers are visited. A VRP is said to be symmetric if $c_{ij} \in C = c_{ji} \in C,\ i, j \in \mathbf{N}$. We assume the triangle inequality holds, e.g., $c_{ij} \leq c_{ik} + c_{kj},\ i, j, k \in \mathbf{N}$.

We will now give some of the common extensions and adaptations to the standard VRP setting. This list is not intended to be exhaustive, but instead should be considered an overview, with a specific focus on settings referenced in later sections. For a fuller summary of the VRP, see Laporte (2009).

2.1.1 The Capacitated VRP (CVRP)

The Capacitated Vehicle Routing Problem is one of the most common extensions and assumes that goods delivered to customers have some physical volume, while vehicles have some capacity. Naturally, the total volume of goods delivered by a truck must not exceed its capacity. Specifically, the volume of goods demanded by customer i is given by d_i and the capacity of a vehicle is given by Q. Equally, we could also impose a similar constraint on the weight of goods delivered. Some of the common extensions found in the literature are given below, but for a broader review, we direct the reader to Semet et al. (2014).

- **Multi–dimensional packing constraints:** In some settings, we may wish to model goods as a p dimensional object and vehicles as a p dimensional container (in many real world settings, there are $p = 3$ dimensions representing width, length and height). We must find a packing of goods into trucks such that no goods overlap in any dimension (it may be possible to rotate some parcels). In this setting, the set of goods demanded by each customer is represented by a series of p–tuples, where each tuple represents a single parcel. Similarly, the capacity of a vehicle is represented by a single p–tuple, $Q = (Q^0, Q^1, \ldots, Q^p)$.
- **Heterogeneous vehicles:** A natural extension is to allow vehicles to have differing capacities. In this case, the capacity of vehicle $k \in K$ is denoted by Q_k.
- **Vehicles with compartments:** We can assume a vehicle's storage compartment is divided into fixed compartments. Goods can be stored in any compatible compartment, provided the capacity of individual compartments are respected. Assume the

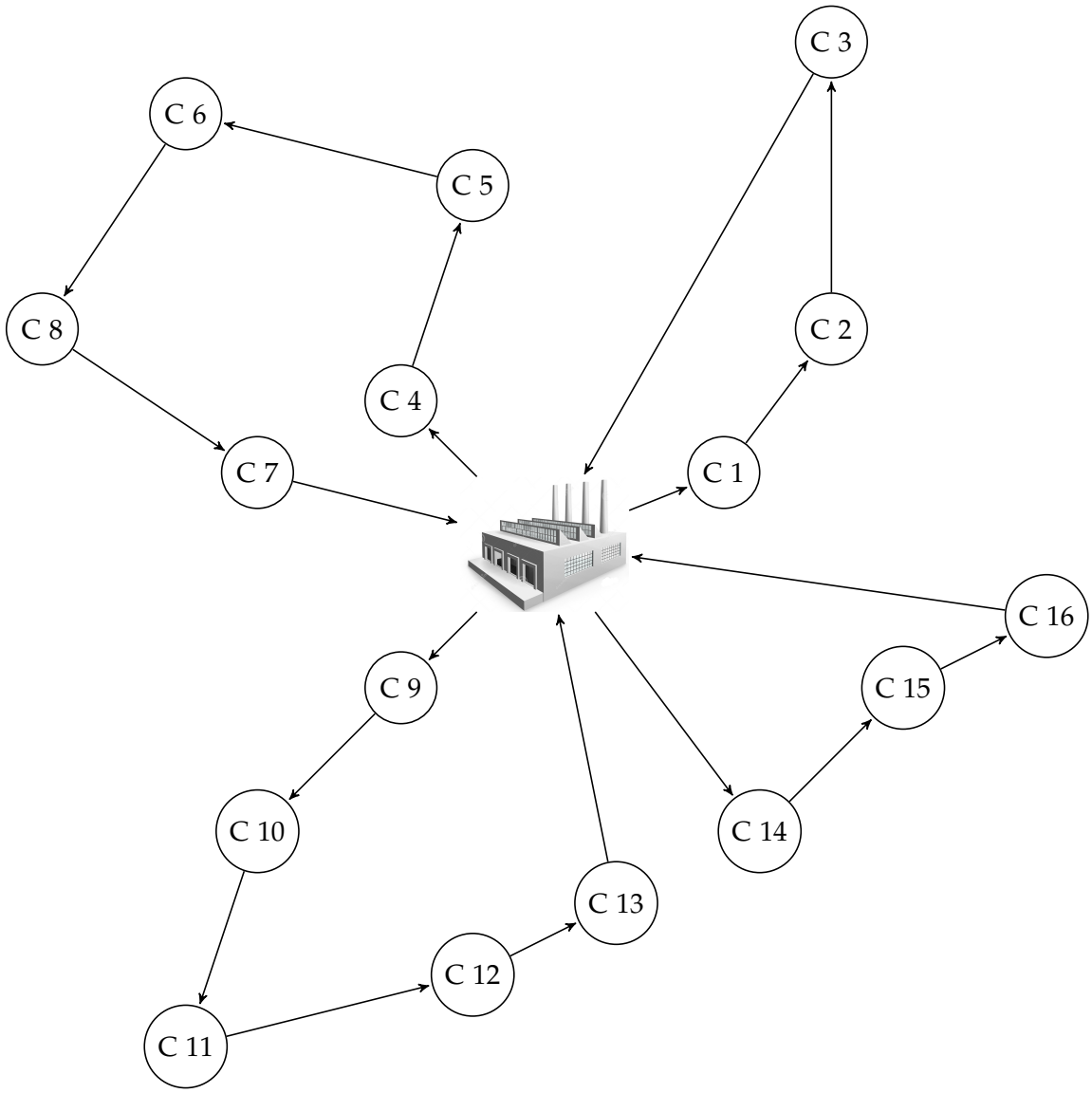


Figure 2.1: Example of a Vehicle Routing Problem (VRP)

set of compartments for vehicle $k \in K$ is given by $C(k)$ and the capacity for compartment $c \in C(k)$ is given by $Q_{c,k}$.

2.1.2 The VRP with Time Windows (VRPTW)

Another common variant is to assume customers have parameterized time windows in which delivery must begin. For customer $i \in C$, service must start in the interval given by $[e_i, l_i]$. In a similar fashion, we can have operational windows within which vehicle $k \in K$ must leave from and return to the depot, given by $[e_k, l_k]$. There is also a related variant involving *soft time windows*, where violation of time windows does not prevent feasibility, but is penalized in the objective function. For a survey on the VRPTW, we refer the reader to El-Sherbeny (2010) or Hashimoto et al. (2010).

2.1.3 Multiple Depot VRP (MDVRP)

To obtain efficiencies from operating at scale, many delivery firms operate from multiple depots simultaneously and can choose which depot customers are served from. This is common for firms that ship large quantities of goods (requiring multiple depots) but have a small number of product types (so all depots have the full range), e.g., fresh food distributors (Tarantilis and Kiranoudis, 2002). This is closely related to the variant containing intermediate facilities (VRPIF); the distinction being vehicles must start and end at a depot but can restock at intermediate facilities (Sevilla and de Blas, 2003).

2.1.4 The VRP with Multiple Use of Vehicles (VRPM)

In some settings, vehicles can make another trip upon returning to the depot, perhaps after reloading or changing drivers. For the original paper on this, see Taillard et al. (1995).

2.1.5 The VRP with Multiple Time Windows (VRPMTW)

In this setting, each customer is associated with multiple time windows and must be visited during exactly one of them (Fleischmann, 1990). This can arise when customers are incentivised to be flexible (through cheaper rates) or when planning an itinerary for a tourist group (where each "customer" represents an attraction with time windows indicating opening hours), as given in Dunstall et al. (2003).

2.1.6 Selective VRPs (SVRP)

In some settings, it is neither desirable or practical to serve all customers and the delivery firm must choose a subset to visit. We detail three natural variants of this problem below,

but for a full review we refer the reader to Feillet et al. (2005), Jozefowiez et al. (2008) or Archetti et al. (2013).

- **The Team Orienteering Problem (TOP):** In this setting, we have some limiting resource which prevents us from visiting all customers. This could be temporal e.g., an insufficient number of driver hours (Tang and Miller-Hooks, 2005), or be financial e.g., a budget on the operational cost of vehicles (Li, 2012). Each customer is assigned some weight, and the goal is to maximize the sum of weights of visited customers while observing this resource constraint.
- **The Prize Collecting Vehicle Routing Problem (PCVRP):** This can be described as the reverse of the OP, where the sum of weights (or prizes) of visited customers must equal or exceed some threshold and we wish to use as little of the corresponding resource as possible. This was initially designed to schedule jobs at a steel mill, where each customer represents a job (with a prize representing the profit earned upon completion), and the distance between locations giving the time required to configure the machine for the next job. The aim is to find a set of jobs that earn a minimum amount of profit in the shortest length of time (Balas, 1987). It is also used to model regulatory inspectors who are required to visit a certain percentage of sites and wish to do so at minimum cost.
- **The Vehicle Routing Problem with Profits (VRPP):** In this problem, the reward for visiting customers and the cost associated with traveling between two locations are of the same units, and the goal is to maximize the difference between the rewards gained and the costs incurred. One example is that of a delivery company who can choose which customers they serve and wants to find the subset that maximizes their net profit (Archetti et al., 2009).

2.1.7 The Dynamic VRP (DVRP)

In Dynamic VRPs, some information is not known at the start of the planning horizon, but is revealed over time (after some, but not all, routing decisions have already been made). A later example (studied in Chapter 5) is a problem faced by delivery firms who have a mixture of regular customers (with pre–booked orders known at the start of the planning horizon) and new customers (who appear partway through the planning horizon). The delivery firm must make routing decisions regarding the service of known orders while anticipating the unknown customers; once new information becomes available, the firm has an opportunity to make further decisions (although this may be restricted by previous choices).

2.1.8 The Stochastic VRP (SVRP)

In the Stochastic VRP some aspect of the problem is random and won't be known until all decisions have been made and finalized, i.e., unlike the DVRP, there is no chance to respond to the new information. Common examples include the travel time between two locations not being realized until after the journey, or the quantity a customer requires being

unknown before service begins. The aim is to minimize the *expected* cost across all possible outcomes, accounting for the relative likelihood of each scenario.

2.1.9 The Multi–Objective VRP (MOVRP)

In the Multi–Objective VRP, there are (at least) two competing aims that must be considered when scheduling vehicles, e.g., we may wish to minimize both the operational cost and the vehicular emissions produced. The exact method used for Multi–Objective VRPs depends on the relationship between the objectives, but we give a broad overview of three common variants, and refer interested readers to Ehrgott and Wiecek (2005) for a more comprehensive analysis.

- **Objective weights:** The objectives may be assigned numerical weights, which can be used to form a combined objective function. A standard VRP is then solved and a single solution is returned.
- **Hierarchical objectives:** If the objectives are given as an hierarchy, then it is sufficient to solve a series of single objective problems. First, solve the problem considering only the first objective, then re–solve considering only the second objective, but with a constraint preventing deterioration in the first objective. This process of changing and constraining objectives is repeated until all objectives have been considered and again, a single solution is found.
- **Pareto front:** If the objectives are neither ranked or weighted, then we generate a representative range of solutions, each with different trade–offs between the objectives. Formally, such a collection is referred to as a *pareto front*, where there is no solution s^1, that is at least as good in all objectives as another solution s^2, and is strictly better in at least one. This set of solutions is given to the relevant stakeholders, who then make the final decision. There are many different methods available to generate the pareto front; for a full list, see Ehrgott and Wiecek (2005).

2.2 Related Problems

There are many settings that have close links to the VRP, but are sufficiently different to justify their classification as related problems (a short summary of the most common problems is given below).

2.2.1 The Pickup and Delivery Problem (PDP)

Like the VRP, the Pickup and Delivery Problem involves a service provider using a fleet of vehicles to visit a set of pick up locations, often at minimum cost. Unlike the VRP, each pick up location has an associated delivery location, and the vehicle must carry a parcel between the two. This is often encountered by courier firms, who collect parcels from

clients and deliver them to their final destination (without consolidation at a depot). The VRP can be seen as a special case of the PDP, where all customers share a common pick up (or drop off) location (representing the depot). PDPs can involve the same extensions given above for the VRP (like capacitated vehicles, time windows and multiple depots), but for a broad review of the problem, see Parragh et al. (2008).

2.2.2 The Dial a Ride Problem (DARP)

The Dial a Ride Problem is closely related to the PDP and generally involves the movement of people instead of freight. This means the quality of service offered must be of a higher standard, which usually means tighter time windows and/or a maximum time limit between pick up and drop off. For a recent survey, see Cordeau and Laporte (2007).

2.2.3 The Time Window Assignment VRP (TWAVRP)

In the Time Window Assignment Vehicle Routing Problem (Spliet and Gabor, 2014), customers have time windows of a parameterized length, but the position of these windows is set by the delivery firm. Additionally, each customer's demand is stochastic and is not revealed until after time windows are set. The aim is to construct routes that minimize the expected delivery cost (including some penalty for unsatisfied demand) across all realizations, and that also satisfy the agreed upon time windows.

2.2.4 The Consistent VRP (ConVRP)

The Consistent Vehicle Routing Problem (Groer et al., 2009), involves a fleet of vehicles delivering parcels to customers over multiple days (no customer gets more than one delivery a day, but some customers get deliveries on multiple days). A customer must always be served by the same driver and the variation between the arrival times (on different days) must be less than some parameterized limit. Within these constraints, the objective is to minimize the overall routing cost.

2.2.5 The Fleet Size and Mix Vehicle Routing Problem (FSMVRP)

The Fleet Size and Mix Vehicle Routing Problem (Golden et al., 1984), recognizes that real world delivery fleets involve heterogeneous vehicles with different characteristics (e.g., capacity, driving speed) and cost structures. Typically, this includes fixed costs of ownership (e.g., depreciation, registration) and operational costs that vary with use (e.g., fuel, maintenance). Given a set of customers and a time frame, the task is to determine the mix of vehicles that feasibly serves all customers and minimizes the total cost over the prescribed horizon.

2.3 Solution Methods

Given the amount of research into VRPs, the sheer number of solution methods employed is not surprising. Here we detail three common approaches which are utilized later in the thesis, but for a fuller review, see El-Sherbeny (2010).

2.3.1 Integer Programming

Integer Programming (IP) is a technique commonly used by the Operations Research (OR) community to model not only vehicle routing, but also resource assignment, planning, auction design and a host of other problems. As we use IP methods extensively throughout this thesis, we will give a brief introduction here, but a full review can be found in Wolsey and Nemhauser (2014). In a Mixed Integer Linear Program (MILP), practitioners formulate their chosen problem as a mathematical model using a mixture of continuous and integer *variables*. These variables are included in linear inequalities, called *constraints* (that must be satisfied), which restricts the set of values variables may take. We define an *objective function* as a linear expression of the variables, the evaluation of which is called the *objective value*. A *solution* is a vector of values (one per variable) such that all constraints are satisfied, and a solution is *optimal* if there exists no other solution with a smaller objective value.

The canonical form of a MILP is given below, where matrix $\mathbf{A}$ and vector $\mathbf{b}$ represent the known coefficients and constants associated with each constraint and the continuous and integer variables are denoted by vectors $\mathbf{x_c}$ and $\mathbf{x_i}$ respectively, with $\mathbf{x} = [\mathbf{x_c} \;\; \mathbf{x_i}]^T$.

Integer Program 1:

$$\begin{aligned} \text{minimize: } & \mathbf{c}^T\mathbf{x} \\ \text{subject to: } & \mathbf{A}\mathbf{x} \leq b \\ & \mathbf{x_c} \geq 0 \\ & \mathbf{x_i} \in \mathbb{K}^+ \end{aligned}$$

To solve an integer program, we typically use an approach referred to as *Branch and Bound*, where we first remove the integrality constraints to obtain a Linear Program (LP). This problem, sometimes referred to as the *linear relaxation*, is well known to be polynomially solvable and there exists a large number of efficient algorithms to solve these. We will assume a working knowledge of this area, but a full review can be found in Vanderbei (2014) or Hillier et al. (2015). For any LP, a vertex of the feasible region will have the minimal objective value – but the relevant variables may not be integer. We can impose new constraints that remove the optimal solution (to the relaxed problem) and partition the space into two disjoint regions to be explored independently. The integer solution inside either region with the most favorable objective value is optimal for the original integer program. As an example, consider the case given in Figure 2.2(a) and assume the objective is to maximize $z = x_1 + x_2$. The optimal solution to the linear relaxation is $x_1 = \frac{14}{3}, x_2 = \frac{16}{3}$ but this doesn't satisfy the integrality constraints, so we choose one of the two possible

branching strategies from Figure 2.2(b). We could impose the constraints outlined in green – this would form two sub–problems, which will both have linear relaxations with integer solutions (it turns out both solutions will have the optimal objective value, although this won't always be the case). Alternatively, if we had imposed the constraints given in red, then the resulting sub–problems are either infeasible or remain fractional.

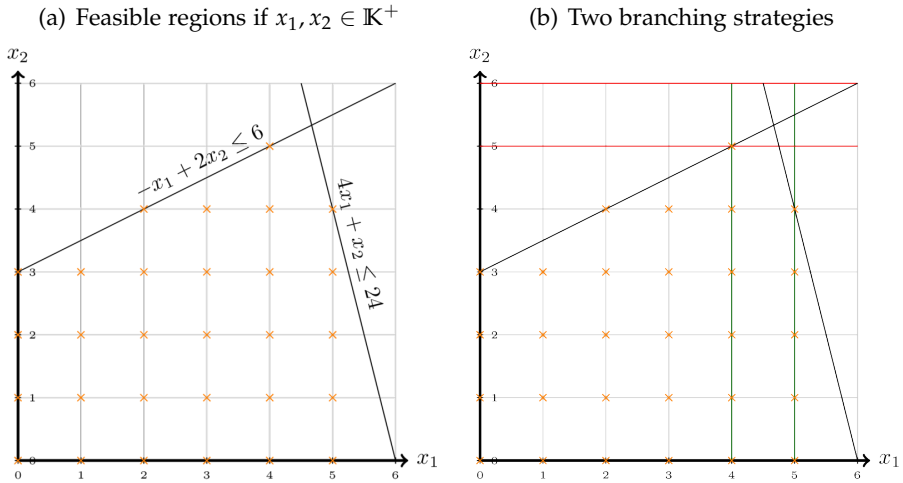


Figure 2.2: Different branching strategies

Conceptually, the process of solving an IP can be visualized as a *branch and bound tree*, with each node representing a single subproblem and child nodes representing subproblems formed by branching. Figure 2.3 gives the trees associated with the IP and branching strategies presented in Figure 2.2(b). Although not shown here, a node may also be excluded from further consideration (referred to as *bounded*), if it represents a fractional solution with a worse objective value than a known integer solution (as further branching will never improve the objective value).

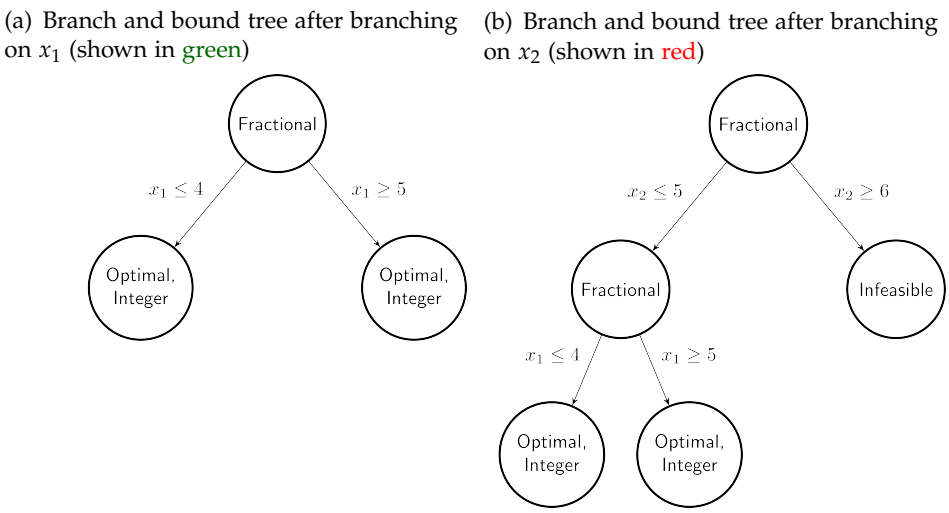


Figure 2.3: Example branch and bound trees

2.3.1.1 Formulation

We are now ready to give the standard IP formulation for VRP problems. The basic variant is presented here, with common extensions discussed in Section 2.3.1.2.

When presenting our formulation, we reuse the notation given previously in Section 2.1, with C and K denoting the sets of customers and vehicles respectively. The sole depot is represented by node n_0, and c_{ij} represents the cost of going from node i to node j. Our decision variables, x_{ij} $i, j \in \mathbf{N}$ are defined below:

$$x_{ij} = \begin{cases} 1 & : \text{If a vehicle travels between nodes } i \text{ and } j \\ 0 & : \text{Otherwise} \end{cases}. \tag{2.1}$$

Integer Program 2:

$$\min \sum_{i,j \in \mathbf{N}} c_{ij} x_{ij} \tag{2.2}$$

$$\text{s.t.}$$

$$\sum_{i \in \mathbf{N}} x_{ij} = 1, \; j \in C \tag{2.3}$$

$$\sum_{j \in \mathbf{N}} x_{ij} = 1, \; i \in C \tag{2.4}$$

$$\sum_{i \in C} x_{0i} = K \tag{2.5}$$

$$\sum_{i \in C} x_{i0} = K \tag{2.6}$$

$$\mathbf{x}_{ij} \in \{0, 1\} \tag{2.7}$$

The objective (2.2) aims to minimize the operational cost of routing the vehicles. Con-

straints (2.3) and (2.4) ensure that exactly one vehicle enters and leaves every node, while constraints (2.5) and (2.6) perform the equivalent check for the depot. Constraints 2.7 enforce the relevant domains for our decision variables.

Unfortunately, this actually is not a valid formulation as it permits solutions with closed loops (known as *subtours*) that do not include the depot (an example is given in Figure 2.4, where an illegal subtour contains customers 1, 2 and 3). The naive solution is to include constraints which prohibit such solutions, as given in (2.8) and (2.9).

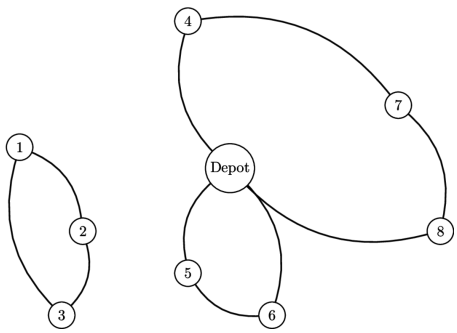


Figure 2.4: A subtour inside a VRP solution

$$\sum_{i \in S} \sum_{j \notin S} x_{ij} \geq 1, \ \forall S \subseteq C, |S| \geq 2 \tag{2.8}$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1, \ \forall S \subseteq C, |S| \geq 2 \tag{2.9}$$

Constraints (2.8) ensure that for all subsets of customers, there is at least one arc leaving that subset, preventing any closed loops that exclude the depot. (2.9) takes the converse approach by considering all subsets of customers and limiting the number of arcs between nodes to one less than the subset's cardinality, preventing the creation of loops. Note that just one of these constraint sets is sufficient to ensure feasibility.

Unfortunately, the number of constraints needed under this method is of order $\mathcal{O}(2^{|C|})$, meaning the task of enumerating the constraints is intractable for problems above a trivial size. However, most of these constraints will be inactive at the optimal solution, giving rise to *lazy constraint generation* methods. These initially solve the problem without the subtour elimination constraints and checks if subtours exist in the optimal solution. If found, the constraints required to remove the subtours are added; otherwise, we know the solution is optimal (and that the excluded constraints are inactive at this optimal solution).

2.3.1.2 Modeling Extensions

Now that we have given the basic VRP formulation, we will now present ways to model the common extensions. As in Section 2.1, this list is not intended to be exhaustive but

should act as an introduction for the more complex variants introduced in later sections.

The Capacitated VRP (CVRP):

There are two different ways in which we model the CVRP; the first (more complex) method also prevents the formation of subtours while the second (simpler) method does not.

The first method uses *Capacity Cut Constraints* (CCC), given in equation (2.10), where $r(S)$ returns the minimum number of vehicles needed to serve customers in S.

$$\sum_{i \notin S} \sum_{j \in S} x_{ij} \geq r(S), \ \forall S \subseteq C, S \neq \emptyset \tag{2.10}$$

Constraint (2.10) enforces capacity constraints by ensuring that the flow of vehicles into any subset of customers is sufficient to serve the demand of that set. This requirement for external flow naturally prevents the formation of subtours. Note that evaluating $r(S)$ requires solving a *Bin Packing Problem* (BPP), which although NP hard, can generally be solved quickly by sophisticated algorithms (or accurately estimated with quality heuristics). For more detail, we refer the reader to Martello and Toth (1990).

The second method is simpler and only involves a linear number of constraints (with respect to the number of customers). It involves tracking the level of stock delivered by vehicles as they progress through the network; specifically, let q_i be the total quantity delivered by a vehicle after departing node i, $i \in \mathbf{N}$. These are enforced in constraints (2.11)–(2.13), where d_i and Q again represent the demand of customer $i \in C$ and the capacity of each (homogeneous) vehicle respectively.

$$q_i = d_i + \sum_{j \in \mathbf{N}} q_j x_{ji}, \ i \in C \tag{2.11}$$

$$q_i \leq Q, \ i \in C \tag{2.12}$$

$$q_0 = 0 \tag{2.13}$$

Constraints (2.11) sets the quantity delivered as the q_j variable of the previous node, plus the quantity demanded by the current customer. Constraints (2.12) ensures the quantity delivered never exceeds the vehicle's capacity and constraint (2.13) sets the decision variable q_0 for the depot. Note that although presented as an equality, constraints (2.11) can be expressed by an inequality, as given in $(2.11')$. Although this change permits solutions where the q_i variables overstate the quantities delivered, constraint (2.12) ensures vehicle capacity is still respected, and the optimizer will drive the q_i variables to their lower bound if required for feasibility. This alternative formulation allows our IP model to have slightly fewer constraints, possibly resulting in a faster solve time. This can also be linearized, as given in $(2.11'')$. In this case, $\hat{M}$ can be any sufficiently large number, but it is well known that setting $\hat{M} = Q$ is sufficient.

$$q_i \geq d_i + \sum_{j \in \mathbf{N}} q_j x_{ij},\ i \in C \tag{2.11'}$$

$$q_i \geq d_i + q_j - \hat{M}(1 - x_{ij}),\ i \in C,\ j \in \mathbf{N} \tag{2.11''}$$

The Heterogeneous Fleet VRP (HFVRP):

Next, we consider a service provider who has a fleet of heterogeneous vehicles comprised of m_n different vehicle classes, represented by the set $M = \{1, ..., m_n\}$. For each class $m \in M$, the travel cost between any two locations is given by $c_{ij}^m,\ i, j \in \mathbf{N}$. and there are n_m vehicles are available at the depot. As each vehicle type has different characteristics, we must index the arc choice decision variables by the vehicle type, as given in $(2.1')$.

$$x_{ij}^m = \begin{cases} 1: & \text{If a vehicle of type } m \text{ travels between nodes } i \text{ and } j \\ 0: & \text{Otherwise} \end{cases} \tag{2.1'}$$

Next, we must extend our objective function to account for the different vehicle types, as given in $(2.2')$. Similarly, we should extend our flow conservation constraints to ensure a customer is still only visited once (by any vehicle type), as given in $(2.3')$ and $(2.4')$. Similarly, constraints $(2.5')$ and $(2.6')$ ensure the correct number of each vehicle type is used.

$$\min \sum_{m \in M} \sum_{i,j \in \mathbf{N}} c_{ij}^m x_{ij}^m \tag{2.2'}$$

$$\sum_{m \in M} \sum_{i \in \mathbf{N}} x_{ij}^m = 1,\ j \in C \tag{2.3'}$$

$$\sum_{m \in M} \sum_{j \in \mathbf{N}} x_{ij}^m = 1,\ i \in C \tag{2.4'}$$

$$\sum_{i \in C} x_{0i}^m = n_m,\ m \in M \tag{2.5'}$$

$$\sum_{i \in C} x_{i0}^m = n_m,\ m \in M \tag{2.6'}$$

The VRP with Time Windows (VRPTW):

Another common variant is the VRPTW, where we impose time windows in which deliveries to customers must occur, given by $[e_i, l_i], i \in C$. To enforce these windows, we introduce new decision variables $T_i, i \in \mathbf{N}$ which gives the vehicle's arrival time at location i, and parameters t_{ij}, which give the travel time between locations i and j. We enforce time windows using constraints $(2.14-2.17)$.

$$T_i \geq \sum_{j \in \mathbf{N}} (T_j + t_{ji}) x_{ji}, \; i \in C \tag{2.14}$$

$$T_i \geq e_i, \; i \in C \tag{2.15}$$

$$T_i \leq l_i, \; i \in C \tag{2.16}$$

$$T_0 = 0 \tag{2.17}$$

Constraints (2.14) set the arrival time variables as the departure time at the previous node, plus the travel time between the two locations. The inequality can introduce wait time in to the schedule (possibly to respect time windows), but this can be minimized in a post processing phase. Constraints (2.15)–(2.16) enforce the relevant bounds and constraint (2.17) sets the departure time at the depot.

Finally, constraints (2.14) need to be linearized before inclusion in our model, as given below:

$$T_i \geq T_j + t_{ij} - M(1 - x_{ij}), \; i, j \in \mathbf{N} \tag{2.14'}$$

As smaller values of M generally introduce less fractionality and allow reduced solve times, we give a simple upper bound on the value of M in (2.18). Specifically, we find the latest time a vehicle could return to the depot after serving each customer and take the latest time over all customers.

$$M = \max_{i \in C} \{l_i + t_{i0}\} \tag{2.18}$$

The Multiple Depot VRP (MDVRP):

Many scenarios involve a set of n_d depots, given by $D = \{d_1, d_2, \ldots, d_n\}$ and each depot $d \in D$ has its own fleet of vehicles, K_d. To model this, we simply have a node for each depot and restate the relevant flow constraints, as given in $(2.5'')$ and $(2.6'')$.

$$\sum_{i \in C} x_{di} = K_d, \; d \in D \tag{2.5''}$$

$$\sum_{i \in C} x_{id} = K_d, \; d \in D \tag{2.6''}$$

Selective VRPs (SVRP):

In all variants of the SVRP, only a subset of customers need to be served and this requires modified flow conservation constraints. Specifically, the inequality in constraint $(2.3'')$

allows up to one vehicle to visit each customer, while constraint $(2.4'')$ ensures vehicles leave customers they visit.

$$\sum_{i \in \mathbf{N}} x_{ij} \leq 1, j \in C \tag{2.3''}$$

$$\sum_{j \in \mathbf{N}} x_{ij} = \sum_{j \in \mathbf{N}} x_{ji}, i \in C \tag{2.4''}$$

In the variant known as the Orienteering Problem (OP), one of the relevant resources is limited and the amount consumed across all vehicles must be below some limit L. Specifically, if $r_{ij},\ i, j \in \mathbf{N}$ is the amount consumed by traveling between nodes i and j, then constraint (2.19) must be respected. The objective is to maximize the prizes earned by visiting customers, $p_i,\ i \in C$, as given in (2.20).

$$\sum_{i \in \mathbf{N}} \sum_{j \in \mathbf{N}} r_{ij} x_{ij} \leq L \tag{2.19}$$

$$\max \sum_{i \in \mathbf{N}} p_i \sum_{j \in \mathbf{N}} x_{ij} \tag{2.20}$$

In the Prize Collecting Vehicle Routing Problem (PCVRP) we must collect a minimum value of prizes, P, enforced by constraint $(2.19')$. The standard VRP objective from (2.2) is typically used, with the cost coefficients measuring consumption of the limited resource, e.g., $c_{ij} = r_{ij},\ i, j \in \mathbf{N}$

$$\sum_{i \in \mathbf{N}} p_i \sum_{j \in \mathbf{N}} x_{ij} \geq P \tag{2.19'}$$

In the Vehicle Routing Problem with Profits (VRPP), the objective is to maximize the difference between the prizes obtained and the cost incurred, as given in (2.21).

$$\max \sum_{i \in \mathbf{N}} p_i \sum_{j \in \mathbf{N}} x_{ij} - \sum_{i \in \mathbf{N}} \sum_{j \in \mathbf{N}} c_{ij} x_{ij} \tag{2.21}$$

The Dynamic VRP (DVRP):

In the DVRP, information is progressively revealed over time. There are two main solution approaches:

- **Rolling horizon:** This involves solving a standard VRP considering currently available information, and then re–solving an updated version at regular time intervals, or whenever new information becomes available. This allows standard algorithms

to be used, but such methods only respond to (instead of preempting) the newly revealed information.

- **Sampling methods:** These approaches attempt to anticipate the uncertainty by generating a series of scenarios (representing different realizations of the unknown information) and then solving a standard VRP for each scenario. Alternatively, we can generate a series of solutions to the original problem and measure their performance across different scenarios. Information is then somehow aggregated to generate a solution before the hidden information is revealed. Naturally, the accuracy of these methods rely on their ability to generate representative scenarios.

The Stochastic VRP (SVRP):

Like the DVRP, the SVRP involves hidden information, but this is only revealed after all decisions have been made (preventing *rolling horizon* solution approaches). There are many variants of this problem and for a more thorough review, see Klein Haneveld and van der Vlerk (1999), Powell and Topaloglu (2003), Shapiro and Philpott (2007) or Bianchi et al. (2009). We will focus on the simplest case, where different realizations of an unknown aspect are represented by a series of scenarios $s \in S$, which each occur with probability p_s. Depending on the form the uncertainty takes, each scenario will have a slightly different integer program, with altered coefficients in the objective function and/or the constraint matrix, or a different vector of right hand side constants. To extend the notation given in Integer Program 1, each scenario has a vector of objective coefficients c_s, a constraint matrix A_s and a right hand side vector b_s. If our decision variables are still represented by the vector $\mathbf{x}$ (comprising of two sub–vectors, $\mathbf{x_c}$ and $\mathbf{x_i}$, representing continuous and integer variables respectively), then we can define a larger integer program, given below:

Integer Program 3:

$$\begin{aligned} \text{minimize: } & \sum_{s \in S} p_s \mathbf{c_s}^T \mathbf{x} \\ \text{subject to: } & A_s \mathbf{x} \leq b_s, \ s \in S \\ & \mathbf{x_c} \geq 0 \\ & \mathbf{x_i} \in \mathbb{K}^+ \end{aligned}$$

The objective function aims to minimize the *expected* cost, while the series of constraints ensure feasibility under all scenarios. If some constraints are duplicated exactly across multiple scenarios, then we can remove all copies after the first to improve computational efficiency.

The Periodic VRP (PVRP):

In the PVRP, the planning horizon spans T time periods and each customer must be visited k times (with no more than a single visit per time period). Furthermore, these visits may be restricted to certain combinations of time periods. For example, if we set $T = 5$ (with each period representing a day in a working week) and set $k = 2$, customers might only accept visits on the set of days {Monday, Wednesday}, {Tuesday, Thursday} or {Wednesday, Friday}. The goal is to design a schedule for each time period such that all

customers are visited the appropriate number of times during a feasible set of periods and the overall cost is minimized.

2.3.1.3 Branch and Price

Although the IP formulations given above are relatively simple and widely applicable, their linear relaxations are very weak and require large branch and bound trees, limiting their use to small instances. However, there is a related IP approach called *branch and price*, that uses a different model with a stronger linear relaxation. We will briefly outline the approach here, but for a full review, we refer the reader to Feillet (2010). Conceptually, this method builds a pool that (implicitly) contains every feasible vehicle route and selects a subset (one for each vehicle) such that every customer is visited once and the total cost is minimized. More formally, let Ω be the set of routes, where route $r \in \Omega$ has an associated cost c_r. Let $a_r,\ r \in \Omega$ be a vector with cardinality $|C|$, where the ith element, given by $a_{ir},\ i \in C$ is a binary indicator showing if customer i is served by route r. We also define a binary variable d_{ijr} which indicates if arc $(i,j) \in \mathbf{A}$ is used in route $r \in \Omega$. Finally, let $\theta_r,\ r \in \Omega$ be the number of times r is used by the final solution. The resulting IP, referred to as the *Master Problem* or $MP(\Omega)$, is given below:

Integer Program 4:

$$\min \sum_{r \in \Omega} c_r \theta_r \tag{2.22}$$

$$\text{s.t.}$$

$$\sum_{r \in \Omega} a_{ir} \theta_r \geq 1, j \in C \tag{2.23}$$

$$\sum_{r \in \Omega} \theta_r = K \tag{2.24}$$

$$\theta_r \in \mathbb{N} \tag{2.25}$$

$$r \in \Omega \tag{2.26}$$

(2.22) is the objective and minimizes the total cost of selected routes. Constraints (2.23) ensure all customers are served and constraint (2.24) selects a single route for each vehicle. Finally, constraints (2.25) enforce integrality. Note that although the θ_r variables allow a customer to be visited multiple times, doing so incurs an additional cost, so the objective function will prevent this from occurring in the optimal solution.

Unfortunately, the set Ω grows exponentially with the number of customers, so simply enumerating all routes is impractical for any reasonably sized problem. Consequently, the Master Problem contains only a subset of all possible routes. This reduced set is denoted by Ω' and the smaller problem is called the *Restricted Master Problem*, or the RMP. As the RMP is clearly dependent on the routes considered, it is often expressed as $RMP(\Omega')$. Given an optimal solution to the $RMP(\Omega')$, it is possible to generate the routes $r \in \Omega \setminus \Omega'$ that are needed to further improve the solution. We use what is known as a *Pricing Problem* (PP), which for VRP applications generally takes the form of an *Elementary Shortest Path Problem with Resource Constraints* (ESPPRC). This contains the same nodes ands arcs as

the original VRP, but with different arc lengths. Specifically, the lengths are a function of the original objective coefficients and the dual variable of the constraint from (2.23) associated with the customer at the head of the arc. In some sense, this variable estimates how efficiently that customer is served. It can be shown that a negative length cycle (using the new arc lengths) that includes the depot represents a route that, if added to Ω', will improve the current solution. There may be additional constraints on the PP, such as those relating to vehicular capacity and time windows. The absence of a negative length cycle implies the optimal solution to the $RMP(\Omega')$ is also optimal for the $MP(\Omega)$.

The algorithm iterates between the Restricted Master and Pricing Problems (with each providing new information for the other), until the PP fails to find an improving route (and optimality for the RMP is achieved). However, there is nothing to guarantee intergrality for the θ_r, $r \in \Omega'$ variables and branching may be required (similar to Section 2.3.1). The naive strategy would choose a single (fractional) θ_r variable and derive two branches where the variable is set to 0 and 1 respectively. However, this is an inefficient strategy, primarily because it leads to a highly unbalanced search tree; while the first branch has almost no impact on the solution space, the second has a very strong impact. Additionally, the θ_r variables are in some sense artificial and implicitly store less information about the local routing decisions being made. Finally, enforcing $\theta_r = 0$ requires us to forbid discovery of route r in the PP, which can significantly increase solve times (again, for a fuller discussion of these matters, see Feillet (2010)).

A better strategy is to identify an arc that carries factional flow and branch on the corresponding decision variable from the original problem. To do this, we select an arc $(i, j) \in \mathbf{A}$ such that $0 < f_{ij} = \sum_{r \in \Omega'} d_{ijr}\theta_r < 1$, and derive two branches; one where $f_{ij} = 0$ (forbidding the use of arc (i, j)) and one where $f_{ij} = 1$ (requiring the use of arc (i, j)). To enforce the first branch, we simply remove all columns from Ω' which use arc (i, j), and remove the arc from the network in the PP. To enforce the second, we note that as every customer must be visited exactly once, we can equivalently forbid the use of arcs (i, h), $h \neq j$ and $(h, j), h \neq i$. Again, this is done by removing the routes from Ω' which use these arcs and modifying the network used in the PP. These branching rules lead to a (more) balanced search tree, utilize local information inherent in the original variables and do not complicate the PP. Of course, if a subproblem resulting from a branch is infeasible, or has an optimal objective value worse than an existing integer solution, then it can be excluded from further consideration.

2.3.2 Constraint Programming

Another very popular method for generating exact solutions is *Constraint Programming* (CP). Broadly speaking, this approach works by defining a feasible set of values for each variable and reducing it by using logical reasoning on the interaction between constraints, e.g., time windows may imply two customers must be visited in a certain order. This process is called *constraint propagation*, and is computationally easy to do. If this does not return a solution, then we use a *search* algorithm, where we select a value for a certain variable and apply constraint propagation to the updated domains. To guarantee optimality, this process must be repeated for all values the chosen variable can take. For a broad re-

view of CP, we refer the reader to Rossi et al. (2006) and for CP techniques relating specifically to the VRP, see Kilby and Shaw (2006).

In a general sense, Constraint Programming is similar to Integer Programming; both are exact methods that implicitly search the whole solution space, using a search tree to branch on variable domains. However, they have differing strengths – CP approaches typically find feasible solutions (comparatively) quickly, but the approach inherently puts less focus on improving the objective value (while the reverse can be said about IP algorithms). Interestingly, the modeling techniques used with Constraint Programming tend to be more expressive and allow a wider range of constraints (Kilby et al., 2000). It is worth noting that CP techniques can also be used within heuristics, as discussed in Section 2.3.4.6. In this setting, the introduction of non–standard constraints requires little adaptation of the solution method (Kilby et al., 2000); in contrast, important routines in other heuristics (like the one that checks if a new solution is feasible) often make assumptions about the constraints present and would require rewriting. There is no general consensus as to if CP or IP approaches are better, as the relative merits depend on the setting under consideration.

2.3.3 Heuristics

As discussed above, the complexity of the VRP means finding optimal solutions is computationally intractable, or at least sufficiently difficult to render the process uneconomic. For this reason, many researchers make use of *heuristics* – "methods which on the basis of experience or judgment seems likely to yield a good solution to the problem, but cannot be guaranteed to produce an optimum" (Foulds, 1983). In this setting, we review common classes of heuristics and present the state of the art in relation to VRPs; for a deeper survey, we refer the reader to Vidal et al. (2013a).

2.3.3.1 Construction Heuristics

Not surprisingly, *construction heuristics* build a complete solution from an empty one; in VRP settings, this is typically through the considered insertion of customers into partial routes. There are many natural ways in which this can be done and we detail some of the common methods below.

Savings Algorithm (Clarke and Wright, 1964)

One of the earliest approaches is Clarke's and Wright's savings heuristic. This starts by assuming each customer is served by its own vehicle and calculates the savings made by consolidating routes. An example is given in Figure 2.5, where the savings from serving both customers with a single vehicle are given by $c_{10} + c_{02} - c_{12}$. We repeat this calculation for every pair of customers and rank the savings in decreasing order. We then descend the list, pairing customers were possible, without deleting a previously established direct connection between two customers, or exceeding the vehicle's capacity.

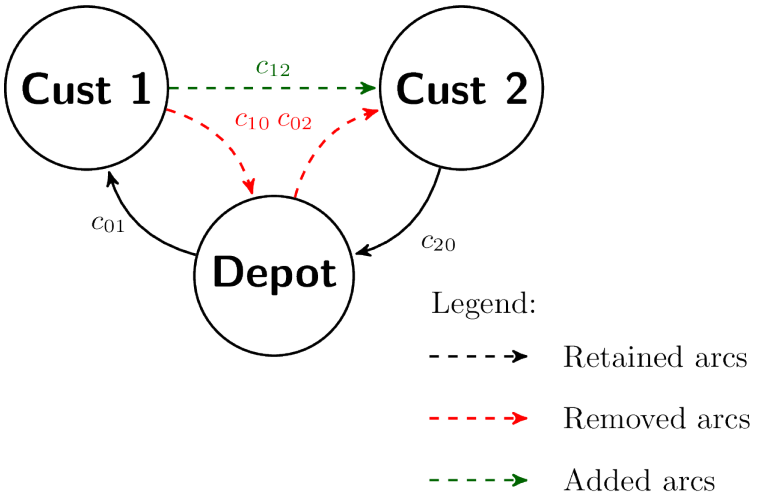


Figure 2.5: Illustration of Clarke and Wright's savings algorithm

Nearest Neighbor Insertion (NNI) (M. Bellmore, 1968)

Nearest Neighbor Insertion algorithms start with an empty schedule and insert the unserved customer that increases the cost by the smallest amount (referred to as the *insertion cost*). Again, this could be done sequentially for each vehicle, or for all vehicles in parallel.

Sweep Algorithm (Wren and Holliday, 1972)

The Sweep Algorithm first allocates customers into clusters and then builds a route involving each group. More specifically, to build a new route k, we start by adding a random customer $i \in C$ to k and create a line between the depot and the customer, denoted by L_{0i}. We then find customer $j \in C$ such that the angle between L_{0i} and L_{0j} is minimized and add customer j to route k. This process is repeated until the vehicle's capacity is reached. Depending on the setting, routes can be built sequentially or in parallel. Once a route is finished, we then solve a Traveling Salesman Problem (either heuristically or exactly) involving the selected customers to determine the final schedule.

k–Regret Heuristics (Potvin and Rousseau, 1993)

Regret algorithms still build up a schedule through the sequential insertion of customers, but attempt to estimate the insertion cost in a more forward looking manner. In the basic 2–regret variant, for each customer $i \in C$, we calculate the difference in the insertion cost between the cheapest and second cheapest positions, given by r_2^i (note that these positions must be in different schedules). We then select the customer with the biggest difference, e.g., $\max_{i \in C}\{r_2^i\}$. This prioritizes the inclusion of customers who may be difficult to insert later and delays the addition of those who can easily be served by multiple vehicles. This can be generalized to a k–regret variant where for each customer $i \in C$ we find the difference in insertion cost between the cheapest and all subsequent positions (in different vehicles) and take the sum of these values, e.g., $\max_{i \in C}\{\sum_{j=2}^{k} r_j^i\}$ (Pisinger and Ropke, 2007).

2.3.3.2 Improvement Heuristics

Improvement heuristics take an existing solution and attempt to improve it by repeatedly applying a simple alteration or *move*, e.g., changing the position of a customer in their route. The generality and versatility of these approaches means there are a large number of different methods; a full review can be found in Laporte et al. (2000) and some of the common variants are detailed below.

- **Relocate:** A simple and obvious heuristic is simply to move a single customer into a new position, as shown in Figures 2.6(b) and 2.6(c).
- **Swap:** We simply swap the position of two customers, as shown in Figures 2.6(d) and 2.6(e).
- **Two–opt** We remove two arcs and connect the two former head (tail) nodes, reversing intermediate arcs as required (see Figures 2.7(b) and 2.7(c)).
- **Or–opt:** We remove a string of consecutive customers and insert them into a new position, either in the same or different route. To further add diversity, the order of the visits could be reversed (note that this is a generalization of the **Relocate** move).
- **Three-point move:** We swap a consecutive pair of customers with a single customer. This is a generalization of the **Swap** neighborhood and can be further generalized.
- **Tail exchange:** We simply swap the last part of one route with the last part of another. Again, this is a generalization of the **Swap** neighborhood.
- **Three-opt move:** We remove three arcs and reconnect the network in all possible ways, reversing intermediate arcs if necessary. Again, this can be generalized to a k–opt move, where k is bounded by the schedule size.

In some cases, the number of new solutions reachable within one move is very high and evaluating all possible realizations is computationally difficult. There are several ways to improve efficiency:

- **Pre–processing/forward checking:** By performing some pre–processing between accepted moves, we can check the feasibility of a new move in constant time (for certain heuristics); for more detail, see Savelsbergh (1992).

- **Check change in objective value:** For other heuristics, checking if a neighboring solution has an improved objective value is faster than checking for feasibility.

- **Granular neighborhoods:** Often, there are many moves we could reasonably expect to be infeasible or to worsen the objective value, e.g., moves that shift customers with tight time windows several places along a schedule, or moves that introduce very expensive arcs. Excluding these focuses exploration in promising areas of the solution space; for the original paper on this concept, see Toth and Vigo (2003).

Figure 2.6(a): Original schedules

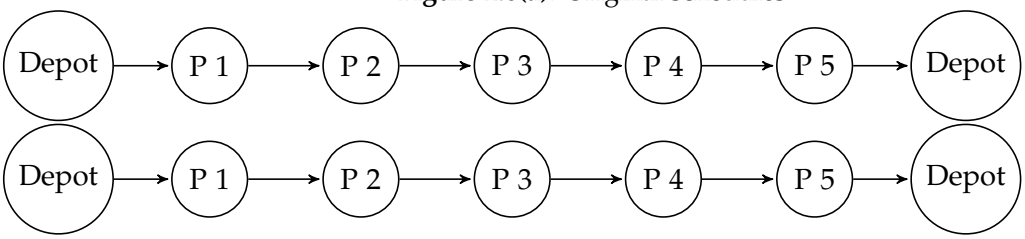


Figure 2.6(b): Relocating a customer within a schedule

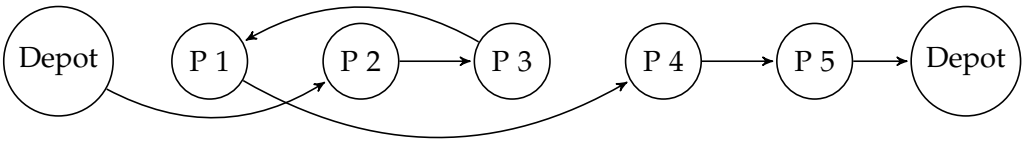


Figure 2.6(c): Relocating a customer between schedules

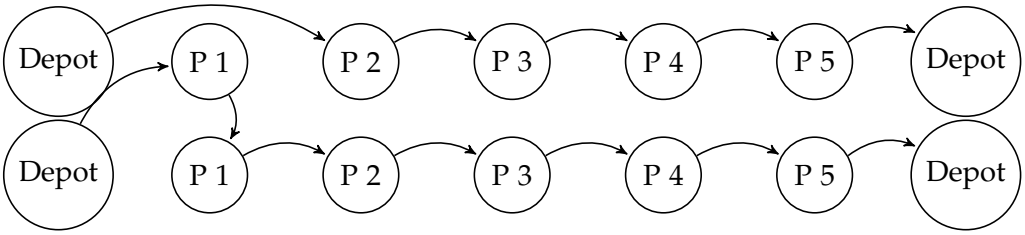


Figure 2.6(d): Swapping customers within a schedule

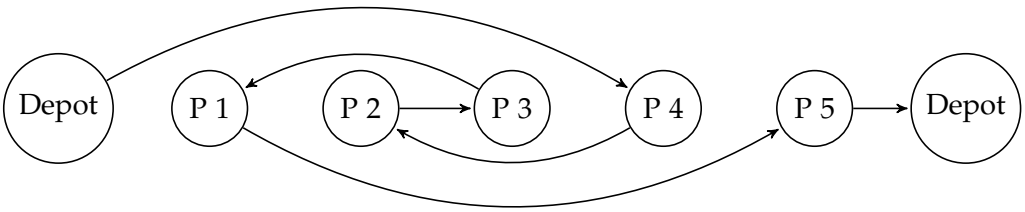


Figure 2.6(e): Swapping customers between schedules

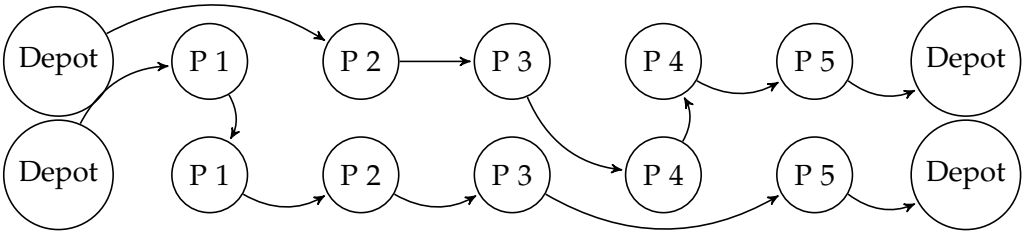


Figure 2.6: Relocate and Swap moves

Figure 2.7(a): Original schedules

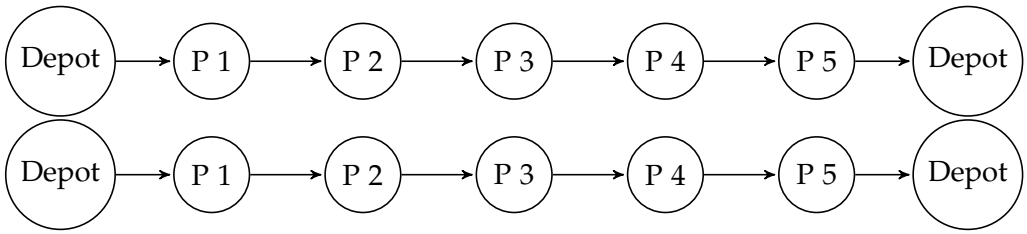


Figure 2.7(b): 2–opt within a schedule

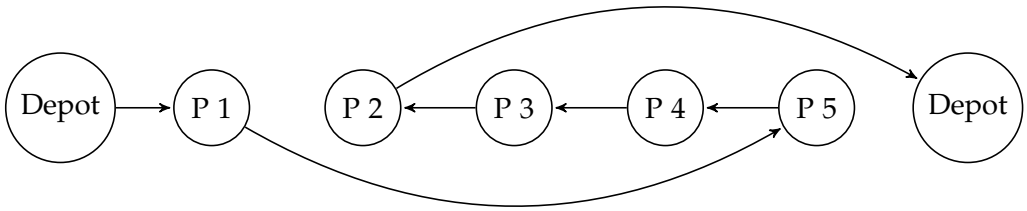


Figure 2.7(c): 2–opt between schedules

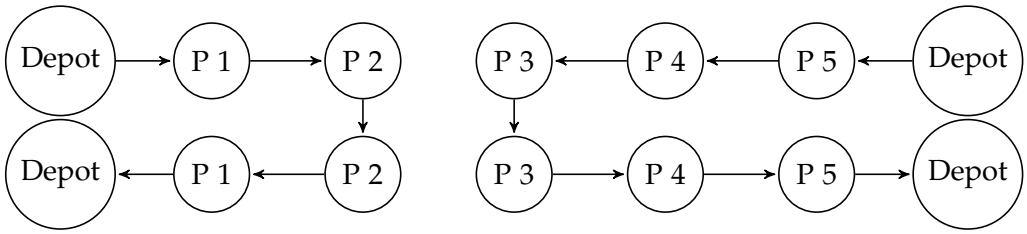


Figure 2.7: 2–opt moves

2.3.4 Meta–Heuristics

Like heuristics, meta–heuristics are processes designed to quickly find good solutions, but have some way of escaping from local optima and continuing the search. Although varied, meta–heuristics typically share four common phases:

- **Initialization:** All meta–heuristics need to generate an initial solution, possibly using an approach from Section 2.3.3.1.

- **Intensification:** At certain points in the meta–heuristic, we might decide the current solution is of a high quality and wish to explore the surrounding solution space thoroughly. This could be done through the repeated application of the improvement routines from Section 2.3.3.2, or through specific operators outlined below.

- **Diversification:** At other points in the process, we might want to leave the current region of the solution space and explore elsewhere (generally because we have found the local optimum). This often involves either a *shaking* step or *hill climbing*, though other methods are possible. The former uses "ruin and recreate" heuristics to destroy and rebuild a solution (representing a sudden jump through solution space). *Hill climbing* may involve the application of the improvement heuristics given above, where moves that worsen the objective value can be accepted (representing a climb away from the local optimum).

- **Termination:** Meta–heuristics typically have some end condition; this could be exceeding some fixed duration (measured in time or a maximum number of iterations) or until the best known solution has not been recently improved (within a certain length of time or number of iterations).

Meta–heuristics can be parameterized through numerical constants that affect search patterns, e.g., the chance of accepting a worse solution in Simulated Annealing, or the number of forbidden features in Tabu Search (these concepts are discussed in detail below). These should be set through computational experiments, as described in Coy et al. (2001) and Barbosa et al. (2015).

In response to the growing availability of cheap computing power, some practitioners have developed mechanisms to split the search across multiple, parallel processors. The exact approach depends on the meta–heuristic under consideration, but usually involves running multiple instances of the algorithm (with different parameters and search settings) simultaneously. When certain criteria are met, information and solutions are synthesized and transferred across the instances. This approach attempts to encourage rapid exploration of the solution space, while ensuring the best information is collected and utilized.

We now detail the most common meta–heuristics and give details on the current state of the art as it applies to vehicle routing problems. For a wider review, we refer the interested reader to Boussaid et al. (2013).

2.3.4.1 Simulated Annealing (SA)

The *Simulated Annealing* meta–heuristic, given in Kirkpatrick et al. (1983), is inspired by the process of annealing from the field metallurgy, which involves the heating and controlled cooling of a metal. Notably, rapid cooling allows material to be processed faster, but changes the strength and condition of the final product. Similarly, the algorithm has an internal variable tracking the simulated temperature; the rate at which this falls affects the algorithm's run time and the properties of the final solution.

The process begins by constructing some initial solution and setting the simulated temperature to a high value. Each iteration involves a single move from a randomly chosen improvement heuristic, and any move that improves the objective value is automatically accepted. Worsening moves are accepted probabilistically, depending on the size of the difference in objective values and the current temperature, with small deteriorations and higher temperatures giving a greater chance of acceptance (for details of the criterion used, see Metropolis et al. (1953)). Between iterations, the temperature is systematically reduced. At first the algorithm is highly explorative, with high temperatures allowing the search to travel far from the initial solution in a semi–guided manner. Once the temperature starts to fall, the algorithm intensifies the search around the current solution, looking for small, local improvements. When the temperature falls below some critical threshold, the process either restarts from a new initial solution or terminates (and returns the best solution found). For a recent review, see Dowsland and Thompson (2012).

One of the earliest uses of Simulated Annealing on routing problems is Golden and Skiscim (1986), where it was applied to TSP instances and compared to contemporary methods in extensive computational tests. Alfa et al. (1991) is the first to apply SA to the VRP, using a 3–opt neighborhood. Teodorović and Pavković (1992) extend this work with the consideration of stochastic demand. Graffigne (1992) and Azencott (1992) introduce Parallel Simulated Annealing which is later extended in Czech and Czarnas (2002). Breedam (1995) compare the performance of various SA approaches with competing meta–heuristics, noting that the former found high quality solutions but required very long run times. However, Chiang and Russell (1996) compare their SA algorithm with competing methods on the VRPTW and reports favorable results for both solution quality and run time. Lin et al. (2006) give an SA approach for the CVRP that, at the time, was one of the best approaches for this problem. Harmanani et al. (2011) investigate a typical CVRP, but their neighborhood moves incorporate a large amount of randomness, which they claim helps for the specific instances under consideration. SA approaches have also been applied to multi–objective VRP variants; Banos et al. (2013) investigate an MOVRPTW where both the total distance and the difference in workloads between vehicles must be minimized. Tavakkoli-Moghaddam et al. (2011) also study the MOVRPTW, where they aim to minimize the routing cost while maximizing the volume of goods sold. Interestingly, the latter is dependent on how quickly the customer is serviced (visiting too late allows a competitor to steal the business).

2.3.4.2 Tabu Search (TS)

Tabu Search, described in Glover (1989) and Glover (1990), essentially combines improvement routines with dynamic rules preventing the acceptance of certain solutions. Like

most meta–heuristics, TS repeatedly applies neighborhood search to the current solution until some stopping criterion is satisfied. To encourage the exploration of the broader solution space, the algorithm maintains a list of features seen in recent solutions, and disallows the acceptance of new solutions which exhibit these features for some period of time. The definition of these features depends on the approach, but often refers to a set of customers affected by a specific neighborhood routine, i.e., customers moved between schedules by an **Or–opt** routine cannot be shifted again for a certain number of iterations. This is particularly useful after a *hill climbing* phase, to forbid the reversal of moves that worsen the objective function. The list of banned features is called the *Tabu List* and the length of this list is referred to as the *Tabu Tenure*. Longer lists encourage the exploration of the wider solution space, while shorter lists allow an intensive search around the current solution. To understand the etymology of the name, we note that the word *Tabu* refers to a Polynesian concept indicating something is sacred and should be avoided. For a recent review of these methods across a range of problems, we refer the reader to Glover and Laguna (2013).

One of the first to apply Tabu Search to the VRP, Gendreau et al. (1994) introduce two mechanisms that have become widespread. First, they record the specific neighborhood move used to create each solution, and penalize those formed by the relocation of frequently moved customers. Secondly, various numerical parameters (which control key algorithmic decisions) are dynamically adjusted based on recent performance. Crainic et al. (1993) and Garcia et al. (1994) introduce parallelism in TS, and are later extended by Badeau et al. (1997); Cordeau and Maischberger (2012) and Banos et al. (2013). Renaud et al. (1996) build on previous approaches for the MDVRP. Taillard et al. (1997) apply TS to a VRP with soft time windows, and Gendreau et al. (1999) extend this to include dynamic customers. Cordeau et al. (2001) present a general Tabu Search for variants with a range of temporal and depot specific constraints. The concept of Granular Tabu Search, introduced in Toth and Vigo (2003) and extended in Escobar et al. (2014), proves to be very effective and enables the discovery of many new best known solutions for standard test instances. Gendreau et al. (2008) investigate a CVRP with two–dimensional loading constraints. Moccia et al. (2012) (formally) introduce the idea of *Incremental* Tabu Search, which stores computed information regarding attempted insertions and deletions for routes left unaltered between iterations. Brandao (2011) apply TS to a VRP with a heterogeneous fleet, introducing novel neighborhood granularization techniques and a new deterministic shaking method. Taken together, these advancements allow the discovery of a small number of new best known solutions. Finally, Paquette et al. (2013) apply TS to a multi–objective DARP, balancing operational costs with customer service objectives, using real–world data from a major Canadian transporter provider.

2.3.4.3 Variable Neighborhood Search (VNS)

Variable Neighborhood Search, first given in Mladenović and Hansen (1997), is based around systematically exploring a series of nested neighborhoods, each one larger than the previous and finding the minima with respect to each in turn. Every time an improving move is found, the search resumes from the innermost neighborhood, before again moving outwards. If none of the neighborhoods offer an improvement, we either apply a shaking step or restart from a new solution. The inner neighborhoods are generally fast to

explore, with the larger, outer neighborhoods only being used when necessary. This can be imagined as the heuristic starting from an initial solution and systematically exploring larger portions of the surrounding solution space, before restarting from a new location when the local minima is found. For a wide review of VNS approaches, we refer the reader to Hansen et al. (2010).

An early example of VNS applied to the VRP is given in Bräysy (2003). They were the first (within VNS) to dynamically adjust their search parameters based on recent performance and also used a modified objective function to escape local optima. In addition, they propose four local search procedures that later became popular. Together, these advancements helped produce a small number of new best known solutions. Kytöjoki et al. (2007) investigate very large scale VRPs (with up to 20,000 customers), which naturally introduces many computational and implementational difficulties. They reduce memory requirements by storing travel times and distances in a condensed form (though this requires recomputing every time a figure is requested). They also restrict their insertion heuristics by only attempting the insertion of certain customers and recording costs between iterations. Finally, they describe a new data structure for storing solutions which reduces the asymptotical complexity of inserting customers into solutions. Computational experiments show results near the best known in much shorter time frames than had previously been seen. Fleszar et al. (2009) investigates the standard VRP, developing neighborhoods centered around reversing and/or swapping segments between routes. Belhaiza et al. (2014) give a hybrid VNS–TS algorithm for the VRPMTW, presenting an interesting approach for minimizing the duration of a given route. Armas and Melian-Batista (2015) use VNS on an rich VRP with many additional complications, including heterogeneous vehicles, multiple and soft time windows, customers priorities and dynamic customers. They achieve positive computational results on test instances and significant cost savings after their algorithm was embedded into a Spanish fleet management system.

There has also been a lot of work applying VNS specifically to the Team Orienteering Problem (TOP) and its related variants. Sevkli and Sevilgen (2006) give a VNS approach for the TOP, investigating the relative impact of different neighborhood search routines and shaking procedures. Labadie et al. (2012) produce a highly granular VNS for the TOPTW. Specifically, they formulate a relaxation as an (easily solvable) assignment problem and use dual information to identify promising arcs. They achieve new best known results in around 20% of test instances. Divsalar et al. (2013) apply a VNS to a variant of the TOPTW where routes last several days and they minimize the sum of vehicle operating costs and driver accommodation costs. Tricoire et al. (2010) apply a VNS to a periodic OP with multiple time windows (motivated by a real world industrial transportation problem) and present efficient ways of checking the feasibility of routes.

2.3.4.4 Large Neighborhood Search (LNS)

As the name suggests, Large Neighborhood Search (first introduced in Shaw (1998)), makes use of a single, large neighborhood. The associated moves are quite complex and computationally slow (especially compared to VNS approaches), but can explore vast areas of the solution space. In VRP settings, these moves are "ruin and recreate" approaches, where multiple customers are removed and then reinserted according to some set of rules (using

Constraint Programming techniques to find changes in cost and feasibility). The likelihood of a specific rule being chosen may change dynamically based on past performance; this is typically referred to as *Adaptive* Large Neighborhood Search (ALNS). For a review of recent advances in LNS, see Pisinger and Ropke (2010).

In terms of past work, Bent and Van Hentenryck (2004a) present a two phase algorithm using SA and LNS to minimize the number of vehicles used and total distance traveled respectively. They improve the best known solution in 17% of their test instances and match the best known solution in the remainder. Ropke and Pisinger (2006a) introduce the idea of Adaptive Large Neighborhood Search (ALNS) on the PDPTW and improve the best known solution in about half of their instances. Ropke and Pisinger (2006b) extends this to more general variants and achieves even better results, finding new best known solutions in around two thirds of cases trialled. Pisinger and Ropke (2007) give an ALNS for a rich PDPTW, presenting very general neighborhoods that still enjoy wide use today. Prescott-Gagnon et al. (2009) give a LNS for the VRPTW that uses a heuristic column generation approach to generate new routes, improving best known solutions in around 50% of benchmark instances. Azi et al. (2010) give an ALNS for a VRPM involving trips over multiple days that uses neighborhoods operating on very different scales. Some affect a single customer, while others affect whole days or even collections of days, with the relative impact of each being shown through computational results. This is extended in Azi et al. (2012), where new customers appear dynamically and accept or reject decisions must be made quickly. Kim et al. (2013) investigate the relative effectiveness of common LNS neighborhoods, although they note their findings are limited to instances similar to those investigated. Salazar-Aguilar et al. (2014) apply an ALNS to a rich TOP and present computational results on a new suite of test instances.

2.3.4.5 Genetic Algorithm (GA)

Genetic Algorithms, introduced by Holland (1992), are meta–heuristics based around the ideas of evolution and natural selection. Instead of focusing on one solution at a time, GAs typically maintain a pool of solutions, referred to as a *population* of *individuals*. We define a metric called the *fitness function*, which gives the strength and quality of a solution; this is usually the objective function, although some authors propose different measures (Chakraborty, 2004; Goncalves, 2007). The individuals with the greatest fitness values are carried over to the next iteration (or *generation*). Solutions may progress unaltered (referred to as *asexual reproduction*), or could be combined with another solution (*crossover*). In either case, small random perturbations (*mutations*) to the solution may be made to escape local optima and explore new regions of the search space. Diversity may also be added through *immigration*, where new solutions (potentially from a parallel algorithm) are introduced to the population. Some approaches apply improvement routines to individual solutions; these are called *memetic algorithms*. For a recent survey of advances in Genetic Algorithms, see Anita and Rucha (2012).

For early examples of GAs applied to the VRP, see Thangiah et al. (1991), Blanton and Wainwright (1993), Thangiah and Gubbi (1993), Thangiah et al. (1993), Thangiah (1993), Thangiah (1995), Potvin and Bengio (1996), Potvin et al. (1996), Aggarwal et al. (1997), Baker and Ayechew (2003) and Prins (2004). From these, there are two advancements that

deserve special mention. The first is the idea of *optimized crossover*, given in Aggarwal et al. (1997), where crossover creates a specific pair of children. The first child, called the *Optimum* or *O*–child, has the most favorable objective value amongst all possible children and the second child, called the *Exploratory* or *E*–child, is intended to maintain diversity in the population. For a recent application of this on the CVRP, see Nazif and Lee (2012). The second (and more significant) contribution, is the *Split* routine given in Prins (2004). This involves joining all routes in a solution together (with trips to the depot removed) and then assigning consecutive subsequences to vehicles such that the overall cost is minimized. The resulting routes are guaranteed to be feasible (avoiding the need for repair heuristics) and is compatible with traditional crossover mechanisms. This paper either equaled or improved the best known solution on almost every instance tested and was the first example of GAs being competitive for VRPs.

Following this landmark paper, Berger and Barkaoui (2004) introduce a parallel GA involving several populations, each placing a different emphasis on favorable objective values and minimal constraint violation. The paper also provides various mechanisms to transfer information between populations and produces a small number of new best known solutions. Ho et al. (2008) present a GA for the MDVRP and investigate the impact of high quality initial solutions. Prins (2009) introduce a memetic algorithm for a VRP variant with a heterogeneous fleet and improve several previous best known solutions. Liu et al. (2009) investigate the related problem by determining the optimal fleet mix, and also improve many best known solutions. Ursani et al. (2011) introduce an interesting variant called the *Localized* Genetic Algorithm, which starts by decomposing an initial solution into various sub–problems (that are solved iteratively using a GA). The resulting solutions are combined to form an overall solution, which is again decomposed into sub–problems and the process iterates. Vidal et al. (2013b) present an efficient Genetic Algorithm suitable for VRPs with a combination of periodic, temporal, multi–depot, site–dependent and duration constraints. They evaluate the impact of their moves on temporal feasibility in amortized constant time and present decomposition techniques that significantly reduce run times. Their approach equaled or bettered all competing approaches on benchmark instances, performing especially well on large problems. This algorithm is then extended and further generalized in Vidal et al. (2014), which represents one of the most effective and widely applicable VRP meta–heuristics available today. Finally, Cattaruzza et al. (2014) consider the VRPM, presenting a novel local search operator which enables them to find solutions to previously unsolved problems.

2.3.4.6 Constraint Programming (CP)

Next, we will briefly discuss methods that utilize traditional Constraint Programming techniques, often in conjunction with some of the meta–heuristics mentioned above. Pesant and Gendreau (1999) and De Backer et al. (2000) implement local search and iterative improvement techniques (combined with CP) inside a Tabu Search framework for the TSP and VRP respectively. Kilby et al. (2000) compare the relative performance of approaches utilizing various amounts of CP technology for a range of VRP variants and find that more complex problems benefit most from heavier methods. Rousseau et al. (2002) present a series of neighborhoods designed to exploit the strengths of CP inside a Variable Neighborhood Search meta–heuristic. Kilby and Urli (2015) apply CP, embedded in

an Adaptive Large Neighborhood Search algorithm, to an incredibly rich FSMVRP. Using a year's worth of collected data, they produce a range of pareto–optimal fleets, trading off total cost against the likelihood of needing to hire additional vehicles, and report positive computational results compared to an equivalent MIP formulation.

2.3.4.7 Hybrid approaches and relative performance

When considering different approaches, there is a natural question as to their relative performance; unfortunately, comparing competing approaches is quite difficult, primarily due to a lack of common test instances. Historically, authors used the Solomon Instances, but many feel that these are no longer sufficiently difficult to represent a challenge to state-of-the-art VRP solvers Li et al. (2005), Uchoa et al. (2014). These authors have proposed new benchmark instances, but they have not been widely adopted (at least on the scale of the Solomon instances). This is partly because modern research has focused on variants that require specialized/modified instances, e.g. our Extended Demand Responsive Connector Problem introduced in Chapter 3. This means comparisons of sheer numerical effectiveness are extremely difficult.

Additionally, comparisons are complicated by the interactions between the meta-heuristic framework, and the specific strategies employed. To illustrate this, imagine there was a new VNS that outperformed all existing methods – did this arise from an inherent benefit in the VNS framework, or were the specific neighborhoods used particularly strong? These comparisons have become more difficult with the recent trend towards hybrid meta-heuristics Belhaiza et al. (2014), Bent and Van Hentenryck (2004a), Rousseau et al. (2002), Kilby and Urli (2015).

2.4 Applications

As evidenced by the numerous variants available, the VRP is relevant in a wide range of settings; in this thesis, we select three particular real world applications for further study. In later chapters, we propose ways to improve current operating procedures and discuss how optimization techniques could help achieve greater efficiency. In this section we provide the background knowledge required for these applications and review the surrounding literature.

2.4.1 The Demand Responsive Connector (DRC)

The design and optimization of traditional public transport schemes has received a lot of attention in the literature (Liebchen, 2007; Guihaire and Hao, 2008; Sun et al., 2008; Wong et al., 2008; Yan et al., 2012; Van Oort, 2014). However, in recent years, focus has turned to more flexible transportation systems that eschew fixed services in favor of routes and schedules determined by user bookings. These new variants are especially valuable in cases where a traditional service may be uneconomical, (e.g., areas with a low density,

or in times of low demand), and provide a practical alternative to the private motor vehicle (Fu and Xu, 2001). Outside of these cases, some jurisdictions use the greater service quality offered by flexible systems to encourage use of public transit and relieve vehicular congestion. For studies showing the potential advantages of these enhanced transport systems, we refer the reader to Fu and Xu (2001), Ferreira et al. (2007), Diana et al. (2009) and Levine et al. (2000). Overviews and examples of real world schemes as well as discussions of related current issues can be found in Potts et al. (2010), Aldaihani and Dessouky (2003), Koffman (2004), Laws et al. (2009), Mulley and Nelson (2009), Velaga et al. (2012), Nelson et al. (2010), Daniels and Mulley (2010), Mulley and Nelson (2009) and Wright (2013). On a higher level, this can be viewed as part of an emerging class of VRPs with Synchronization Constraints; for a recent review on this, we refer the reader to Drexl (2012).

2.4.1.1 Problem Description

We focus on a system known as the Demand Responsive Connector (DRC), which connects a residential area to a major transit network through one or more transfer points. In this problem, there is a group of commuters wishing to travel from their home address to the city center using public transport. Interestingly, they have the option of being transported from their house to the *closest* transit station by a shuttle service. This restriction means each vehicle operates in a geographical *zone* centered around a single station and only serves passengers in their zone, as shown in Figure 2.8. We focus simply on the task of getting passengers to the transfer point, where they can complete their journey without further assistance. While likely appropriate for settings with highly directional traffic e.g., peak hour in large cities, it is a straightforward extension for vehicles to carry passengers both to and from the station. Naturally, real world settings involve high amounts of dynamism, with uncertainty around demand levels and required travel times.

To access this service, passengers must pre–book and indicate a station further down the line they wish to travel to. Information regarding journey timing is also exchanged and a time window for collection is agreed upon (different providers offer varying levels of flexibility). Once all bookings are known, the service provider will find a suitable schedule and inform passengers of their planned pickup time (which must fall within the agreed window). The schedule must satisfy a number of natural restrictions: vehicles must start and finish at a depot, the number of passengers in a vehicle at any one time can't exceed the vehicle's capacity, the same vehicle must pick up and drop off a passenger, pick up and drop off times must fall within the agreed windows and ride time limits restrict how long passengers can spend in the vehicle. The resulting optimization problem is then to find a feasible schedule which minimizes the costs faced by the service provider.

It is not clear how to balance the inherent conflict between vehicular operational cost and the level of service received by passengers. While open to interpretation, we argue the perceived service quality is determined by the length of pick up windows given to customers and the gap between the actual pick up and the end of the window. Specifically, we say customers favor smaller windows (easier to plan around) and being collected near the end of their window (prefer to wait at home instead of in a vehicle with strangers or at a transit station). Naturally, the more restrictions that are imposed, the greater the operational

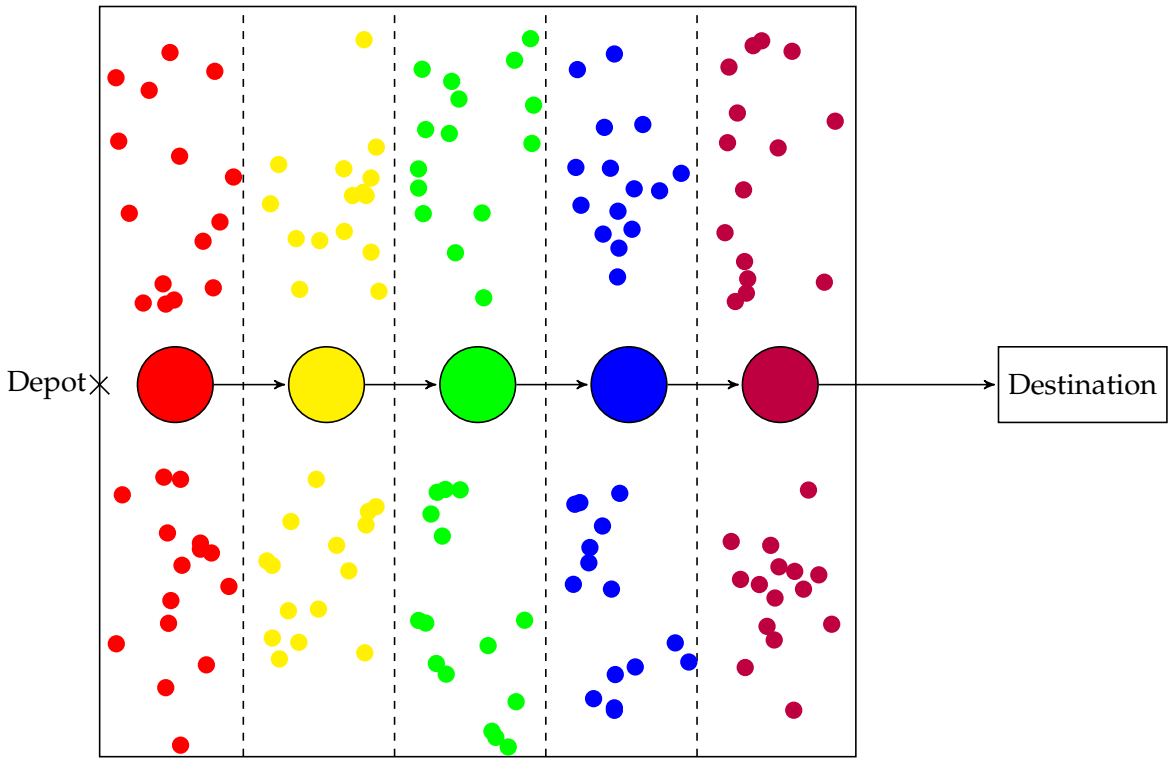


Figure 2.8: Operational layout for a DRC

cost. For our purposes, we assume the length of time windows given when a booking is a strategic policy decision and cannot be changed. We account for the second factor by performing a post optimization phase – once a schedule is found, we shift all pick up times as late as possible, without violating any time windows. Having noted these considerations on service quality, our sole objective is to minimize operational costs, which we assume is proportional to total distance traveled.

It is not clear how to set fares charged to passengers. Proponents of these schemes argue DRCs are simply feeder buses connecting passengers to transit hubs, and should be priced as such. They of course bring the positive societal externalities of public transport, such as less vehicular congestion and reduced environmental impact and serve as a vital link for those without alternative transport. Others suggest this scheme approaches something similar to an on–demand taxi service and should be priced accordingly. The required subsidy, although modest, is high compared to traditional services when expressed on a per passenger basis and exceeds what some believe is a reasonable amount. Most transport providers strike a compromise between these views and charge a modest premium on top of their transit ticket, while still running the service at a reasonable loss.

We note that this particular problem shares many similarities with some of the more general variants introduced in Section 2.3.1.2. Specifically, the DRC can be viewed as a DARP where all customers share a common destination (the station) and many share arrival windows (corresponding to catching the same service).

2.4.1.2 Literature Review

We will now briefly survey the literature surrounding DRCs. An early real world case study can be found in Cayford and Yim (2004), where they surveyed interest in, constructed machinery for, and implemented a close variant of a DRC in the city of Millbrae, California. However, unlike modern settings, the system is still centered around (multiple) fixed routes and predetermined stop locations – the savings came from adjusting which particular routes and stops were serviced at which times. Most other work focuses on the inevitable issues surrounding the operational zones. Li and Quadrifoglio (2009) investigate how to determine the optimal number of zones to employ; too many leads to a duplication of service with unneeded vehicles sitting idle, but too few results in vehicles traveling excessive distances to collect passengers. Chandra and Quadrifoglio (2013a) consider the case where a single vehicle operates from each station and investigate how frequently it should operate. Infrequent services means more passengers are carried each trip, but their average riding/waiting time will increase. Conversely, shorter cycles are preferred by customers, but the lower vehicle utilization means additional kilometers driving to and from the depot. They give various ways of measuring this conflict and present methods to find acceptable trade–offs. Chandra and Quadrifoglio (2013b) investigate the impact of network connectivity on the performance of a feeder service; well connected networks allow for easy operation, but in situations where this is not possible, it's important to know the expected loss of efficiency. Finally, the decision to employ a DRC scheme or a traditional feeder services depends on the level of demand; at higher levels, scheduling sufficient vehicles to collect passengers from individual houses becomes both slower for customers and prohibitively expensive for the provider. Ways to determine this critical threshold are investigated in Li and Quadrifoglio (2010).

2.4.1.3 Problem Formulation

We now formally introduce the scheduling of a DRC vehicle as an optimization problem, and give a mixed integer linear programming formulation. As noted above, the inherent structure allows each station to be considered independently, so our formulation will be presented in this manner. For the transit station under consideration, let P denote the set of pickup requests. For each request $i \in P$ there is a quantity of passengers to be picked up, a ride time limit, an earliest time and a latest time at which the passengers can be picked up, given by q_i, r_i, e_i^{PickUp} and l_i^{PickUp} respectively. Extending this, for each pickup request $i \in P$, there is an associated latest scheduled departure from the station that ensures arrival at the downstream station at or before the specified time; this departure occurs at l_i^{Station}. Of course, a commuter may be dropped off with sufficient time to take an earlier service. Finally, let K denote the set of vehicles, with each having a capacity Q. Again, we assume we have sufficient vehicles to serve all passengers. A trivial bound on K is the number of passengers, though solutions from a heuristic can be used as a more informed guide.

Finally, we should discuss the creation of time windows. As mentioned above, upon booking the customer gives their destination station and the service provider (immediately) gives an earliest pick up time. This is found by calculating the latest time a passenger

could leave his residence and drive to the nearest train station (arriving at the same moment the train does) and subtracting a small, fixed amount of time, referred to as the *time flexibility* or t_{flex}. This implies for a request $i \in P$, $e_i^{\text{PickUp}} + t_{\text{flex}} = l_i^{\text{PickUp}}$. If the direct travel time request i to the station is given by $t_{i,\text{station}}$, then the earliest the passenger may arrive at the station is given by $e_i^{\text{PickUp}} + t_{i,\text{station}}$; similarly, the latest time is given by $l_i^{\text{PickUp}} + t_{i,\text{station}}$. An illustrative diagram is given in Figure 2.9.

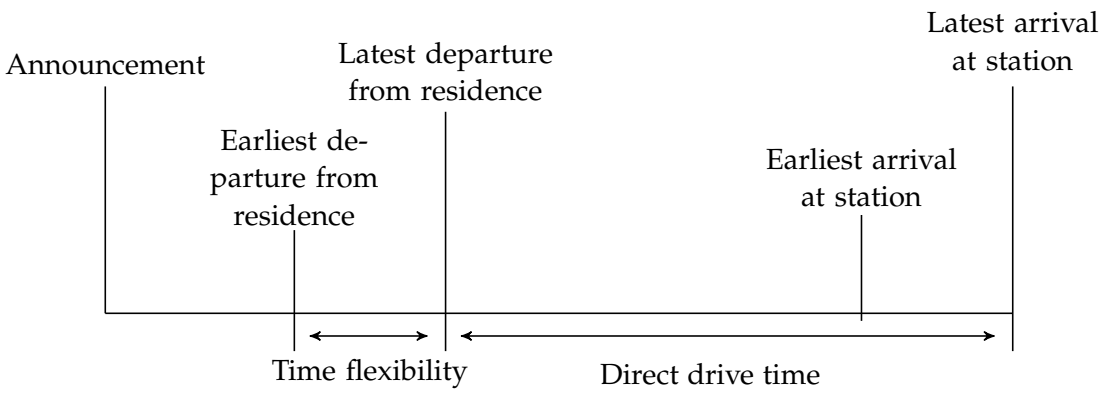


Figure 2.9: The time line of a single passenger

Like most VRP variants, the DRC problem can be represented on a graph, $G(\mathbf{N}, \mathbf{A})$. The node set contains a pickup node for each request $i \in P$, plus a depot node, given by $\{0\}$. In lieu of a station node, we include a dropoff node for each pickup request. These represent the single station, but may have different time windows (as passengers need to catch trains at different times). Specifically, these nodes are represented by the set S, with the node pertaining to customer i given by $s_i \in S, i \in P$. Consequently, the full node set is given by $N = P \cup S \cup \{0\}$. For now, we assume that an arc exists between every pair of nodes in N, i.e., $A = \{(i, j) \in N \times N\}$ and each arc has an associated cost and travel time, given by c_{ij} and t_{ij} (naturally, $c_{ij} = t_{ij} = 0$ if both i and j are dropoff nodes). Let x_{ij}^k be a binary decision variable indicating if vehicle $k \in K$ traverses arc $(i, j) \in A$ ($x_{ij}^k = 1$) or not ($x_{ij}^k = 0$). Let T_i be the time that a vehicle departs from node $i \in N$. Finally, let Q_i give the number of passengers in a vehicle when it departs from node $i \in N$. Finally, we now present the integer programming formulation of the Demand Responsive Connector problem:

Integer Program 5:

$$\min \quad \sum_{i\in N}\sum_{j\in N}\sum_{k\in K} c_{ij}x_{ij}^k \tag{2.27}$$

$$\text{s.t.} \quad \sum_{j\in N}\sum_{k\in K} x_{ji}^k = 1, \qquad i \in P \tag{2.28}$$

$$\sum_{i\in P} x_{0i}^k \leq 1, \qquad k \in K \tag{2.29}$$

$$\sum_{j\in N} x_{ji}^k - \sum_{j\in N} x_{ij}^k = 0, i \in N\backslash\{0\}, \quad k \in K \tag{2.30}$$

$$\sum_{j\in N} x_{ji}^k = \sum_{j\in N\backslash\{0\}} x_{js_i}^k, \qquad i \in P, k \in K \tag{2.31}$$

$$T_j \geq \sum_{k\in K} x_{ij}^k \left(T_i + t_{ij}\right), \qquad j \in N \setminus \{0\}, i \in N \tag{2.32}$$

$$e_i^{\text{PickUp}} \leq T_i \leq l_i^{\text{PickUp}}, \qquad i \in P \tag{2.33}$$

$$e_j^{\text{Station}} \leq T_j \leq l_j^{\text{Station}} \qquad j \in S \tag{2.34}$$

$$T_{s_i} \geq T_i, \qquad i \in P \tag{2.35}$$

$$T_{s_i} - T_i \leq r_i, \qquad i \in P \tag{2.36}$$

$$Q_j = 0, \qquad j \in \{0\} \cup S \tag{2.37}$$

$$Q_i = \sum_{j\in N}\sum_{k\in K} x_{ji}^k (Q_j + q_i), \qquad i \in P \tag{2.38}$$

$$Q_i \leq Q, \qquad i \in P \tag{2.39}$$

$$x_{ij}^k \in \{0,1\}, \qquad i,j \in N, k \in K \tag{2.40}$$

$$T_i \geq 0, \qquad i \in N \tag{2.41}$$

$$Q_i \geq 0, \qquad i \in N \tag{2.42}$$

The objective function minimizes the operational cost of the service vehicles. Constraints (2.28) ensure every request is fulfilled by a vehicle. Constraints (2.29) and (2.30) ensure that vehicles start and finish at the depot and depart from other locations they visit in between. Constraints (2.31) ensure that the vehicle which drops off a passenger also collected them originally. Constraints (2.32) ensure the consistency of the departure times along the route of a vehicle. Constraints (2.33) and (2.34) enforces the time windows associated with requests. Constraints (2.35) ensure that passengers are picked up before they are dropped off. Constraints (2.36) checks that passengers aren't in the vehicle for longer than their permitted ride time limit. Constraints (2.37) ensure that vehicles depart from the depot empty and that when a vehicle arrives at a transit station, all passengers on board are dropped off. Constraints (2.38) update the number of passengers after each pickup and constraints (2.39) ensure that this never exceeds vehicle capacity. Finally, constraints (2.40), (2.41), and (2.42) set the variable types and bounds. Clearly, constraints (2.32) and (2.38) are nonlinear. However, they can easily be linearized as shown below:

$$T_j \geq T_i + t_{ij} - M(1 - \sum_{k\in K} x_{ij}^k), \qquad j \in N \setminus \{0\}, i \in N \tag{2.32'}$$

$$Q_i \leq Q_j + q_i + M(1 - \sum_{k\in K} x_{ji}^k), \qquad i \in P, j \in N \tag{2.38'}$$

$$Q_i \geq Q_j + q_i - M(1 - \sum_{k\in K} x_{ji}^k), \qquad i \in P, j \in N \tag{2.38''}$$

This formulation can easily be strengthened with the following steps:

- **Remove unnecessary variables:** As presented, this formulation contains arcs rendered infeasible in integer solutions due to time windows. However, they may be feasible in fractional solutions, leading to unnecessary nodes in the branch and bound tree.
- **Remove symmetry:** With a homogeneous fleet, we can see a permutation of the vehicle indices gives rise to otherwise identical solutions. This symmetry is well known to cause difficulty in solving the VRP (Wolsey and Nemhauser, 2014).
- **Valid cuts and inequalities:** Although valid cuts can be useful for all integer programs, there has been much work specifically on the PDP/DARP, as in Cordeau (2006), Røpke et al. (2007) and Parragh (2011). As the DRC is a special case of the PDP/DARP, these cuts can be applied directly, but we expect there are natural extensions that exploit the inherent structure introduced by the overlap between passengers' destinations.

2.4.2 Ride Sharing

Ride Sharing systems involve matching participants with similar travel plans so they may share their journey and the associated costs. In their broadest form, they can make use of almost any medium, e.g., online forums, community notice boards, word of mouth or smart phone applications. Importantly, there is some way for participants to exchange the relevant information, such as their respective origins, destinations and any time restrictions. In the last few years there has been a rapid growth in the number and diversity of ridesharing schemes – for a recent review and survey, we refer readers to Furuhata et al. (2013).

2.4.2.1 Dynamic Ride Sharing

For this thesis, we are interested in Dynamic Ride Sharing (DRS), characterized by non–recurring matches organized at short notice (typically the minimum time required between a trip announcement and the preferred departure time is around 30 minutes, though participants may announce their trip earlier if desired). For a more elaborate discussion of dynamic ridesharing schemes, we refer the reader to Agatz et al. (2012a). We focus on ridesharing schemes that involve drivers who will make their planned trip regardless of whether they are matched with a rider (as supported by Flinc and Carma (formerly Avego)), instead of ridesharing systems in which a driver is making a trip for the sole purpose of earning the fare (as offered by Uber, Lyft and SideCar). The latter schemes notably cause significant disruption for the (highly regulated) local taxi industry, as drivers typically operate outside of expensive licensing schemes.

DRS is particularly useful in locations with a density insufficient for traditional public transport services. They also offer the same societal benefits, such as reduced congestion and vehicle emissions. Naturally, many participants may be attracted more by the personal benefits (e.g., sharing expenses, or saving time by being able to use express or High

Occupancy Vehicle (HOV) lanes). Finally, DRS schemes have the added advantage of not requiring the re–prioritisation of existing road space or a large capital investment (which can cause political difficulties).

We will now formally describe the problem of Dynamic Ride Sharing and give the notation used. There are two sets of participants: drivers, denoted by D, who are driving a vehicle with spare capacity, and riders, denoted by R, who want to travel with a driver going in a similar direction. Each participant $i \in D \cup R$ has an origin, a destination, an earliest time he can depart from his origin, e_i^{Dept} and a latest time by which he should arrive at his destination, l_i^{Arr}. Denoting the travel time between the participant's origin and destination by $\hat{t}_i$, we can calculate an earliest arrival time, $e_i^{\text{Arr}} = e_i^{\text{Dept}} + \hat{t}_i$ and a latest departure time, $l_i^{\text{Dept}} = l_i^{\text{Arr}} - \hat{t}_i$. The time flexibility, t_{flex}, indicates how much earlier a participant is willing to leave in exchange for being paired with another participant, i.e., $t_{\text{flex}} = l_i^{\text{Arr}} - e_i^{\text{Dept}} - \hat{t}_i$. We assume that the value given by t_{flex} is a maximum, and participants prefer to depart closer to l_i^{Dept} (and arrive closer to l_i^{Arr}). Note that by construction, satisfying the time window at the origin guarantees satisfying it at the destination. There is also a lead time, which represents the time between the provider becoming aware of participant i and e_i^{Dept}. We assume that drivers can only serve one rider and that riders can only be served by one driver (i.e., we are not considering transfers). Restricting ourselves to matches between a single driver and a single rider not only simplifies the planning process for the service provider, but also ensures that the resulting trips are easy to execute. It is relatively easy in this setting to accommodate multiple passengers traveling in a group (provided they all have the exact same travel plans), by treating them as a single request (and checking that the vehicle of the driver assigned to the group has sufficient spare capacity).

Because many participants are motivated by the opportunity to reduce the cost of transport, matches are only permitted if the cost incurred is less than if the participants traveled by themselves (such matchings are said to be *cost–feasible*). Assuming that the cost is proportional to the distance traveled, this implies that the length of the combined trip has to be shorter than the sum of the lengths of the two individual trips. In Figure 2.10, this occurs when $d_1 + d_2 > d_3 + d_2 + d_4$. If $c(d_i)$ gives the cost associated with distance d_i, then the cost saving of the match is given by $c(d_1) - c(d_3) - c(d_4)$.

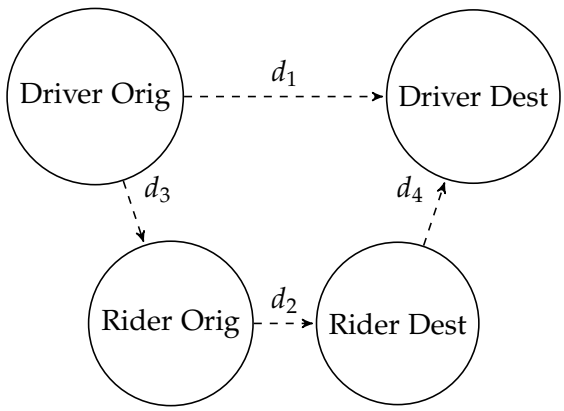


Figure 2.10: Potential matching between a driver and a rider.

We should now discuss the issue of determining the fare paid by the rider (or equivalently, how the savings are split between the two participants and the service provider). Typically the provider estimates the total cost savings (compared with each participant traveling separately) and takes a commission (either a flat fee, or a percentage of the savings). The remainder is then distributed amongst the participants through some manner. Agatz et al. (2010) suggest a pricing scheme in which the costs of the combined trip are shared in proportion to those of the individual trips, as given in (2.43) and (2.44), where f_d and f_r are the costs borne by the driver and rider respectively.

$$f_d = \frac{c(d_1)}{c(d_1)+c(d_2)} \left(c(d_3) + c(d_2) + c(d_4)\right) \tag{2.43}$$

$$f_r = \frac{c(d_2)}{c(d_1)+c(d_2)} \left(c(d_3) + c(d_2) + c(d_4)\right) \tag{2.44}$$

This scheme guarantees that cost incurred by each participant is less than the cost incurred by driving by themselves. In the future, it may be of interest to consider other factors borne by the driver and somehow split these as well e.g., avoiding tolls charged to single occupancy vehicles, or the cost of parking at the destination. A more comprehensive cost assessment may result in more cost-feasible matches and thus, potentially, in larger system-wide cost savings. A naive implementation is to simply include these factors with the existing distance–based costs and share them proportionally, as outlined in Equations (2.43) and (2.43). However, there are two broad problems with this approach.

First, when discussing additional costs, it is not always clear how they should be shared, especially when they would have been incurred by only one of the participants (if both remained unmatched). As an example, consider a driver/rider pair where the driver travels a tolled road only because he is dropping off the rider; his optimal unmatched route is untolled). Of course, even determining the driver's "optimal unmatched" route is not straightforward – given a choice between a short, tolled route, and a longer, untolled one, it is unclear what should be considered optimal. Similarly, you could have a matched driver/rider pair where one participant wants to take the tolled road, but the other does not; this is another practical consideration a real world operator would need to overcome.

Secondly, the inclusion of non–distance based cost will mean some matches will represent a saving, even if the participants combined travel distance exceeds what it would be if they traveled individually. This can increase total emissions and congestion, and these adverse societal impacts weakens any case for public subsidies.

Naturally, the service provider may wish to maximize their own profit – this could involve maximizing matches (if they charge a flat fee) or maximizing system wide cost savings (if they take a percentage based commission).

2.4.2.2 Literature Review

With respect to past work, Agatz et al. (2010) is amongst the first papers published on DRS, and provides an excellent overview. They give the key features and characteristics

which distinguish DRS from related schemes and discuss a wide range of practical considerations concerning implementation. They formulate the basic problem and discuss natural extensions and variants. Finally, they list the challenges and obstacles DRS practitioners faced at the time of writing. Agatz et al. (2011) simulates the operation of a ridesharing scheme in Atlanta, investigating the performance of the system under different participation rates, with different algorithms to match participants, and when participants have different levels of flexibility (either through larger time windows, or in the role that they play). They also show that the system typically stabilizes to a constant participation rate and discuss the factors which affect this. Agatz et al. (2012b) investigates the situation where a driver and a rider, matched by the provider but not to each other, can improve their cost savings by forming a private match. A way to prevent this from happening is proposed and its effect on solution quality is investigated. Furuhata et al. (2013) provides an in–depth classification system for existing ridesharing schemes, and identify some of the current challenges and opportunities.

Solving the basic problem given in Section 2.4.2.3 is computationally simple and there are several suitable polynomial time algorithms available; see Hopcroft and Karp (1973) or Alt et al. (1991) if all the objective coefficients are the same, or Schwartz et al. (2005) for more general cases. The natural extensions to this problem, (like drivers serving multiple riders or riders transferring between drivers) break the problem's inherent structure and force the use of more general (and non polynomial) solution methods.

DRS can be seen as crowdsourcing applied to a transportation problem (Alt et al., 2010; Pedersen et al., 2013). Of particular interest to our upcoming work is Arslan et al. (2015), where a large retailer complements their traditional delivery service with so called "crowdshipping". Under this setting, parcels can be assigned either to traditional couriers (who are expensive but will deliver anywhere), or with private commuters (who cost less but can't detour greatly from their original route). DRS is also somewhat related to the problem of sharing taxis between riders; for a recent review of this, see Hosni et al. (2014).

2.4.2.3 Problem Formulation

Finding an optimal pairing between drivers and riders can be done by solving a weighted bipartite matching problem (Agatz et al., 2011), where an arc exists between a driver and a rider only if the match is both cost and time feasible, and the weight on the arc is the score representing the desirability of the match (an example matching network is given in Figure 2.11). Specifically, the match (or arc) between driver $i \in D$ and rider $j \in R$ is represented by binary variable x_{ij}, and has an associated objective function coefficient $\hat{c}_{ij}$. An integer program that finds the most desirable set of matches is given below.

Integer Program 6:

$$\begin{aligned} \max \sum_{i \in D} \sum_{j \in R} \hat{c}_{ij} x_{ij} \\ \text{s.t.} \sum_{j \in R} x_{ij} \ \leq \ 1, i \in D \\ \sum_{i \in D} x_{ij} \ \leq \ 1, j \in R. \end{aligned}$$

If desired, artificial participants and dummy variables can be added to convert the inequalities to equalities (which is necessary for some solution algorithms).

2.4.2.4 Extensions

We will now detail some additional considerations surrounding DRS and discuss possible extensions.

Funding and subsidies:

We have so far assumed that the operator's only source of funding was commissions earnt from matching participants, but there is a reasonable case to be made for public subsidies. Many of the benefits (reduced congestion, lower vehicular emissions) are societal in nature, and the costs involved are small compared to that for new infrastructure projects. However, we acknowledge that financial support for private organizations can be politically sensitive, especially in an age of austerity when other programs are being defunded. Alternatively, a public institution could provide non–financial support, such as free use of office space, or publicity through local publications like community newspapers.

Choice of objective:

Aside from maximizing their revenue, there are other objectives the provider could reasonably pursue. One such choice is to simply maximize the number of participants who are successfully matched (although pairings must still be feasible and represent a cost saving compared to participants traveling individually). From a commercial perspective, this will maximize the reach of the program and can help establish a customer base (although some matches may be of a low quality).

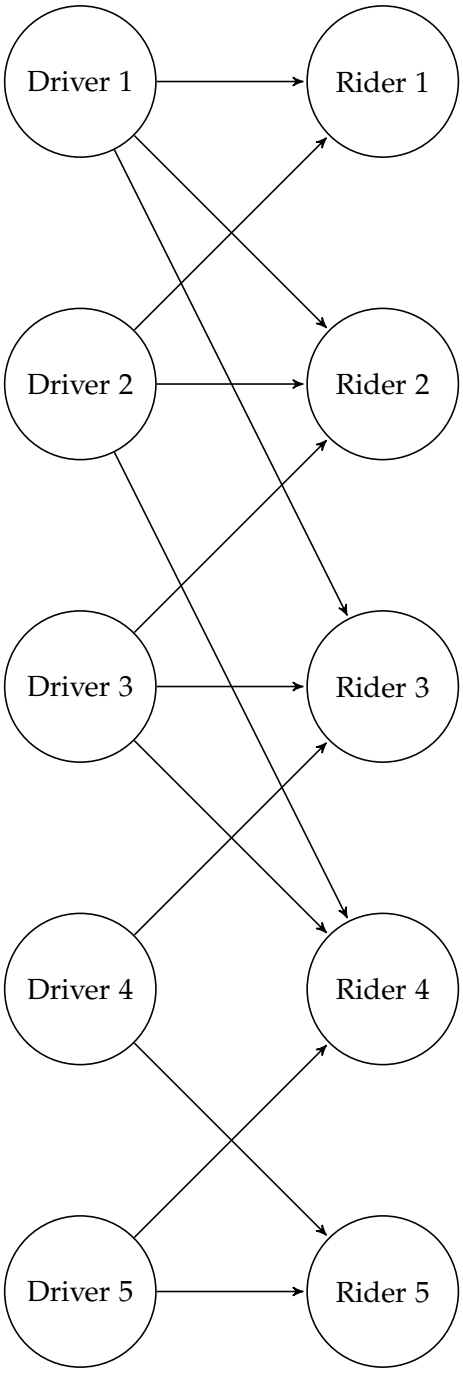


Figure 2.11: Example matching between ride sharing participants

If the provider is receiving public funds, they may be required to consider wider societal benefits. This could mean minimizing the total miles driven or overall vehicular emissions (again, we assume unmatched participants simply drive themselves). It is interesting to note that as the cost of a trip tends to be proportional to distance traveled, solutions which maximize cost savings (and revenue for the service provider) tend to also minimize miles driven. Similarly, as vehicular emissions also tend to be proportional to distance traveled, solutions that minimize total distance tend to also have low vehicular emissions. Importantly, this means that a "selfish" service provider who maximizes their own revenue still provides close to the maximum amount of wider benefits.

Time constraints:

The management and implementation of time window constraints can have a large impact on the perceived quality of service, with both riders and drivers wanting a match that deviates as little as possible from their preferred latest departure time. This can be enforced through smaller values of t_{flex}, although by making it too small, we can reduce the number of feasible matches (and the resulting cost savings). By the same token, the required lead time also impacts how the service is perceived; participants want the flexibility of committing as late as possible, but also want the better quality of matches enabled by longer lead times.

Multiple Passengers:

A natural extension is to allow drivers to serve multiple riders simultaneously (we assume riders have different origins and/or destinations). Such a scheme would likely allow more riders to be served and would add robustness to the system (by protecting against a shortage of drivers). However, this would break the nice structure inherent in the optimization problem and necessitate more general solution methods. The policy also introduces equity concerns when matching participants; specifically, how do you decide which riders will share a journey, and who gets picked up / dropped off first. Also, riders may want a discount as compensation for the longer journey (and some riders may be prepared to pay more for a direct trip); similarly, drivers would also (quite justifiably) expect an additional payment. In both cases, we would need to extend our proportional method used for determining fares to allow for multiple participants. Finally, if there is not a shortage of drivers, then assigning multiple riders to a single driver may mean other drivers do not receive a match at all – if this occurs regularly, unmatched drivers may leave the system.

Transfer constraints:

A similar extension is to allow riders to transfer between different drivers. This greatly expands the solution space and will likely allow improved objective values. However, this also breaks the natural integrality of the problem and requires more generalized (and much slower) solution methods. It also introduces fairness concerns (regarding the selection of riders that will have to transfer) and represents a deterioration in the quality of service received. For this reason, the rider may expect a financial discount; even without this, we would still need to extend our method for calculating fares.

Round Trips:

A concern shared by many potential riders relates to the reliability of the system with respect to return trips, i.e., that they'll find a match on their outgoing trip, but will be unable to find a journey home. To get around this, riders could specify that they only want a match for one leg if they can be guaranteed a match for the other (with a driver who has already announced a suitable trip). Naturally, this requirement will reduce the number of pairings found, although long lead times would make it easier to offer this guarantee.

2.4.3 Dynamic/Stochastic VRP

While Sections 2.4.1 and 2.4.2 discussed the movement of people, there has also been a lot of attention paid to the efficient movement of goods (often motivated by commercial delivery firms looking to reduce their operational expenditure). One challenge of particular interest is how to handle various sources of uncertainty inherent in real world settings. This can include unknown travel times (Laporte et al., 1992) or vehicles that can break down, forcing the rapid generation of a new operational plan (Li et al., 2009; Mu et al., 2011). Oftentimes the uncertainty can come from the customer set, with many firms facing problems with a mixture of *pre–booked* customers (known from the start of the planning horizon) and *unknown* customers (who are only revealed part way through the planning horizon, but should still be served if possible (Pillac et al., 2012)). A related variant is when all customers are known in advance, but the provider doesn't know which ones will actually require a visit (Waters, 1989). Additionally, customers could have an unknown service time or require an uncertain quantity of goods (Dror et al., 1989), with the true values not being realized until their service concludes. These form part of the field known as *Robust Vehicle Routing*, which involve VRPs where there is uncertainty in some information, such as travel costs/time, demand, or the set of customers Ordóñez (2010). When discussing sources of uncertainty, there is an important distinction to be made; in *dynamic* problems, the operator has a (potentially limited) opportunity to make changes once the unknown information is revealed, but in *stochastic* problems, no such opportunity exists (for a full review of the similarities and differences between these problems, see Ritzinger et al. (2015)). In this thesis we focus on dynamic problems, where the operator has a chance to respond. Although there are many different solution methods available, most fall into two broad classes:

- **Periodic reoptimization / Rolling horizon strategies:** These approaches generate an initial solution at the start of the period, using all available information. The optimization algorithm is then reapplied, either when new information becomes available, or after predetermined intervals. Importantly, this can be done using algorithms designed for the static version of the specific VRP variant under consideration. However, the optimization needs to be completed before a revised plan can be provided, potentially delaying the rerouting of vehicles. In some settings, there may be a cost associated with changing decisions made in previous iterations – this often represents the inconvenience of informing system participants of changed/updated

times/routes. For past papers using these methods, see Yang et al. (2004); Chen and Xu (2006) or Montemanni et al. (2005).

- **Continuous reoptimization:** As the name suggests, these strategies continuously optimize the solution throughout the planning horizon, often maintaining a pool of high quality solutions. When new information becomes available, the stored solutions are analyzed and combined to develop a good response. This approach often allows decisions to be made in a shorter time frame, meaning routes can be updated sooner. Naturally, as a practical consideration, operators are incentivized to not commit to operational decisions until the last possible moment (to maximize their ability to respond to late changes). A common method used is Approximate Dynamic Programming; for examples of these approaches, we refer the reader to Powell (2007), Godfrey and Powell (2002) and Simão et al. (2009). For examples of other approaches used, we refer the reader to Taillard et al. (1997); Gendreau et al. (1999); Ichoua et al. (2000, 2003); Chang et al. (2003); Bent and Van Hentenryck (2004b) or Attanasio et al. (2004).

Finally, as a practical consideration, operators are incentivized to not commit to operational decisions until the last possible moment (to maximize their ability to respond to late changes). This applies regardless of the solution method used. If a decision is due to be made before an optimization run will finish, it should be fixed (based on information from previous runs).

2.5 Summary

In this chapter, we have given an introduction to the Vehicle Routing Problem and its numerous variants. We have also discussed a range of different solution methods and reviewed the surrounding literature. Finally, we have presented three real world applications which we will build upon and extend in the following chapters.

Chapter 3

The Extended Demand Responsive Connector

3.1 Introduction

Worldwide, rising population levels are causing the urbanization of ever smaller cities, placing additional pressure on straining roading networks. Instead of expensive infrastructure upgrades, many transport authorities want to relieve this through greater utilization of public transport services. In this chapter, we investigate one such scheme, the Demand Responsive Connector (DRC), which shows great promise as a way to connect commuters from low density areas with the wider public transport network. However, critics (with some justification) say the service is too expensive and a poor use of limited transit funds. Indeed, with the recent global economic downturn and the so called "Age of Austerity", such claims have only intensified. It is our belief that this expense is, at least in part, caused by unnecessarily strict operational policies that limit vehicles to only serving customers from a single zone. For this reason, we propose a variant where vehicles are free to serve all passengers and may take them to any compatible transit station. While the removal of the zonal restrictions should lower operational costs, there is a legitimate argument that this will reduce the quality of service received by customers. We design and build a simulation environment to measure the performance of both the traditional and extended schemes under a range of conditions, and we illustrate that our variant can provide significant cost savings with minimal deterioration in service quality. We end with final remarks and a discussion of future work.

To perform these simulations, we provide both exact and heuristic methods that involve mechanisms designed for the complexities of our extended problem. Specifically, we note the permissible arrival windows at transit hubs are dependent on both the customer and station under consideration, an intricacy not seen in related variants.

3.2 The Extended Demand Responsive Connector (DRC)

The *Demand Responsive Connector* (DRC), given in Section 2.4.1, is an effective way to connect a residential area to a major transit network through one or more transfer points. Previous DRC services have involved vehicles operating inside fixed *zones* (as in Figure 2.8),

which significantly reduced the difficulty of vehicle scheduling. However, we feel that with the recent advancement in routing algorithms and the availability of cheap computational power, this represents an unnecessary (and expensive) operational restriction. We propose a variant, the Extended Demand Responsive Connector (EDRC), where vehicles may collect any passenger and drop them off at any station (provided the customer can still catch a suitable service to their final destination). The scenario in Figure 3.1, illustrates the benefit of removing the zonal boundaries. By allowing a single vehicle to serve the two customers near the boundary, we introduce a single additional vehicular trip, given by the green arc. In exchange, we remove the need for the three red arcs and reduce the number of vehicles needed; indeed, the latter is often an operator's primary objective due to the high fixed and capital costs involved.

Additionally, as commuters' arrival times at CBD stations during rush hour are normally distributed and are independent of the original station (McGuckin and Srinivasan, 2003), the arrival of passengers at upstream stations must follow the same distribution. This gives rise to a series of consecutive peaks, as shown in Figure 3.2 (we assume each station has a similar number of passengers, although this needn't be true). This naturally suggests a strategy of redeploying vehicles at later stations to match the shifting demand, which is obviously prohibited under the zonal system. Finally, although we show a single depot in Figure 3.1, such restrictions are unnecessary, and we now assume each vehicle may have a unique depot (possibly the driver's own residence). Of course, this generalization is not unique to the extended variant.

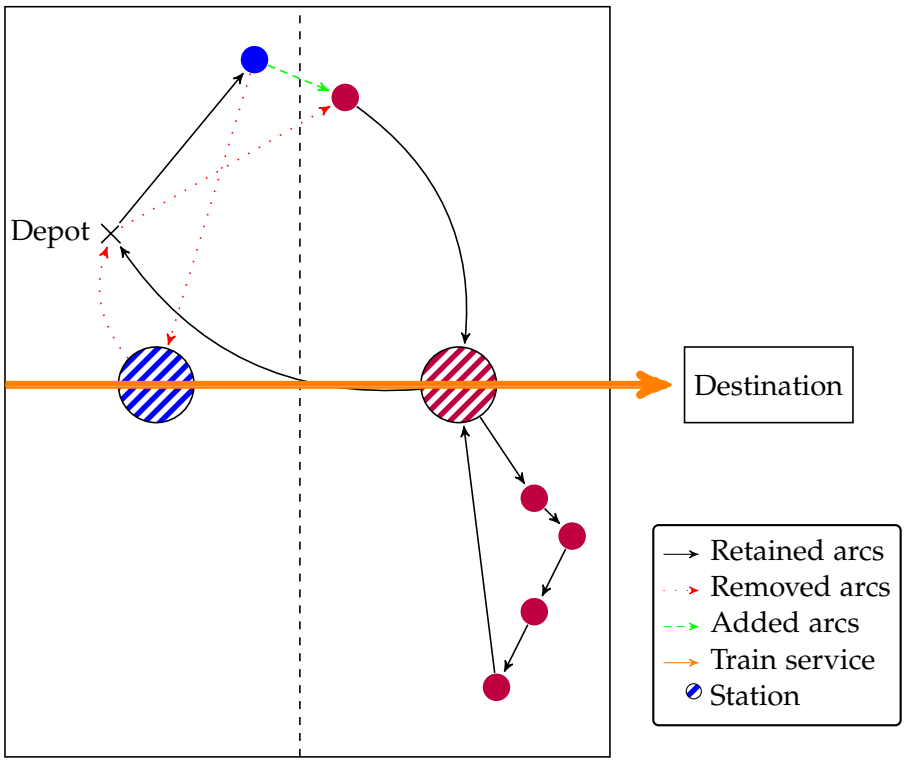


Figure 3.1: Network reduction arising from the EDRC

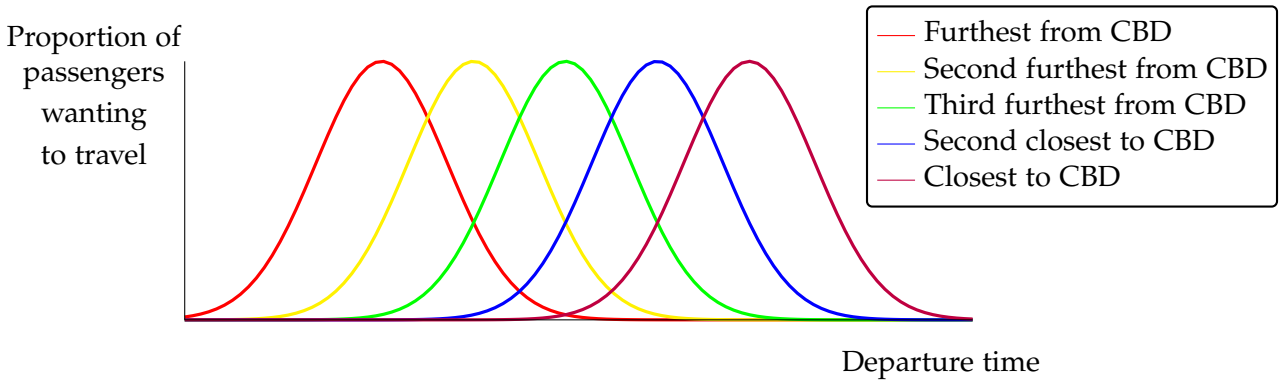


Figure 3.2: Departure time for passengers in different locations

3.2.1 Problem description

As in the simpler, zone based variant described in Section 2.4.1.1, customers wishing to use the EDRC must book in advance and provide key information (like their origin, the number of people traveling and contact details). However, instead of their preferred departure time from the closest station, they will be asked for their preferred arrival time at a downstream station of their choice. If this doesn't coincide with the arrival time of a service, it will be mapped to the closest one. They will still be immediately given an earliest time at which they must be available for pickup, and a subsequent planned pick up time once all bookings are known. The schedule must again satisfy the same natural constraints: vehicles must start and finish at a depot, the number of passengers in a vehicle at any one time can't exceed the capacity, the vehicle that picks up a passenger must also drop them off, arrival and departure times must fall within the agreed upon windows, and ride time limits restrict a passenger's maximum journey time.

The resulting optimization problem is then to find a feasible schedule which minimizes the cost faced by the service provider. As discussed earlier, this is a combination of the vehicular operational cost and the fixed cost of vehicle ownership, e.g., registration, storage, insurance, etc. Any transport firm providing this service on a regular basis will naturally aim to serve all customers on the vast majority of days (especially if they are receiving public funds), suggesting that the fleet will often be larger than is necessary. Additionally, such high level decisions cannot be altered in the short term. Hence, minimizing the operational cost and considering the vehicle limit to be a constraint represents a better model for a daily planning problem. Of course, determining the optimal fleet is a well studied optimization problem in itself. The static version can be formulated using only a slight alteration of the work covered here, but stochastic variants require complex, specialized algorithms.

When designing schedules for the larger public transport network, transit authorities have two (conflicting) objectives; they want simple schedules (which are easy to remember) that accurately match demand patterns (so services are actually used). A common approach is to run a variety of services on a regular timetable and complement these with limited–stops or "express" versions at peak hours. For our problem, this means every passen-

ger/station pair has a latest service feasible for that commuter, with some known departure time. In general, the quickest journey from the passenger's residence may require a station other than the closest; consequently, determining customer time windows follows a slightly different process to that from Section 2.4.1.3. Specifically, we note that the last service associated with each passenger/station pair implies a latest time the commuter could leave home; we simply take the latest departure time over all stations. To find an earliest pick up time, the provider subtracts a small, fixed amount of time, again called the *time flexibility* or t_{flex}. Passengers must accept that they can be picked up at any point in time within this window, with larger values of t_{flex} allowing for greater cost–effectiveness. An illustrative diagram is given in Figure 3.3, where the closest transit hub is station two.

We note that in free flowing traffic conditions, or under infrequent public transport schedules, it may be faster to drive most of the journey, and simply catch the train at the last possible station. This would lead to all customers having their time windows calculated as if they were departing from the final station (though they could still be dropped off at any compatible upstream station). However, such conditions are not representative of rush–hour (where these transport schemes are typically used).

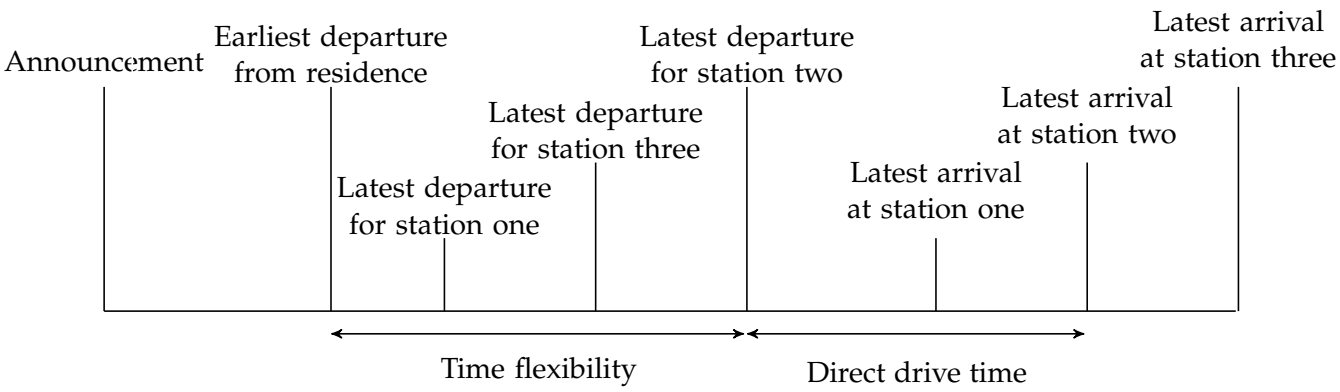


Figure 3.3: The time line of a single passenger

3.2.2 Related Work

Our problem is similar to a number of existing VRP variants. As a vehicle can drop off passengers at any station from a feasible set, it has links to problems involving multiple depots and/or intermediate facilities. Additionally, as vehicles make additional trips after dropping off passengers at a station, there are similarities with variants that use vehicles multiple times. Lastly, there are obvious links with the Dial–A–Ride–Problem (DARP), which also involves the transport of passengers using a fleet of service vehicles. However, there are also critical differences. In most vehicle routing and scheduling variants, the depots seldom have time constraints (other than the start and end of the planning period), and even if they do, they are fixed. However, in the context of the EDRC, the concept of a depot is unclear; the base from which vehicles start and leave is different to the stations, where customers (equivalent to goods in most VRP variants) travel to. Additionally, a passenger's time windows at a station are both station and passenger dependent; this dual–dependency is quite complex and is not seen in other VRP variants. This work is also quite different from the typical DARP; in our setting, passengers wish to travel to one

of a set of locations (without having a preference), which both complicates the scheduling task and provides opportunities for greater efficiency.

If we consider transit services to be vehicles with predetermined routes, then the problem becomes similar to those involving transfers. Shen and Quadrifoglio (2012); Quadrifoglio et al. (2008); Masson et al. (2014) and Deleplanque and Quilliot (2013) all study generalized Dial–A–Ride Problems with Transfers (DARPT), where the region of interest is divided into zones with dedicated transfer points (the use of which may be optional). Broadly speaking, they find that fewer, larger zones lead to greater cost effectiveness, while potentially degrading the quality of service received. In contrast to our problem, each customer could have a unique destination, reducing the scope for efficiency gains through effective routing. Additionally, transfers in the DARPT require the coordination of different vehicles; in our problem, the fixed nature of the transit schedule (and its use in creating passenger time windows) poses different challenges and opportunities. Especially relevant to our work is the Integrated Dial–A–Ride Problem (IDARP), introduced in Häll et al. (2009). In this setting, the service provider must create a journey plan for each passenger, satisfying all natural constraints, using a mixture of public transit and the provider's own vehicles. However, there are again key differences: assigning passengers to use public transport is not mandatory, a passenger's final destination need not be a transit hub (meaning they require transport both before and after their public transit use), and the final destinations of customers may again be unique. Importantly, the differences inherent in our problem (such as the common set of stations and the fixed transit schedule) introduce a well–defined structure that can be exploited through specialized algorithms, justifying research in this area.

3.3 An Integer Programming Formulation

Although we use heuristic methods to obtain solutions in our later computational study, we introduce a mixed integer linear programming formulation to present the problem in a precise manner and to validate our heuristic. As this formulation equally applies to the zonal variant (and to avoid duplicating terms), we reuse notation given in 2.4.1.3 wherever possible. Consequently, our network is presented on a graph, $G = (N, A)$. P denotes the set of pickup requests, and each request $i \in P$ has an associated number of passengers, an earliest pick time and a latest pick up time, given by q_i, e_i^{PickUp} and l_i^{PickUp} respectively. Additionally, let the set of vehicles be given by K, with each having a capacity of Q. Let t_{ij} and c_{ij} denote the travel time and travel cost between locations $i,\ j \in N$. Finally, the set of transit stations where passengers can transfer to the major transit network is given by S.

We introduce a series of dropoff nodes for each request, S^i, with one such node for every station associated with (at least) one service that respects the time windows for request $i \in P$. The latest such service departs station $j \in S^i$ at l_j^i (of course, passengers may catch a previous service if dropped off sufficiently early). The latest time the passenger(s) from request i can feasibly leave their house is given by $l_i^{\text{PickUp}} = \max\{l_j^i - t_{ij}\}, j \in S^i$. This is again linked to the earliest pick up time by $e_i^{\text{PickUp}} = l_i^{\text{PickUp}} - t_{\text{flex}}$. The earliest arrival at a dropoff node $j \in S^i$ is given by $e_j^i = e_i^{\text{PickUp}} + t_{ij},\ j \in S^i$. Finally, the set of all dropoff nodes is given by $\hat{S} = \bigcup_{i \in P, j \in S^i} \{s_j^i\}$.

Recalling that each vehicle may have a unique depot, let the set of depots be given by $d \in D, |D| = |K|$, with d^k representing the depot associated with vehicle $k \in K$. This gives a node set $N = D \cup P \cup \hat{S}$. For now, we assume that an arc exists between every pair (i, j) of nodes in N, except for arcs that allow vehicles to visit incompatible depots. For convenience, we introduce the set $S_j = \bigcup_{i \in P} \{s_j^i\}$ which includes all drop off nodes representing the physical station $j \in S$. Naturally, $c_{ij} = t_{ij} = 0$ if $i, j \in S_k, k \in S$).

This formulation can still be used for the traditional, zone based DRC, by setting $|S| = \left|S^i\right| = 1, i \in P$ (as there is only a single station for each passenger) and $\left|\hat{S}\right| = |P|$ (as there is only a single drop off node for each passenger).

We introduce the following decision variables. Let x_{ij}^k indicate whether vehicle $k \in K$ traverses arc $(ij) \in A$ ($x_{ij}^k = 1$) or not ($x_{ij}^k = 0$). Let T_i be the time that a vehicle departs from node $i \in \mathbf{N}$. Finally, let Q_i give the number of passengers in a vehicle when it departs from node $i \in \mathbf{N}$. Below, we present an integer programming formulation for the optimal scheduling of an Extended Demand Responsive Connector:

Integer Program 7:

$$\min \quad \sum_{i \in N} \sum_{j \in N} \sum_{k \in K} c_{ij} x_{ij}^k \tag{3.1}$$

$$\text{s.t.} \quad \sum_{j \in N} \sum_{k \in K} x_{ji}^k = 1, \qquad i \in P \tag{3.2}$$

$$\sum_{i \in P} x_{d^k,i}^k \leq 1, \qquad k \in K \tag{3.3}$$

$$\sum_{j \in N} x_{ji}^k - \sum_{j \in N} x_{ij}^k = 0, i \in N \setminus D, \quad k \in K \tag{3.4}$$

$$\sum_{j \in N} x_{ji}^k = \sum_{l \in S^i} \sum_{j \in N \setminus D} x_{jl}^k, \qquad i \in P, k \in K \tag{3.5}$$

$$T_j \geq \sum_{k \in K} x_{ij}^k \left(T_i + t_{ij}\right), \qquad j \in N \setminus D, i \in N \tag{3.6}$$

$$e_i^{\text{PickUp}} \leq T_i, \qquad i \in P \tag{3.7}$$

$$T_j \leq l_j^i, \qquad i \in P, j \in S^i \tag{3.8}$$

$$T_j \geq T_i, \qquad i \in P, j \in S^i \tag{3.9}$$

$$T_j - T_i \leq r_i, \qquad i \in P, j \in S^i \tag{3.10}$$

$$Q_j = 0, \qquad j \in D \cup \hat{S} \tag{3.11}$$

$$Q_i = \sum_{j \in N} \sum_{k \in K} x_{ji}^k (Q_j + q_i), \qquad i \in P \tag{3.12}$$

$$Q_i \leq Q, \qquad i \in P \tag{3.13}$$

$$x_{ij}^k \in \{0, 1\}, \qquad i, j \in N, k \in K \tag{3.14}$$

$$T_i \geq 0, \qquad i \in N \tag{3.15}$$

$$Q_i \geq 0, \qquad i \in N \tag{3.16}$$

The objective function indicates that the goal is to minimize the cost of running the service vehicles. Constraints (3.2) ensure that all passengers are picked up. Constraints (3.3) and (3.4) ensure that vehicles start and end at the depot and depart from every location they visit in between. Constraints (3.5) ensure that the vehicle that picks up a passenger

also drops off that passenger. Constraints (3.6) ensure the consistency of the departure times along the route of a vehicle. Constraints (3.7) ensure that passengers are not picked up earlier than is allowed and constraints (3.8) ensure that passengers arrive at a dropoff node in time to catch a suitable service to their final destination. Constraints (3.9) ensures that passengers are picked up before they are dropped off. Constraints (3.10) enforces the ride time limit for each passenger. Constraints (3.11) ensure that a vehicle has no passengers on board when it departs from the depot and that when a vehicle arrives at a transit station all passengers that are on board are dropped off. Although the latter implies that passengers picked up at node $i \in P$ may be dropped off at a node $l \in S_j$ for some $j \in S$ that is not in S^i, the objective function ensures that the vehicle will visit the corresponding node in $S^i \cap S_j$ before visiting any other pickup nodes or returning to the depot. Constraints (3.12) ensure that the number of passengers in a vehicle is updated correctly at each pickup location and constraints (3.13) ensures that the number of passengers in a vehicle never exceeds the vehicle capacity. Finally, constraints (3.14), (3.15), and (3.16) set the variable types and bounds. Clearly, constraints (3.6) and (3.12) are nonlinear. However, they can easily be linearized as shown below:

$$T_j \geq T_i + t_{ij} - M(1 - \sum_{k \in K} x_{ij}^k), \qquad j \in N \setminus \{0\}, i \in N \tag{3.6'}$$

$$Q_i \leq Q_j + q_i + M(1 - \sum_{k \in K} x_{ji}^k), \qquad i \in P, j \in N \tag{3.12'}$$

$$Q_i \geq Q_j + q_i - M(1 - \sum_{k \in K} x_{ji}^k), \qquad i \in P, j \in N \tag{3.12''}$$

Integer Program 7 has a large amount of symmetry, which is well-known to cause difficulties when solving using traditional linear programming based branch and bound algorithms. The symmetry is caused by the presence of multiple nodes representing a single transit station $j \in S$. Specifically, a vehicle visiting several nodes in S_j can do so in any order without affecting the cost of the solution (since the cost to travel between the nodes is zero). Figure 3.4 shows an example, where black, red, and blue arcs show three equivalent ways in which to traverse the transit nodes. However, every permutation represents a distinct point in the solution space and a corresponding leaf node in the branch and bound tree that must be discovered.

To avoid this symmetry, for each set S_j, $j \in S$ we create two *gate* nodes, g_j^{in} and g_j^{out}, and redirect all arcs entering or leaving S_j through these gates. We then enforce an arbitrary order in which the nodes of S_j must be visited, preventing alternative solutions (an example is shown in Figure 3.5). Here, transit nodes are ordered from left to right i.e., vehicles may move from h' to any other transit node, and from i' to j', but no other move is permitted. It should be clear that with this modification, there is a unique way to visit a subset of S_j, removing the symmetry. Except for the aforementioned gates, nodes in S_j are not connected to any nodes outside of S_j. The ordering can still be imposed without the gate nodes, but their absence would necessitate the presence of an arc between every passenger node and every transit node, corresponding to a large increase in the number of variables.

We are able to reduce the number of variables in a particular instance. No arc is created

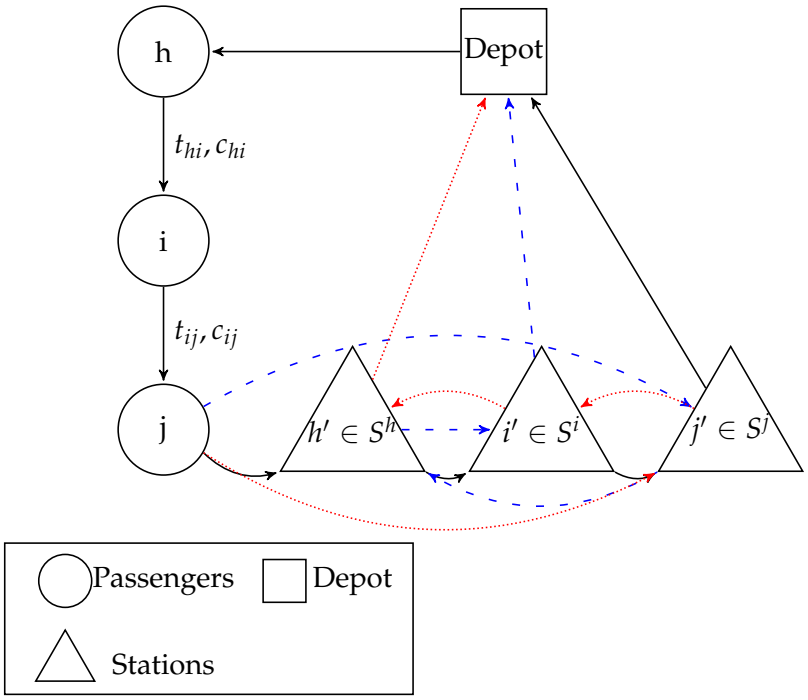


Figure 3.4: Multiple equivalent solutions

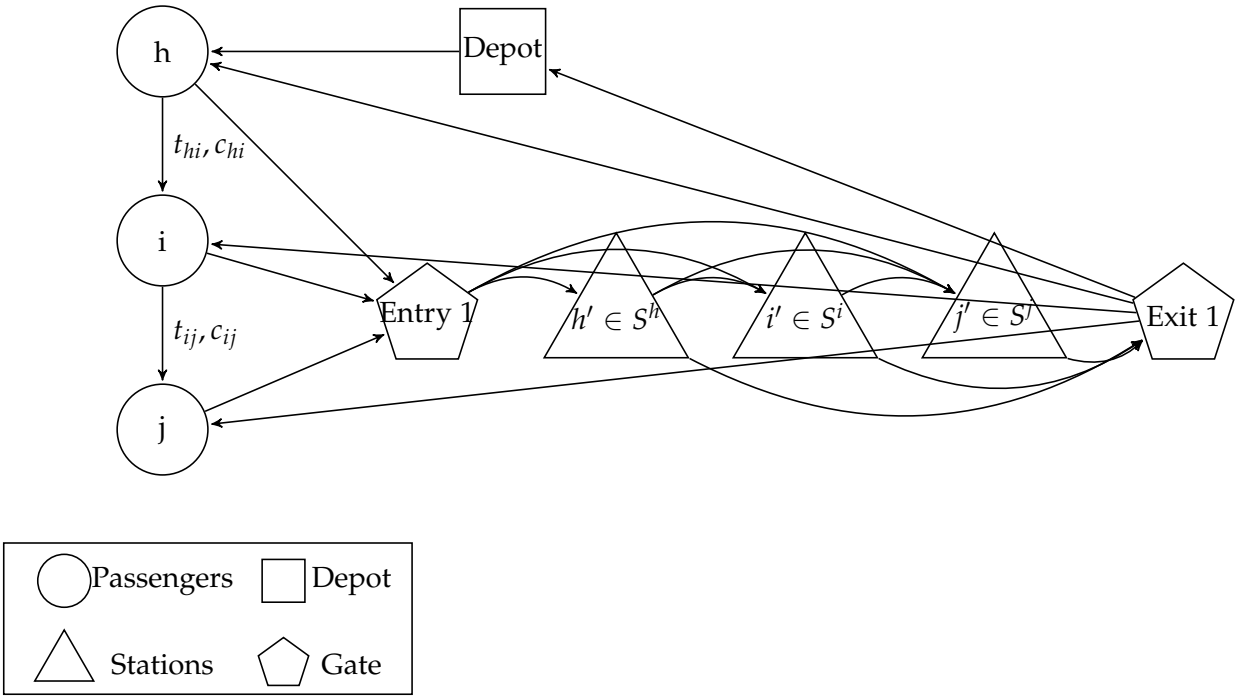


Figure 3.5: Alternative network representation

between nodes $i,\ j \in P$ if the passengers cannot be picked up together because of incompatible time windows. A passenger node i is only connected to an entry gate node if the latter is connected to a drop off node for passenger i. Similarly, an exit node is only connected to pick up node i if it is feasible to drop off passengers for the associated service and then pick up passenger i. Finally, the only arcs leaving the depot are those going to passenger nodes, and the only arcs entering the depot are those from exit gates.

3.4 A Heuristic Algorithm

As mentioned above, the complexity of this problem makes it impossible to exactly solve any instance of a realistic size. For this reason, we present an effective Variable Neighborhood Search meta–heuristic capable of finding high quality solutions (the exact details are given below). We apply it to both the flexible and extended settings, to ensure that benefits observed in the extended system arise from the extra flexibility (and are not the result of different solution methods). The process is repeated five times, with new parameters and a different order for exploring solutions within a neighborhood (for an overview of such ideas, common in multi-start heuristics, we direct the reader to Martí et al. (2010)). Naturally, the best solution found over all five repeats is returned.

3.4.1 Solution Construction

To form an initial solution, we use the sequential construction heuristic outlined in Algorithm 1, which is a slight extension of the Nearest Neighbor Insertion heuristic from Section 2.3.3.1. Initially, a schedule contains a seed passenger, the station closest to this passenger, and the depot locations associated with the vehicle (see Section 3.4.1.1). The schedule is then grown by inserting further pick ups and stations into the schedule (Section 3.4.1.2). When no further passengers can be inserted into a vehicle schedule, we open a schedule for a new vehicle (and continue doing so until all passengers are served). We note that for any passenger in a partially constructed schedule, there is a set of stations at which the passenger can be dropped off and that as further passengers are inserted into the schedule, this set of stations may shrink. Thus, prematurely committing to dropping passengers off at a particular station can make later insertions infeasible, or result in schedules with sub-optimal costs. For this reason, we introduce the concept of *pseudo-schedules*, which allows stations visited in a schedule to be altered as the schedule grows (Section 3.4.2).

3.4.1.1 Seed Customer Selection

The selection of a seed passenger is controlled by a parameter $p,\ 0 < p < 1$ and involves the set of passengers $\hat{P}$ not yet assigned to a route. We order all such passengers ($i \in \hat{P}$) by the earliest time they can be picked up and select the passenger in position $\lceil p * |\hat{P}| \rceil$ as the seed passenger. A different value of p is used for each of the five repeats, specifically we use the values $p = 0.1, 0.3, 0.5, 0.7$, and 0.9.

Algorithm 1: Solution Construction Method

```
Input: Empty set of vehicle schedules V, Set of passengers P
Initialize: P̂ ← P
repeat
    InitializeEmptyVehicleSchedule(v)
    FindSeedCustomer(P̂, v)
    repeat
        FindFeasibleInsertion(P̂, v, foundInsert, c1)
        FindFeasibleInsertionWithStation(P̂, v, foundInsertWithStat, c2)
        if foundInsert or foundInsertWithStat then
            InsertMinimumCostCandidate(v, c1, c2)
            UpdateRemainingPassengers(P̂, c1, c2)
    until !(foundInsert or foundInsertWithStat)
    AppendVehicleSchedule(V, v)
until |P̂| = 0
return V
```

3.4.1.2 Insertion of customers

To grow a partial schedule, we insert further passengers and stations into the schedule. Specifically, the schedule grows in one of two ways:

- **Insertion of a passenger:** For each remaining passenger in $\hat{P}$, we attempt to insert this passenger feasibly into every position in the schedule, and, if the insertion is feasible, the resulting increase in cost is calculated.
- **Insertion of a passenger and a station:** For each remaining passenger in $\hat{P}$, we find the closest station. We then attempt to insert this passenger – station pair into every position in the existing schedule, and, if the insertion is feasible, calculate the resulting increase in cost. If the insertion has the vehicle visiting two stations in a row, the cost is calculated as if the second station (the one originally in the schedule) is not visited. If an insertion for which this occurs is accepted, then the second station is removed.

When calculating the cost of the insertion, there are two relevant concerns – the increase in distance the vehicle must travel (again, our objective is to minimize total distance traveled), and the time the vehicle must wait for the passenger to become available (as time spent waiting makes it difficult for the vehicle to serve other passengers). To balance these two measures, the cost of an insertion is the linear weighted combination of these two. This is governed by a parameter, w, which gives the ratio of $\frac{\text{Weight given to increase in travel distance}}{\text{Weight given to increase in waiting time}}$. The (feasible) insertion with the lowest cost is chosen. If no feasible addition can be found, and there are still passengers remaining, we schedule another vehicle. There is no limit on the number of vehicles available in the construction phase, as later improvements can reduce this number. After a small pilot study, we found the best approach was to set $w = 1.25$

Note that there is no randomness in this construction routine. However, Section 3.4.3.2 explains how the initial solution can be perturbed in a way that allows the meta–heuristic to restart from the set of routes, yet explore different regions in the solution space.

3.4.2 Pseudo–Schedules

When exploring our improvement routines (see Section 3.4.3), it is easy to imagine that the effect a particular move has on the cost of a schedule depends on what stations are visited. This means by considering the proposed move while simultaneously allowing changes to the stations visited, we could discover improvements we would have otherwise missed. For this reason, we introduce a concept called *pseudo–schedules*. To determine if a particular move should be made, we first determine the change in objective value (generally computationally inexpensive). If the move would improve the objective value, we then carry out a more in-depth investigation using pseudo-schedules. Given an existing schedule, we convert it to a *pseudo–schedule* by replacing each station with a *pseudo–station*, or a Restricted Candidate List (RCL) containing stations that can be used at that point in the schedule. When an improving move is found in the Variable Neighborhood Search procedure described in Section 3.4.3, we try all schedules implied by the combination of all RCLs.

To generate an RCL, we construct a box around the station currently visited in the schedule, denoted by s, and add all stations inside this box to the list. More specifically, we have two parameters, δx and δy and take the two locations immediately preceding and following the station, l^1 and l^2, (these will usually be customers, but could also be a depot). We take the first location, l^1, and shift it slightly. More specifically, if it has a smaller (larger) x coordinate than l^2, we shift it by $-\delta x$ (δx); similarly, if it has a smaller (larger) y coordinate than l^2, we shift it by $-\delta y$ (δy). The same procedure is applied to the second location, l^2. The two newly perturbed points define a rectangle and all stations inside this box are added to the RCL (as shown in Figure 3.6).

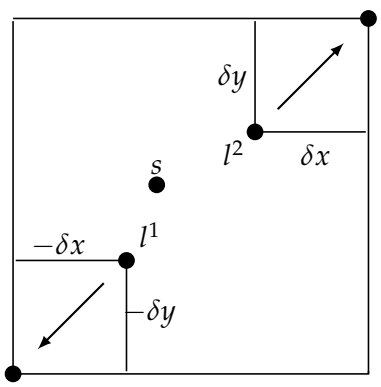


Figure 3.6: Box obtained from locations l^1 and l^2 and used to define the RCL

3.4.3 Variable Neighborhood Search

We will now describe the Variable Neighborhood Search (VNS) scheme used to improve the initial solution. We order our neighborhoods and explore them in sequence. When exploring a given neighborhood, we always accept the first feasible improvement found, and immediately revert back to the first neighborhood. Within a given neighborhood, moves are always trialled in the same order. The algorithm stops when the shaking step occurs twice without an improvement in the best solution. An outline of the VNS scheme is given in Algorithm 2, and a description of the neighborhoods used is given below.

Overview of Neighborhoods in Algorithm 2

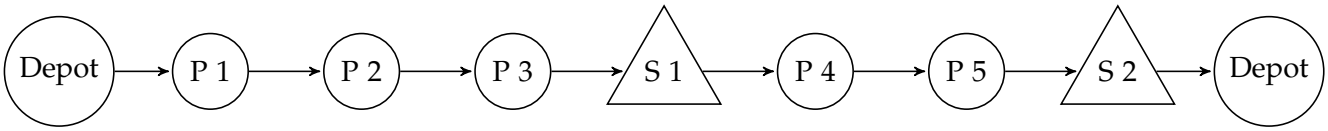
1. **Transfer passenger within vehicle:** We attempt to move a passenger to another position within the same route (Figure 3.7(b)).
2. **Transfer passenger and station within vehicle:** We remove a passenger, p from a given route and attempt their re-insertion (along with their closest station, $S(p)$), into a different position in the schedule of the same vehicle (Figure 3.7(c)).
3. **Swap passengers within vehicle:** We attempt to swap the position of two passengers in the same route (Figure 3.7(d)).
4. **Transfer passenger between vehicles:** We attempt to move a passenger to a new position in a different route (Figure 3.8(b)).
5. **Transfer passenger and station between vehicles:** We select a passenger, p and attempt their insertion (and that of their closest station, $S(p)$), into the route of a different vehicle (Figure 3.8(c)).
6. **Swap passengers between vehicles:** We attempt to swap the position of two passengers served by different vehicles (Figure 3.8(d)).
7. **Change stations:** For each station in every vehicle schedule, we trial their replacement with every other station.

For clarity, the use of specific neighborhoods in Algorithm 2 is controlled by the variable k, e.g., when $k = 1$, the first neighborhood given above is used. After some neighborhoods (specifically neighborhoods one, two, four and five), it is possible that there exists two stations in a row (as discussed in Section 3.4.1.2). If this happens, the second station is removed. Similarly, following neighborhoods four and five, it is possible that a route does not visit any passengers e.g., it only contains stations and the depot; in this case, we remove the vehicle completely.

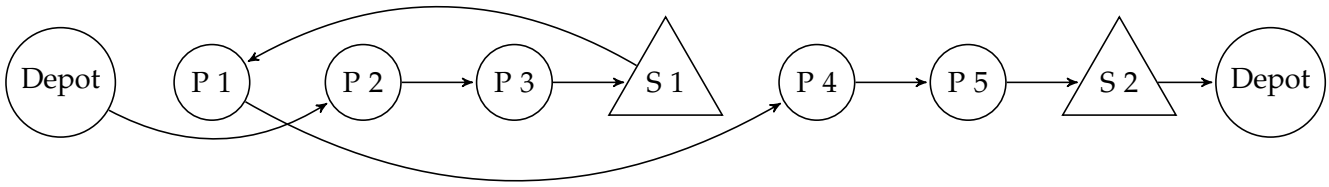
3.4.3.1 Shaking Step

We also include a *shaking step* in our algorithm, which attempts to remove a route by redistributing its constituent passengers. This is partially done to introduce diversity into the search, but also because fewer vehicles can improve the objective by removing trips to and from the depot. We pick the vehicle serving the smallest number of passengers (breaking ties arbitrarily) and, considering each passenger in turn, attempt to feasibly reinsert them into the schedules of all other vehicles at all positions. We consider all feasible

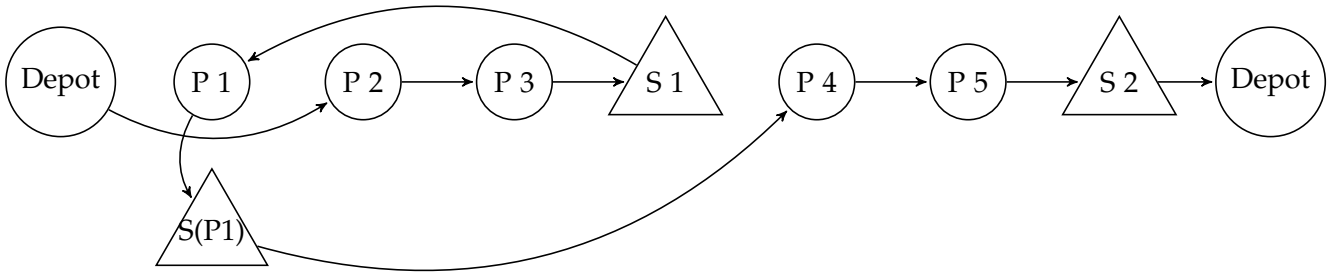
(a) Original route



(b) Transferring a passenger within a route



(c) Transferring a passenger and a station within a route



(d) Swapping passengers within a route

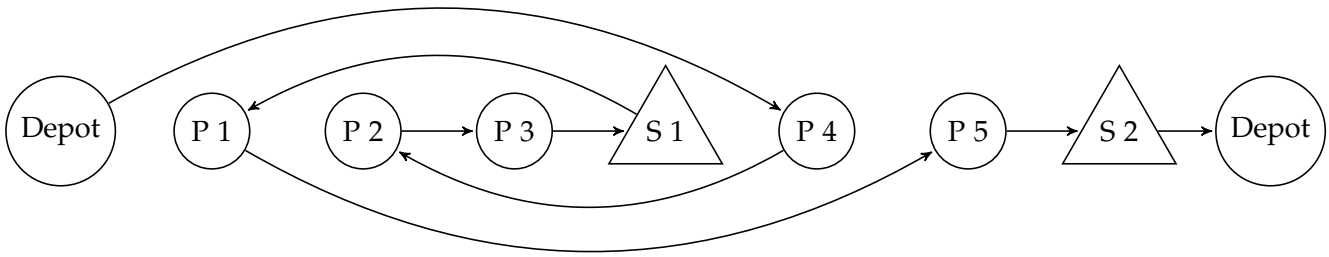


Figure 3.7: Moving passengers within a route

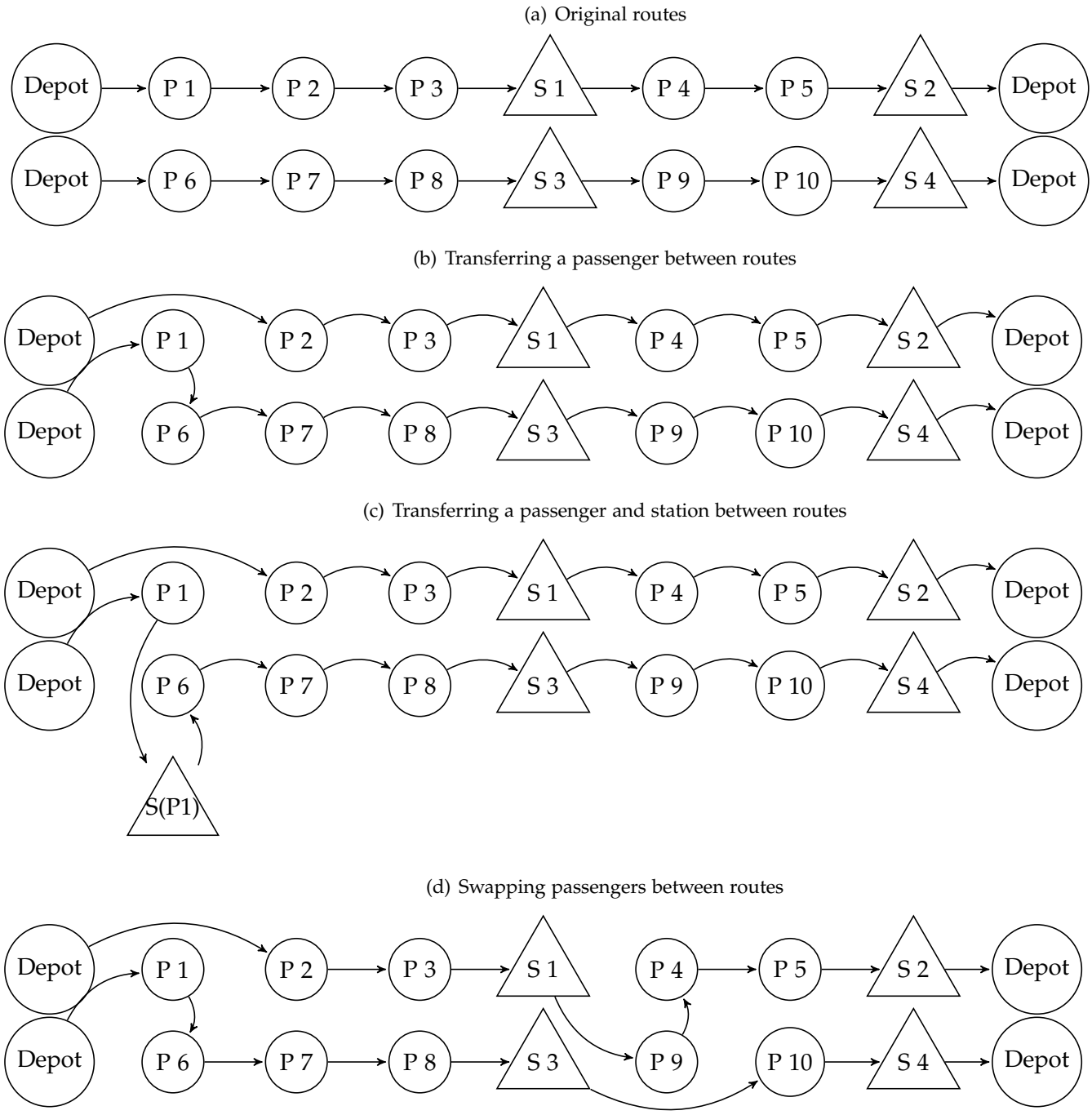


Figure 3.8: Moving passengers between routes

Algorithm 2: Variable Neighborhood Search

```
Input: A feasible initial solution x'
Initialize: noImproveCounter ← 0, x ← x'
repeat
    k ← 1
    repeat
        foundImprov ← ExploreNeighborhood (k, x, x*)
        if foundImprov = TRUE then
            k ← 1
            x ← x*
        else
            k ← k + 1
    until k = 8
    if CostOfSolution(x) < CostOfSolution(x') then
        noImproveCounter ← 0
        x' ← x
    else
        noImproveCounter ← noImproveCounter + 1
    if noImprovCounter < 2 then
        x ← ShakeSolution (x)
until noImproveCounter = 2
return x'
```

insertions, even those which worsen the objective function, and ultimately chose the one which gives the smallest increase. If we cannot find a feasible insertion for a passenger, we move onto the next passenger in the schedule. If we manage to re-insert at least one passenger, we revert back to the first neighborhood (and we remove the vehicle if it no longer contains any passengers). We note that as the inner neighborhoods concern intra–route movements, and as the first improvement is always selected, it is highly unlikely the moves performed in this step will be directly reversed. The heuristic terminates when the shaking step has occurred twice without improving the best known solution.

3.4.3.2 Repeats

Every time the heuristic restarts, it does so using the same initial solution generated in Section 3.4.1, but the sequence in which routes are stored is randomized. This in turn changes the order in which neighboring solutions are found by the improvement routines, and as the first improving move is always accepted, we can hope to find different local optima. This allows us to introduce diversity and randomness into the search, while still using the same, high quality initial solution.

3.4.4 Slack Time Adjustment

Once we have selected the best solution, we perform a small amount of post–processing to minimize the inconvenience faced by customers and maximize the perceived quality of service. We assume that given the choice, customers want to be picked up as late as possible (provided they still make their train), as they prefer to wait in their own house than at a public station. To accommodate this, we simply make all pick up times as late as they can be, without violating any time windows.

3.5 Instance Generation

We will next describe how instances for our computational study are generated. We consider a square geographical region with a length l of 25 kilometers, bisected by a single train line with equidistant stations. All vehicles share a single depot midway on the left boundary, and travel at a constant speed of 30 kilometers per hour (with negligible time spent collecting passengers). We then generate passengers with origins uniformly distributed throughout the region, and assume, without loss of generality, they all wish to travel to the same station 10 kilometers outside the region of interest. While this last point may seem limiting, we note that this train will pass through other stations prior to the assumed one, and that passengers could equally depart at these stations at the corresponding times. The network is given in Figure 3.9, with the zones in the zonal system being shown by the dashed lines.

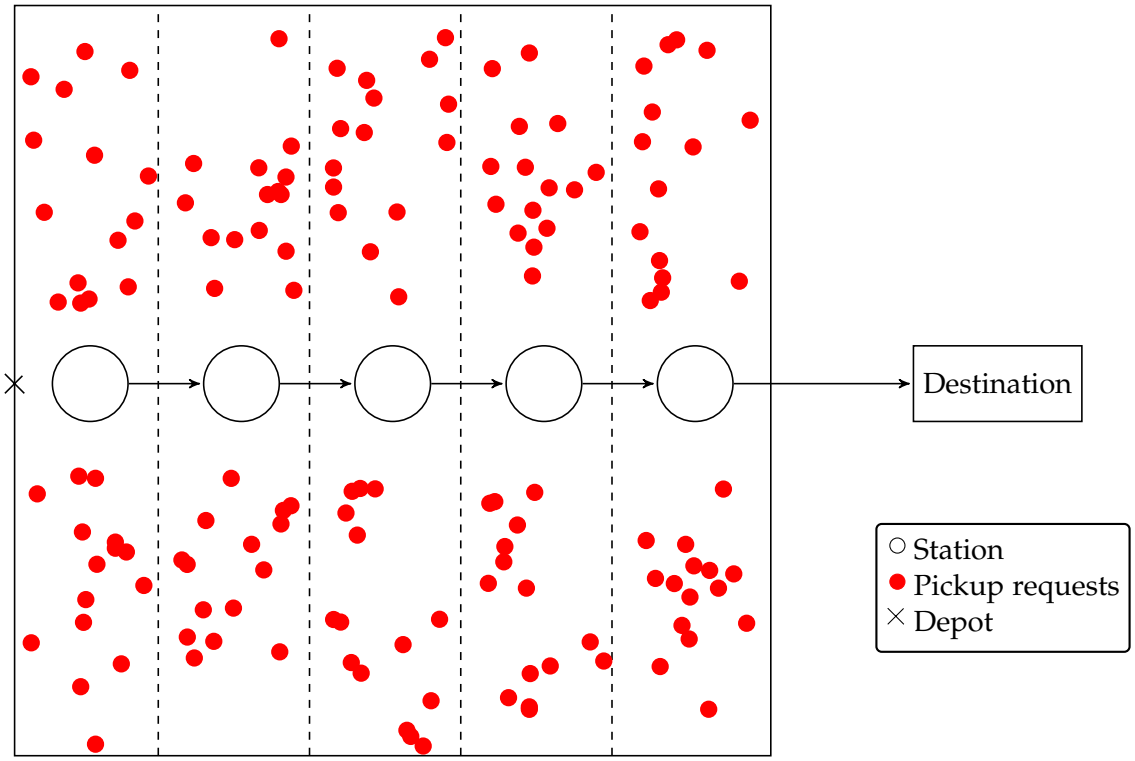


Figure 3.9: Example of how an EDRC operates

We assume the desired latest arrival times of passengers at their downstream station are dispersed over a one hour period. Specifically, desired arrival times at the destination station are generated using a truncated normal distribution with a mean of 30 minutes and a standard deviation of 10 minutes (McGuckin and Srinivasan, 2003). Times outside of this hour period (i.e. more than 3 standard deviations from the mean) are discarded (and a new one is generated). These desired arrival times (at the destination station) are then mapped to a preferred train service, giving the latest train each passenger can take. As an example, suppose we have a train arriving at the destination station at either end and the middle of our 60 minute period of interest. Figure 3.10 shows that if a passenger's desired arrival time falls within a shaded region, they are mapped to the first or third train service, otherwise they are assigned to the second service.

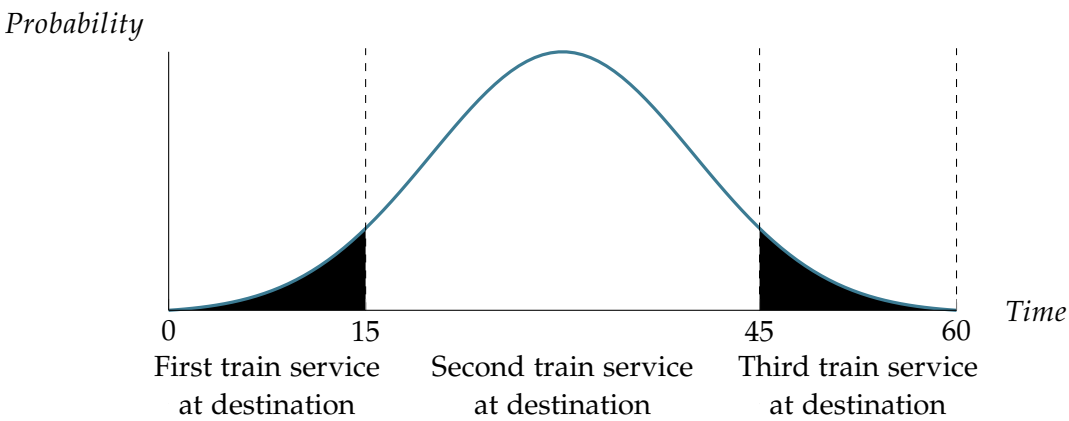


Figure 3.10: Mapping arrival times to train services

We construct a regular train timetable, assuming a constant speed across the network and negligible time spent at stations. A train always arrives at the destination station at the end of the period of interest, and the time between successive arrivals is described by the relevant parameter in Table 3.1. This must be done before calculating passengers' earliest pick up times, as described in Section 3.2.1.

A class of instances is defined by a number of parameters, whose definitions and values used in the computational study are given in Table 3.1. Taken together, the combination of these parameters give a total of 360 instances. We note that the set of passenger locations only changes when the parameter "number of passengers" changes, and of course, for the individual instances within a class. That is, when we compare results for a different number of stations, the set of passenger locations is the same. Additionally, when we do generate new customers, we maintain the existing set, i.e., the passenger set of smaller instances are a strict subset of those in larger instances. As passengers' time windows are not an (explicit) input (but depend on other parameters), these do change.

Table 3.1: Parameters defining classes of instances

Parameter	Definition	Values used
Number of passengers:	The number of passengers to be picked up.	50, 100, 150, and 200 passengers
Train Frequency:	The time between successive train services.	30 and 60 minutes
Number of Stations:	The number of train stations in the region.	3 and 5 stations
Service level:	The size of the flex window.	10, 15, and 20 minutes
Vehicle Capacity:	The capacity of the (homogeneous) vehicles.	5 and 10 passengers
Number of Instances:	The number of instances in an instance class.	5 instances

3.6 Computational Study

3.6.1 Heuristic Validation

To be able to meaningfully compare the performance of a zonal and an extended demand responsive connector, we need to be confident that our heuristic performs well. To demonstrate this, we investigated its performance on instances small enough to be solved optimally with the IP formulation presented in Section 3.3. These instances had either five or ten passengers, with all other parameters as described in Table 3.1 (giving a total of 240 instances). The maximum number of vehicles available is set to one more than the number needed in the best heuristic solution. Although this restriction can theoretically remove the optimal solution (if it uses at least two more vehicles than the heuristic solution), we feel the existence of such a scenario is highly unlikely. Additionally, the tighter formulation reduces the solution space, accelerating the solution process – an effect we feel is much more significant. The results can be found in Tables 3.2 and 3.3.

Table 3.2: Difference between optimal and heuristic solution with five passengers

Average difference in objective value	0.077%
Average difference in number of vehicles used	0 vehicles
Maximum difference in objective value	2.58%
Number of instances where optimality was not achieved	11
Number of instances where optimality was achieved	229

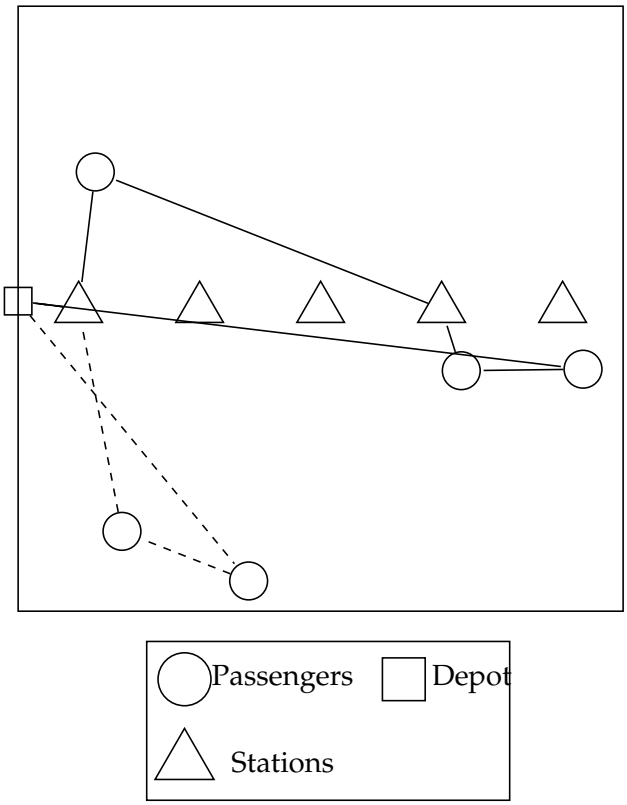
As we can see, the heuristic performs well on instances with five passengers, with the average difference in objective value being very small at 0.077%, and optimality being obtained in the vast majority of instances. Additionally, the heuristic and the optimal solution always use the same number of vehicles. The maximum observed difference between the two solutions was 2.58%; to try and understand this, the optimal and heuristic solution to this instance are given in Figure 3.11.

We observe that to move from the heuristic solution to the optimal solution, we would need to swap the order in which the second vehicle picks up its two passengers, while simultaneously changing the station at which they are dropped off. The **Swap within vehicle** routine does attempt to swap the relevant passengers, but as this alone does not improve the objective value, the use of a different station is not investigated and the optimal solution is not discovered.

Analyzing the results for instances with ten passengers is more difficult as only a few of the instances were solved to optimality by the IP solver (in the given time limit of one hour). Regardless of the result, we split the instances into three classes: where the IP solver found a better solution, where the heuristic found a better solution and where the two approaches found the same solution.

In approximately a third of the instances, the IP solver outperformed the heuristic, finding a solution which was on average, 2.17% better. In about a fifth of the instances, the heuristic found a solution which was (noticeably) better, with an average difference of 6.15%. In both cases, the number of vehicles used was about the same.

(a) Optimal solution



(b) Heuristic solution

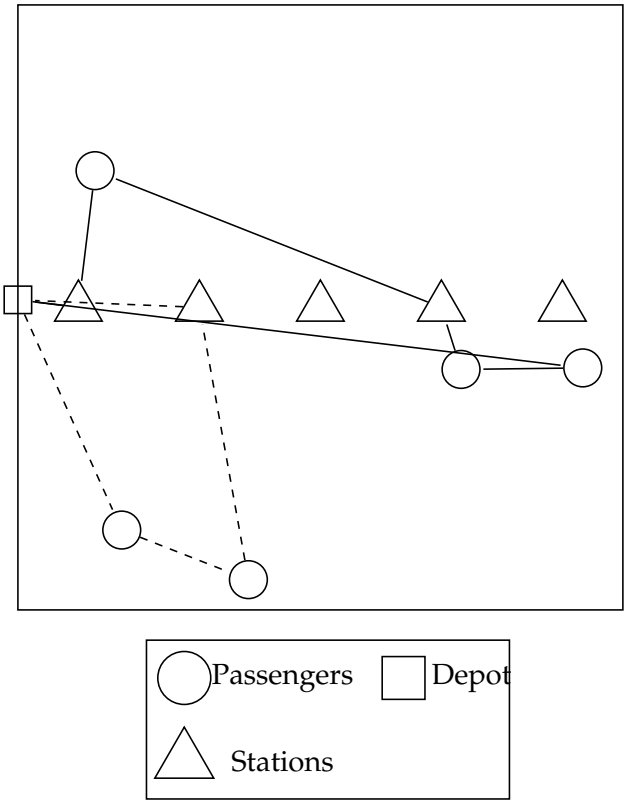


Figure 3.11: Optimal and heuristic solution for instance with biggest optimality gap

Table 3.3: Difference between optimal and heuristic solution with ten passengers

	Instances where the IP had a better objective value	Instances where the heuristic had a better objective value	Instances where the heuristic had the same objective value
Average difference in objective value	2.17%	6.15%	0%
Average difference in number of vehicles used	0.103	0.130	0
Maximum difference in objective value	8.89%	28.31%	0
Number of instances	78	46	116
Number of instances with proven optimality	5	0	2
Average run time of integer program	3243 seconds	3600 seconds	3000
Average run time of heuristic	0.09 seconds	0.10 seconds	0.09 seconds

To further understand the difference in performance, we randomly selected five instances where the heuristic outperformed the IP solver, and five instances where the IP solver outperformed the heuristic but did not prove optimality of the solution found, and ran these instances with a maximum solve time of four hours. In the former five instances, the IP solver found the better heuristic solution four out of five times (while there was no change in the fifth instance). In the latter set, the solution found by the IP solver never improved.

We believe that these computational experiments show that we can be confident that the heuristic approach produces good quality solutions and that using it to analyze the benefits of an extended demand responsive connector over a zonal demand responsive connector is appropriate.

Before investigating the benefits of our extended DRC variant, we present the results of computational experiments that assess the benefits of using pseudo-schedules in the neighborhood search. The fact that the set of stations feasible for a passenger depends on other passengers in the vehicle (and thus changes dynamically during the neighborhood search) is one of the more interesting and computationally challenging features of our problem. Recall that in a *pseudo–schedule* a pseudo–station is "placeholder" for some station, but the actual station has not been determined. Table 3.4 shows the difference in quality of the solutions obtained by the heuristic with and without the use of pseudo-schedules, where we have averaged the differences over all instances with the specified number of requests.

3.6.2 Results

To investigate the benefits of an extended demand responsive connector, we define a number of statistics to analyze the solutions produced by each instance class; (for a full list, see

Table 3.4: Improvement resulting from the use of pseudo–schedules

Number of Requests	Average percentage improvement
50	5.14%
100	6.74%
150	7.07%
200	7.65%

Table 3.5). The statistics are chosen in such a way that we can measure the systems' performance from the perspective of both the service provider and a passenger.

Table 3.5: Solution statistics

Statistic	Definition
Of interest to the service provider	
Cost:	The sum of the (Euclidean) distance traveled by each of the service vehicles.
Empty:Loaded Ratio:	The fraction of a vehicle schedule in which it travels empty (averaged over all vehicles).
Vehicles Used:	The number of vehicles used.
Of interest to the passenger	
Pick Up Inconvenience:	Length of time between the time a passenger is picked up and his latest possible pick up time (averaged over all passengers).
Station Inconvenience:	Length of time a passenger spends waiting at a station for the train to arrive (average over all passengers).
Destination Inconvenience:	Length of time between the time a passenger arrives at his destination and his desired arrival time at the destination (averaged over all passengers).

We note that because the minimum time between train services (either 30 or 60 minutes) is greater than the width of the flex window, it is not possible for passengers to arrive at the destination early – hence the value of *Destination Inconvenience* is always zero and we exclude it from further consideration. As discussed in Section 3.4.4, we perform a post–processing step that improves service quality by minimizing the sum of *Pick Up Inconvenience* and *Station Inconvenience*. Because of this, the amount of time spent waiting at stations is less than a minute in all solutions – so we also exclude this from further consideration.

The results of the computational study show that three parameters, namely the number of passengers, the number of stations and the size of the flex window, have the most significant impact on the savings resulting from an extended DRC (versus a zone based variant). This is illustrated in Table 3.6, which gives the cost savings for various values of these parameters (averaged across all instance classes and all instances within a class).

We see that as passenger density increases, the savings from the more flexible EDRC increase as well. This suggests that even at higher levels of demand, decentralization of the

Table 3.6: Benefits of the EDRC over the traditional variant

Number of Requests	Cost Savings
50	23.30%
100	24.33%
150	26.44%
200	28.68%
Number of Stations	
3	22.29%
5	27.35%
Size of Flex Window	
10	22.37%
15	25.60%
20	26.61%

service can still give cost savings. We see too that as the number of stations increases, the savings from the EDRC increase. Again, this is not unexpected, and is the result of the additional opportunities for passenger drop offs, and the smaller zones in the region setting. Finally, we see that a larger flex window (which lets more passengers be included in a route), achieves greater savings. These results demonstrate that the significant potential of the EDRC, and that the savings are greatest in environments where demand density is low (in terms of geography and time), or when the circumstances inherently provide flexibility to the service provider (greater number of stations and greater flex windows). We repeat that because of the way the solutions are generated, i.e., by using the same construction and neighborhood search scheme for both systems, we argue that the benefits observed are indeed the result of the increased flexibility.

Next, we investigate whether these economical benefits impact the quality of service experienced by the passenger, as measured by the *Pick Up Inconvenience*. The results can be found in Table 3.7. It is reasonable to assume that passengers prefer leaving their residence as late as possible. We note there is an argument that people may prefer to be picked up earlier if they could spend most of their time waiting at the station, but even if we removed the post–processing procedure from Section 3.4.4, this didn't seem to occur. Hence, *Pick Up Inconvenience* can be viewed as the amount of extra time passengers spend traveling compared to if they drove themselves. This value is slightly higher in the EDRC than the traditional version, especially with more stations or a larger flex window, indicating that the cost savings come at a (small and arguably reasonable) price. It is interesting to note that with an EDRC, more stations results in longer passenger travel times due to the increased ability to combine vehicle schedules, while the reverse is true under the zonal case.

In addition to the total cost, a service provider would also be interested in the number of vehicles required and the amount of "empty driving" in the routes, as given in Table 3.8. Although the fleet of vehicles is not alterable in the short term (so is not considered in this daily planning problem), it still represents a significant portion of the overall cost incurred. We can see that the EDRC offers a significant advantage in this respect, meaning a new operator could purchase fewer vehicles (or an existing operator could downsize their

Table 3.7: Indicators of service quality for passengers

Number of Requests	Pick Up Inconvenience (minutes)	
	Extended	Zonal
50	6.39	5.37
100	5.70	3.72
150	5.88	3.69
200	5.94	3.67
Number of Stations	**Pick Up Inconvenience (minutes)**	
	Extended	Zonal
3	5.45	3.75
5	6.19	3.54
Flex Window Width	**Pick Up Inconvenience (minutes)**	
	Extended	Zonal
10	3.47	2.43
15	5.80	3.64
20	8.18	4.86

fleet). Additionally, we see a notable reduction in the *Empty:Loaded Ratio*, especially when passenger density is low.

Table 3.8: Indicators of interest to the service provider

Number of Requests	No. Vehicles		Empty–Loaded Ratio	
	Extended	Zonal	Extended	Zonal
50	9.27	12.35	0.32	0.38
100	14.50	20.48	0.26	0.29
150	19.51	28.55	0.24	0.26
200	24.14	36.05	0.23	0.24
Number of Stations	**No. Vehicles**		**Empty–Loaded Ratio**	
	Extended	Zonal	Extended	Zonal
3	17.11	24.51	0.26	0.29
5	16.60	24.21	0.26	0.30
Extended Time	**No. Vehicles**		**Empty–Loaded Ratio**	
	Extended	Zonal	Flex	Zonal
10	18.70	26.25	0.28	0.31
15	16.67	24.21	0.26	0.29
20	15.19	22.62	0.25	0.28

Finally, we analyze the impact the number of stations has on the total cost; specifically, Table 3.9 shows the cost savings obtained by increasing the number of stations from 3 to 5, under a range of settings. Interestingly, we see no clear pattern. The reason, most likely, is the delicate interaction between two dueling effects. First, as the number of stations increases, the operator has a greater range of dropoff locations, so we might expect the cost to fall. However, a passenger's latest pick up time depends on their distance to a station, so more stations can allow less flexibility. This indicates that defining and calibrating service level offerings is a crucial, yet unintuitive task for a service provider.

Table 3.9: Savings made by increasing the number of stations under the EDRC

Number of Requests	Percentage Saving
50	4.80%
100	4.98%
150	4.37%
200	5.00%
Flex Time	
10	4.70%
15	5.19%
20	5.33%

3.7 Conclusion

We have described a promising transport system, the Demand Responsive Connector (DRC) and identified the operational cost as a key hurdle preventing greater use. To help mitigate this, we have proposed a more flexible and cost efficient variant, called the Extended Demand Responsive Connector (EDRC). We have demonstrated that our scheme offers substantial benefits compared to the traditional version, with the size of these being strongly dependent on environmental characteristics; most importantly the number of passengers, the number of stations in the area, and the size of the flex window. From a practical perspective, the lower cost is not associated with a significant drop in service quality. Our extended variant will allow more cities around the world to implement a DRC, which offers both private benefits (citizens having greater connectivity to the transit network) and societal benefits (through reduced vehicle congestion and emissions).

In order to solve the related scheduling problem, we had to overcome some complexities not seen in other VRP variants. Specifically, our formulation introduced time windows that are dependent on both the request and station under consideration, and may also change dynamically as routes are built or altered. To account for this, we developed and implemented novel mechanisms within our solution methods. For our Integer Programming formulation, we introduced *gate nodes* to remove symmetrical solutions and greatly reduce the number of variables needed. For our meta–heuristic, we presented *pseudo–stations*, which act as a place holder and avoids the heuristic committing prematurely to a specific station.

Chapter 4

Dynamic Ride Sharing with Dedicated Drivers

4.1 Introduction

Vehicular congestion and its associated ills are problems faced by cities around the world and the need for alternative transport schemes is well–documented. However, the task of weaning commuters off private vehicles has (for reasons discussed below), so far proved too great for many cities. In this section, we investigate and extend another promising flexible transport system, known as Dynamic Ride Sharing (DRS). First, we consider a variant involving round trip constraints (which guarantee a return journey), that was previously thought to require a general Integer Program and show that under minimal assumptions, it can be reformulated as the (polynomially solvable) Transshipment Problem. We then investigate a more traditional setting, but allow the service provider to have their own fleet of private vehicles to serve riders who would otherwise go unmatched and investigate the benefits, complexities, and costs of doing so. We explore the different objectives the service provider may have and discuss the desirability of behaviors they encourage. An extensive computational study demonstrates the potential benefits of these additional drivers and identifies environments in which they are most useful.

To perform these experiments, we present three effective solution methods designed to optimally (in some sense of the word) serve riders with these two different driver types. First, we present an Integer Programming (IP) formulation capable of solving medium sized instances under certain conditions. We also develop an efficient metaheuristic that finds high-quality solutions for large–scale instances under general conditions. We also suggest a potential Branch and Price formulation for solving large problems to optimality. Finally, we present powerful preprocessing techniques that (depending on the objective of the service provider) can significantly reduce the problem size. All three of these approaches decompose the problem into two parts and exploit the inherent structure in a novel and interesting way.

4.2 Motivation

Rising population levels in many cities world are causing, amongst other things, increasing traffic congestion. Although the need for a strong, mass transit system is obvious,

there are three common barriers to progress. First, expansions of traditional public transport programs typically require a re–prioritisation of existing road space or a large capital investment, both of which can be politically difficult, especially in a time of global economic misfortune. Secondly, commuters simply like the convenience, flexibility and privacy that a private vehicle offers. Finally, the city may be at an awkward "transition phase", with insufficient population to justify a high frequency, mass transit network, while simultaneously being too large for everyone to drive a private vehicle. One possible solution is the facilitation of a Dynamic Ridesharing System (DRS), given in Section 2.4.2, where participants with similar travel itineraries are paired together. Such schemes require no special infrastructure and simply use the existing road network. They seek to combine the convenience and flexibility of a private vehicle, with the cost effectiveness of public transit. Finally, such schemes may actually work best outside of dense cities; although the larger population base represents more potential users (and better matchings), these locations normally have greater initial congestion. Although ride sharing schemes have societal benefits (e.g., reduced congestion and emissions), many participants may be attracted more by the personal benefits (e.g., sharing expenses, or saving time by being able to use express or carpool lanes). Modern ridesharing schemes make use of a smartphone application which allows the transfer and communication of all relevant information (for a more elaborate discussion of dynamic ridesharing schemes, see Agatz et al. (2012a)).

Outside of academic study, there has been concerns raised regarding the reliability and professionalism of drivers, with critics (justifiably) noting the strict vehicle and license requirements governing other transport operators, e.g., taxi drivers. Such issues are certainly contentious with important implications, but require a legislative and policy based response, so will not be considered further in this thesis.

Assuming an environment suitable for ride–sharing exists, the long term success of these schemes depends on reaching a "critical mass" of participants. This is because participants who repeatedly announce a trip without finding a match are likely to give up and not use the service again. Even matched participants may not return if the pairing was of a low quality (involving a large detour and/or a small cost saving); we note that more participants would naturally give a larger pool of matchings to choose from. This is especially important in the start–up phase of a ridesharing service, when perceptions are formed (at least in part) from the experience of the original participants. This difficultly is magnified for new service providers and in areas where ridesharing is untried, because of the inherent uncertainty associated with unknown companies and new concepts. In short, attracting and satisfying a large number of participants is critical for initiating and sustaining a ridesharing system.

Additionally, even matched participants may have concerns about their counterpart's punctuality and reliability. To counteract this, service operators typically implement a feedback system where participants publicly rate each other (similar to Ebay), with poor performers being removed from the system. Indeed, there are many reports of ride share operators being more responsive to complaints than taxi services. We believe this will alleviate the concern of most users and will focus on the uncertainty around not finding a match.

While all participants want to find a match, the repercussions of not getting one are more severe for riders. An unmatched driver simply has no one to share the journey with, while

an unmatched rider (who can't otherwise access a vehicle) has to hurriedly find alternative means of transport, such as public transport (which might be too slow) or a taxi (which is likely to be expensive). Therefore, schemes designed to boost participation rates have to focus on reducing uncertainty for riders.

System reliability is especially important if the rider is making a return trip and doesn't want to become stranded after the first leg (as the uncertainty may mean they take private transport for the whole trip). For this reason, past work has included variants where riders are only offered an out–going trip if they can be simultaneously guaranteed a trip on the return leg (Agatz et al., 2011). Even without the presence of special drivers, this additional requirement destroys the natural integrality of this problem, making it difficult to solve. However, we show that under weak assumptions about an absence of a temporal overlap between trips, natural integrality can be preserved. Furthermore, riders who violate this assumption may still be included, but won't be guaranteed a return trip. A common real world setting exhibiting this property is the weekday commute in large cities, where riders want a trip in both directions and the trips occur in distinct, non–overlapping time periods, i.e., the morning and afternoon peaks.

Outside rush hour, ride sharing schemes can still be very useful. To improve confidence, the service provider may consider employing a small number of professional drivers to satisfy rider requests that would otherwise remain unmatched (for clarity, these drivers are referred to as *dedicated drivers*, and the other drivers are called *ad hoc drivers*). It is possible that the extra assurance offered by these drivers will induce sufficiently many new participants that the system becomes self–sustaining, largely removing the need for dedicated drivers. In any event, the presence of these drivers in a ridesharing system leads to a number of interesting questions, such as:

- What fare should riders that are served by a dedicated driver pay?
- How should the service provider pay for the costs incurred by employing dedicated drivers?
- What is the the relationship between the number of riders served and the additional cost?
- What settings are most likely to require dedicated drivers?

As we expect the environment under consideration to have a significant effect on the use of dedicated drivers, we identify three key network properties:

- **Number of participants:** A greater number of participants increases both the likelihood of finding a match, and the quality of the best match found. Of course, this only applies if the region is held static, so we are interested in the number of participants per square mile.
- **Travel patterns:** With modern city centers often being associated with high land prices and traffic congestion, many cities have used zoning laws to encourage a more dispersed layout with multiple main hubs. Additionally, even mono–centric cities often exhibit more dispersed travel patterns outside of rush hour. As participants obviously need compatible itineraries, the presence of multiple hubs effectively pro-

duces a series of disconnected subproblems with lower participant densities (and hence lower reliability).

- **Time flexibility:** The willingness to deviate from desired travel times increases the number of feasible matches in two ways. First, it allows greater temporal difference between customers traveling the same route, and it also allows participants to perform greater detours in order to get a match. However, the additional matches generated are likely to be of a lower quality, so will only be included if they allow an increase in system–wide matches.

4.3 Reformulation as a Transshipment Problem

Before turning our attention to dedicated drivers, we will quickly detail some advancements for the more traditional setting made during this thesis. Consider the setting in Section 2.4.2.4, where riders require a guarantee of service on both legs of a round trip. For notation purposes, we say every participant $i \in D \cup R$ has an *out–going* and *in–coming* trip, corresponding to the first and second legs respectively (for now, assume drivers have also announced two trips, although we relax this below). In the absence of further restrictions, this setting requires a general integer program (Agatz et al., 2011). However, we will show that under minimal additional assumptions, it can be expressed as a transshipment problem (which is well known to be polynomially solvable). Specifically, we must assume there is no driver $d \in D$ who can serve the out–going trip of rider $r_1 \in R$ and the in–coming trip of another rider $r_2 \in R$ (although even this is partially relaxed below). Equivalently, this can be viewed as two temporally distinct ride sharing problems that share a common set of riders. We maintain the restriction that each driver can only serve one rider per trip, i.e., a driver can't serve the out–going trip of two different riders (they can of course serve one out–going and one in–coming rider). This temporal separation between trips is similar to a work day commute, where riders request trips going to and from work, and it is impossible for a rider to request a trip that spans both time periods.

For notation purposes, let the set of out–going and in–coming driver trips be denoted by sets D_O and D_I respectively. We can form a transshipment problem with origins D_O, destinations D_I, and transshipment nodes R, where an arc exists between a driver, $d \in D_O \cup D_I$ and a rider $r \in R$ if a ride share between the two parties is feasible, with weight corresponding to the chosen objective. We complete this formulation by adding two sink nodes, which allows us to ensure a minimum number of customers are served. Naturally, all arcs except the one between the two sink nodes have a capacity of one. An optimal solution to this transshipment problem will give an optimal set of ride share pairings, where all matched riders have a driver for both of their trips. An example of this is given in Figure 4.1.

Unfortunately, this formulation will not work as intended, because riders could legally be assigned multiple out and in trips, as long as they receive the same number of each. This is best handled by replacing each rider node by two dummy nodes, with the single connecting arc having a capacity of one.

This setting can actually be further generalized. We do not require that a participant's

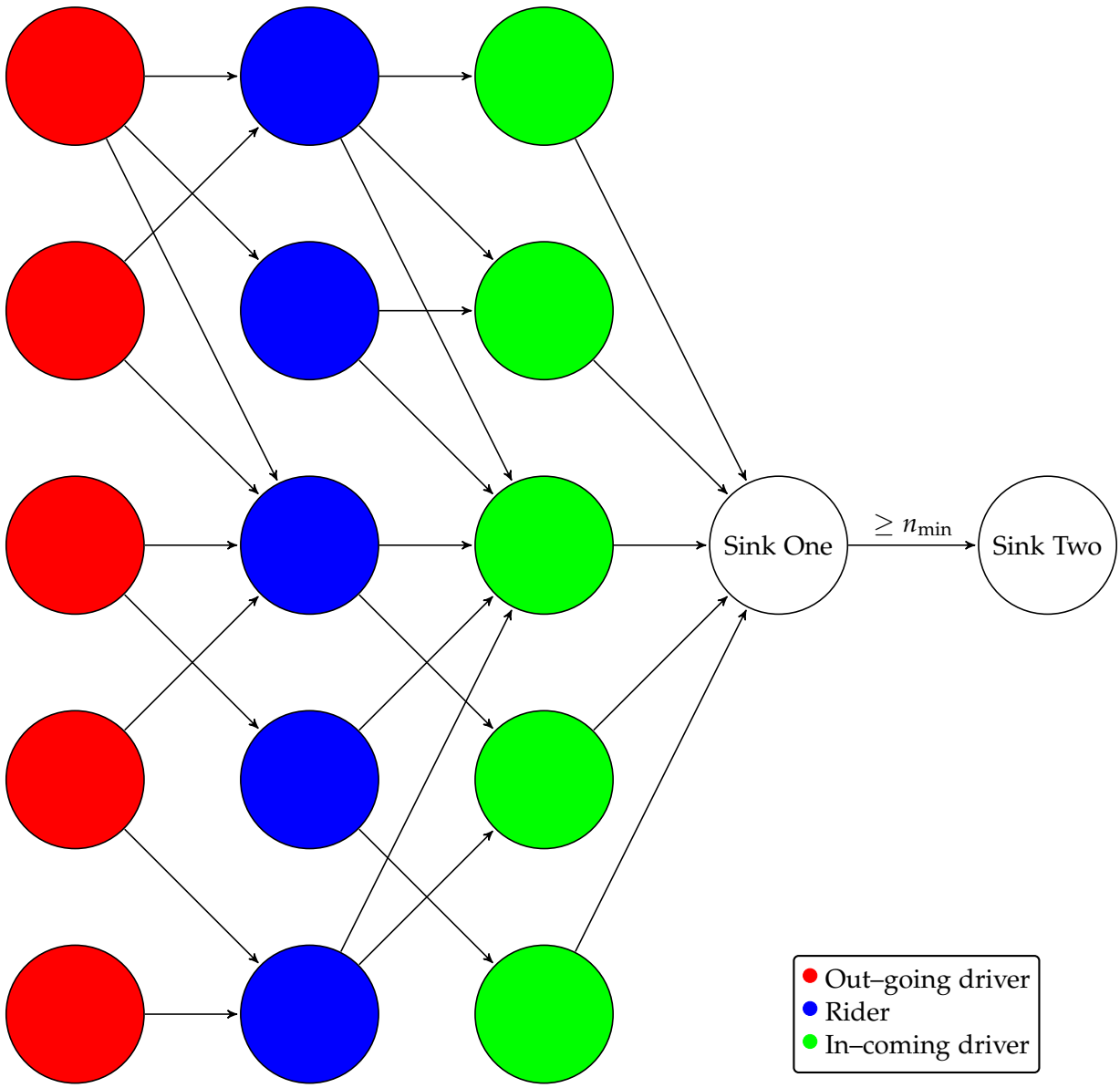


Figure 4.1: Network that enforces return trip constraints

out–going and in–coming trips share geographical locations (although they typically will in most real world settings). Additionally, a driver may only be available for one trip (we simply remove the node corresponding to trip which is not offered). Similarly, if a rider only wants a trip in one direction, we can match them with a dummy driver (at zero cost) for the missing leg.

We can also include riders who violate the temporal separation requirement given above (i.e., there is a driver who can serve their outgoing trip and another rider's incoming trip, or vice versa), by treating each of the former rider's trips as a different, dummy rider traveling a single leg. However, it is possible that only one of these trips will be serviced, e.g., we cannot guarantee a round trip.

Taken together, these generalizations allow common ride sharing variants seen in multiple settings world–wide to be solved optimally in polynomial time, producing better results and offering meaningful impact.

4.4 DRS with Dedicated Drivers (DRSDD)

We now consider the case where the service provider has a fleet of dedicated drivers, who can serve unmatched riders in order to improve the reliability of the system. The provider may choose all aspects surrounding the operation of these drivers, subject to typical routing constraints. Drivers must start and finish from (potentially unique) morning and afternoon depots and may have maximum shift durations. While operating, drivers must respect the time windows of riders they serve and can only serve one rider at a time. If all requests are known prior to drivers starting their shift, then routes can be made in advance and participants notified immediately after. Otherwise, a rolling horizon approach (as given in (Agatz et al., 2011)) is used, where routes are re–optimized at regular intervals (and participants are informed of matches as late as possible). The provider must cover the operational costs of these drivers; possible revenue models and fare levels are discussed in Sections 4.4.1 and 4.4.2. Service providers likely have two goals: to maximize their overall profit (for short term financial benefit) and to serve as many customers as possible (to grow their customer base through better system reliability). Conflict between these objectives can arise if serving customers represents a net cost, as discussed in Section 4.4.3.

4.4.1 Revenue model for the service provider

Before discussing the revenue model adopted in the presence of dedicated drivers, we review the provider's finances, in terms of revenue generated and costs incurred. Every time a rider is matched with an ad hoc driver, they pay the provider some amount f_r, likely calculated using the proportional method discussed in Section 2.4.2.1 and first given in Agatz et al. (2010). From this, the provider takes two small fees ϵ_r and ϵ_d, representing their commission for matching the rider and driver respectively, where $\epsilon_r, \epsilon_d \ll f_r$. The commission may be a percentage of the savings resulting from the matching or a flat fee.

For this study, we we assume the latter, and that $\epsilon_r = \epsilon_d = \epsilon$. The balance, denoted by f_d, is paid to the ad hoc driver as compensation, i.e., $f_d = f_r - 2\epsilon$.

In our extended setting, the provider must cover the operational costs associated with employing dedicated drivers, which can be broken down into *mileage costs* and *wage costs*. That is, the service provider incurs a cost for every mile a dedicated driver travels (whether he is serving a rider or not) and a cost for the total time the driver is away from his home base.

Next, we consider where the required revenue may come from. Every time a rider is served by a dedicated driver, they still pay some fare f_r; different ways to calculate this are discussed in Section 4.4.2. The service provider again takes a commission ϵ_r from the fare and *must* use the remainder, given by $f_{sp} = f_r - \epsilon_r$, solely to fund the operation of dedicated drivers. As a result, the budget required to pay for the dedicated drivers is the sum of mileage and wage costs incurred across all dedicated drivers minus the adjusted fares collected from the riders served by dedicated drivers i.e., the collected fare minus the commission. The greater commission earned from using ad hoc drivers (e.g., 2ϵ vs ϵ) incentivizes their use, which is in line with the traditional ridesharing philosophy and mitigates allegations that the provider is running a disguised taxi service.

It is possible (or for reasons discussed in Section 4.4.2, even desirable) that the fares paid by riders do not fully cover the cost associated with dedicated drivers, so we will consider other ways of raising the required funding. One possibility is an initial investment from a public institution or transport authority to facilitate the uptake of ridesharing, to allow the system to become self-sufficient. This decision could be motivated by the positive externalities associated with a successful and sustainable ride sharing system, like reduced congestion, lower vehicle emissions and increased mobility. We note that although the use of dedicated drivers may result in additional miles driven for the specific riders served (compared to the riders driving themselves), we repeat that the presence of these drivers can allow the system to become larger and self–sustaining, which means more matches with ad hoc drivers. The costs involved will be small compared to the costs of capital works and the support could be for a limited time only.

If the service provider must produce the required budget themselves, and we assume that riders served by dedicated drivers do not cover the full cost, then we must consider alternative funding mechanisms to cover any shortfall. This could include paid advertising on the smartphone application used to book the service, or having a premium subscription where paying participants get special treatment (greater certainty of a match, tighter time windows, smaller lead time, etc). It is even conceivable that ad hoc drivers should contribute to the cost as well, because the presence of more riders increases both the quality and frequency of their own matches.

4.4.2 Fare Levels Charged

When determining fares paid by riders served by dedicated drivers, we must first discuss if we aim to fully cover the associated operating costs. On one hand, the riders are the most obvious beneficiaries of dedicated drivers (receiving something similar to a taxi

service), so should be charged accordingly. However, the scheme is intended as a back up option to reduce uncertainty, and taxi services are generally widely available anyway. Furthermore, if fares are substantially different (compared to riders matched with ad hoc drivers), there are significant equity issues surrounding the assignation of drivers to riders. For this reason, we assume that fares charged to riders are broadly similar regardless of the driver they travel with. With this in mind, a few natural pricing schemes present themselves.

1. **Fraction of direct drive cost:** We could charge a rider a fraction of the cost they would incur by driving themselves, α, where $0 < \alpha < 1$. A possible starting point is to set $\alpha = 0.5$, as riders are essentially ride sharing with a driver between their own origin and destination, however many riders with ad hoc drivers would not get such an advantageous rate. The cost calculations would probably assume a per mile fee for a "standard" vehicle, so may be inaccurate for riders who would otherwise take a highly efficient (or inefficient) car. On balance, this scheme is easy to understand and riders won't feel that they're being "ripped off". However, it ignores the fact that some riders (likely those who will be served by dedicated drivers) may be somehow difficult to serve, and it may be appropriate to add a surcharge on to their trips.
2. **An historical per mile fee:** Instead of assuming a per mile fee based on vehicle efficiency, we can use historical data to determine the average per mile fee paid by a rider matched with an ad hoc driver, as given in Equation (2.44). The scheme can be refined to take distance into account, e.g., by restricting the analysis to historical matches involving riders with a similar length trip. To be even more fine-grained, we could restrict the analysis to matches involving riders with nearby origins and destinations, or at approximately the same time of day. Additionally this scheme is, in a way, self–correcting. If the characteristics of the participating drivers change over time, resulting in higher or lower charges for the riders matched with ad hoc drivers, then these changes are automatically reflected in the per mile fee charged to riders matched with dedicated drivers. This scheme is slightly more complex and less transparent than the first, but should still be viable.
3. **Average of current ad hoc fares** Instead of determining a fare based on historical data, we could look at *currently active* matches the rider could participate in, and take the average of these prices. If there are no such ad hoc drivers, we can revert to the second pricing scheme. Both schemes should give similar prices, except during temporary distortions like special events or holiday periods (where different fares may be reasonable).

4.4.3 Choice of Objectives

We suggest three possible goals the service provider might adopt: to minimize their net costs, to minimize the total system wide miles driven, or to maximize the number of riders serviced. We will discuss each objective in turn, then provide an illustrative example to highlight key points.

4.4.3.1 Maximize Net Profit/Minimize Net Costs

If the service provider is a private firm, they have an obligation to their stakeholders to maximize profits/minimize net costs. As dedicated drivers incur a cost for every trip they perform, we need a policy regarding their use (otherwise, costs are minimized by never using them). We define the *service level*, denoted by $\hat{s}$, which is the minimum percentage of riders that must be served (by a driver of either type). Consequently, the minimum number is given by $n^{\min}$, e.g., $n^{\min} = \lceil \hat{s} * |R| \rceil$. We feel this objective will strengthen claims the provider is really a taxi service and weaken the case for any public subsidies. Finally, the service level requirement gives strong links to the Prize Collecting Vehicle Routing Problem, although the use of net costs on the arcs is reminiscent of the Vehicle Routing Problem with Profits (both problems were introduced in Section 2.1.6).

4.4.3.2 Minimize Total System Wide Miles Driven

Another natural objective is to instead minimize the number of system wide miles driven (by both ad-hoc drivers and dedicated drivers), likely reducing vehicular congestion. Additionally, we note that vehicle emissions are closely related to distance traveled, so this objective can be dual–purposed. Again, we assume that riders not matched will drive themselves, otherwise the solution which minimizes system wide miles involves no matching at all. However, the use of dedicated drivers will still increase total miles driven (again, we argue that if dedicated drivers improve rider retention, then those riders can be matched with ad hoc drivers on other days, making up for the additional miles). As these objectives have mainly societal benefits, they are more likely in settings where public funds are involved. Again, we note the strong links to the Prize Collecting Vehicle Routing Problem and the Vehicle Routing Problem with Profits.

4.4.3.3 Maximizing Service Level

Another natural objective is simply to maximize the number of requests met, with some constraints regarding the use of dedicated drivers. This could involve a maximum number of drivers and/or budgetary constraints on their operational cost. This objective has both private and public benefits; the operator gets the maximum number of participants (showing system reliability and growing towards critical mass), and can't be accused of profiteering. Additionally, society benefits from a reliable mobility system that serves as many people as possible. This setting is similar to the Orienteering Problem (where all nodes have an equal reward).

There are two subtly different ways the budget constraint could be implemented; we could limit the *operational* cost or the *net* cost, with the difference being the former excludes fares paid by riders served by dedicated drivers. As an example, in small network given in Figure 4.2, (where arc labels indicate the sum of the mileage and wage costs), the operational cost of the route is given by $c_1 + c_2 + c_3$, but the net cost is $c_1 + c_2 + c_3 - f_{sp}$. Both settings still allow collected fares to fund the budget, the distinction applies only when determining if the budget has been exceeded. Although these settings may similar, using the net

cost significantly complicates the problem. If given a partial route for a dedicated driver, there may exist a rider that requires such a small detour that serving them incurs a negative net cost, e.g., a net profit, (such an example is discussed in Section 4.4.3.5). If our budget applies to the net cost, this effectively increases the budget and could allow the operator to serve a customer they would otherwise have insufficient budget for. This means serving a rider with an ad hoc driver isn't always best; an unintuitive situation which runs counter to the spirit of ride sharing. For this reason, we will assume the budget applies to the operational cost.

Finally we note that we could apply the concept of operational cost to the first objective, by excluding the fares paid by riders from the objective function. However, we feel this does not sufficiently reward riders the require a small detour and just generally acts as a distortion.

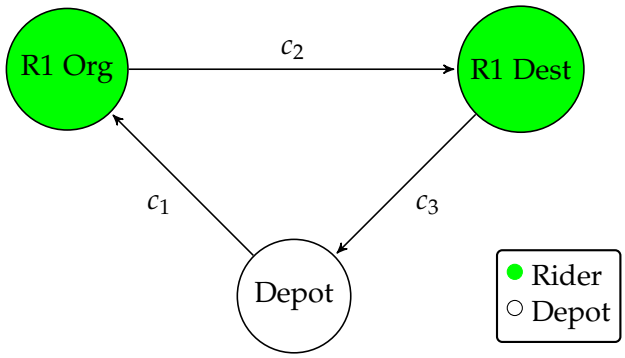


Figure 4.2: Small network to illustrate difference between operational and net cost

4.4.3.4 "Cherry picking" riders

Under all objectives, the service provider has a strong incentive to select the "cheap" riders (the exact meaning of "cheap" is context specific). This is not in itself inherently bad and represents the main point of optimization studies. There may be unintended consequences, e.g., to save costs, dedicated drivers might only serve riders between their morning and afternoon depots, but it is not clear this is undesirable, as it does allow the maximum number of people to be served. These issues are part of real world optimization problems and can be mitigated with careful consideration and additional constraints as required.

4.4.3.5 Objective Comparison

We now present an illustrative examples to demonstrate the relative advantages and disadvantages of each objective. If we consider the network in Figure 4.3, it is not obvious if rider R2 should be serviced by a dedicated driver or by the ad hoc driver, D1. So we can make progress, assume the following:

- **No overlap of time windows** Assume the time windows of the riders have no over-

lap.

- **Order of time windows** Assume that time windows occur in the same order that riders are numbered i.e. the window for rider R1 occurs before that for rider R2, which occurs before that for rider R3 etc.
- **Gap between time windows** Assume there is sufficient gap between successive windows for a dedicated driver to finish servicing one customer and travel to the origin on the next i.e. the dedicated driver can feasibly service all riders present.
- **Service level:** Assume we must serve all riders if possible (note that we can substitute this for a looser requirement by adding in customers who are more difficult to serve).

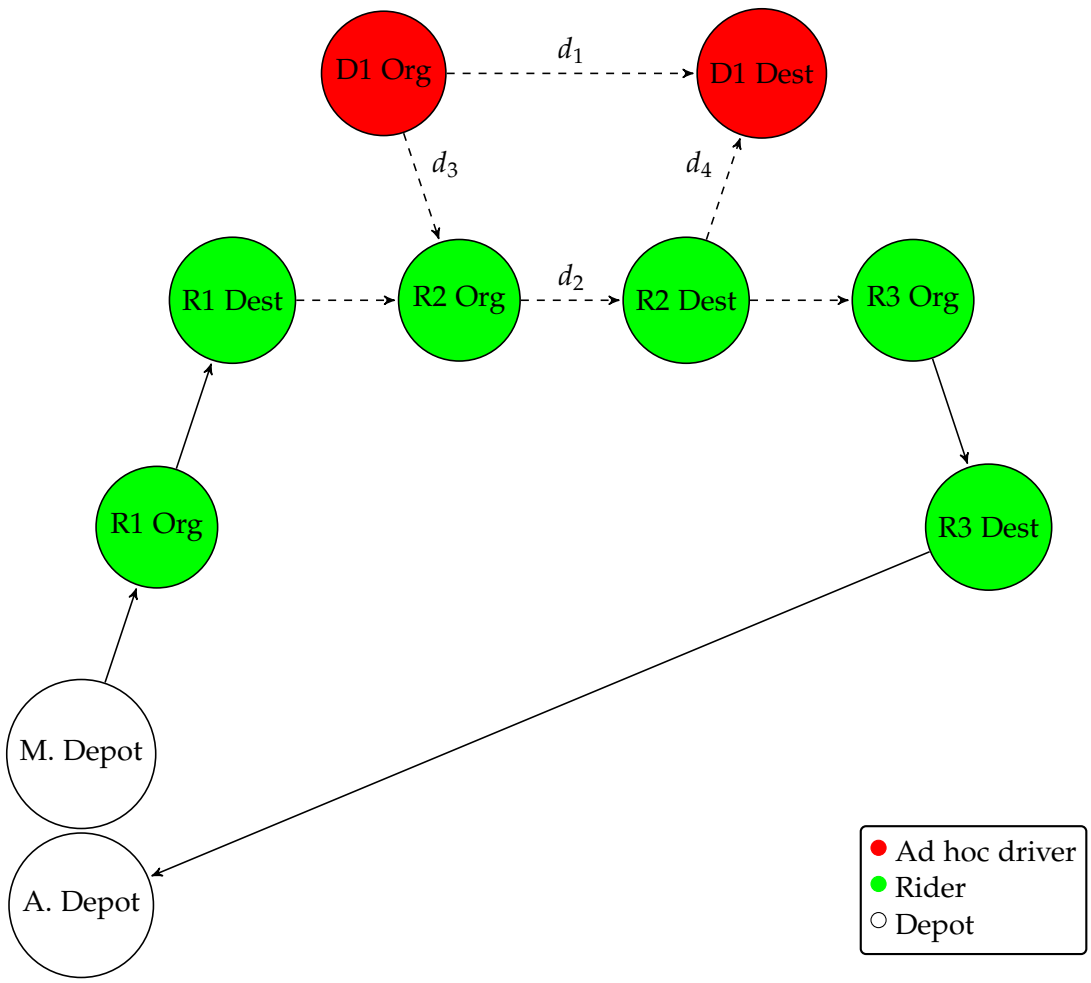


Figure 4.3: Example network to compare objectives

If the objective is to minimize the cost incurred by the provider, then the dedicated driver should serve rider R2. The provider incurs no extra cost for doing so (as the dedicated driver will travel the route anyway), and the fare received by the operator is greater than the commission from matching ad hoc participants. However, we again stress that riders pay less than the true trip cost, so it is impossible for collected fares to exceed the *total* operational cost of dedicated drivers. Additionally, using the dedicated driver goes against the ride sharing philosophy and the absence of a match may discourage the ad hoc driver from announcing future trips. Similarly, if the objective is to minimize system wide miles (and we assume unmatched riders will drive themselves), then we should use the dedi-

cated driver to serve rider R2. However, the use of dedicated drivers must still increase miles driven across the *whole* system. Finally, if the objective is just to maximize matches, then it doesn't matter (with respect to the objective value) who serves rider R2.

This network in Figure 4.3 is obviously somewhat special, as serving the second rider requires no detour. We could shift this rider upwards, until the net revenue earned equals the commission obtained from the matching. Beyond this critical point, there is little argument for using the dedicated driver. Finally, we later present computational experiments showing situations like this are very rare.

4.4.4 Static vs Dynamic Systems

Next we should discuss the differences between *static* and *dynamic* systems and the resulting impact on operational policy and efficiency. In static systems, the entire set of riders and drivers are known when the service provider designs their operating schedule; conversely, dynamic systems involves a set of participants which only become known as time progresses, i.e., when they announce their trip through the smartphone application. There is a natural trade–off between the systems – the additional information in static systems allow for better matches to be found, but the requirement to pre–book will lower participant rates (and the quality of the resulting matches). A common way to estimate the degree of dynamism is the lead time required by the operator (Agatz et al., 2010).

There are multiple points of difference regarding how dynamic systems specifically affect dedicated drivers (as opposed to ad hoc ones). First, we note that in systems with only ad hoc drivers, once lead time reaches a certain level, all feasible matches are possible (and further increases will have no effect). This is because ad hoc drivers only serve one rider, so there are no propagating effects from their use. However, when a rider is assigned to a dedicated driver, it affects that driver's ability to serve subsequent riders, giving a greater penalty for dynamism.

Another complication caused by dynamism on the use of dedicated drivers is periodic re-optimization. With a dynamic customer set, most approaches regularly generate updated solutions as more information becomes available, e.g., the rolling horizon method given in (Agatz et al., 2011). From a modeling perspective, this means each dedicated driver must be considered individually, with a unique depot and start time, representing their current location (or the destination of their current passenger), at the current time (or their expected arrival time). This is well known to complicate solution algorithms by requiring the definition of new variables and expanding the formulation. As an example, in the Integer Program presented in Section 4.5.1, we would need to reference the arc decision variables over each vehicle, greatly increasing the solution space. Similar statements can be made regarding the duplication of sub–problems in Branch and Price algorithms and about various construction or improvement routines inside meta–heuristics.

Periodic re–optimization also raises questions regarding the provider's operating policy. First, it is easy to see the greatest efficiency is obtained when participants are informed about a match as late as possible, in case new customers allow a better solution. However, this uncertainty may annoy participants, especially since last minute departures car-

ries obvious risks with uncertain travel times. Secondly, the operator may wish for vehicles to hold their position after completing a job (if further jobs are expected in that area). It is not clear where exactly the vehicle should wait, especially in dense urban areas where parking is limited. Of course, such issues can arrive in the static case, but it is known for how long vehicles will be stationary, making planning easier (perhaps by scheduling driver's breaks for this time).

4.4.5 Related Work

We will now detail the related work in this area. From the literature relating to Dynamic Ride Sharing, Agatz et al. (2011) contains a discussion concerning the long term sustainability of a DRS scheme. They find that such systems do tend to stabilize around a certain participation rate, which is determined by two important, albeit intuitive factors: the rate at which participants join the system, and the rate at which participants leave the system after repeatedly failing to find a match.

As discussed in Section 4.4.3, depending on the objective and constraints chosen by the operator, our problem of efficiently routing dedicated drivers has strong links to Selective VRPs (introduced in Section 2.1.6). Broadly speaking, these problems involve a team of entities traversing a network of nodes and arcs, where each node has a reward and each arc has a cost (Jozefowiez et al., 2008). There are generally two goals: to maximize the rewards collected from visiting (a subset of the) nodes, but to also minimize the cost incurred while doing so. There are however some key differences. First, in our problem the resource constraint (representing the operational budget) restricts the *total* expense (aggregated over all drivers). This expands the solution space and would require the extension of Selective VRP methods; Integer Programming approaches need new valid inequalities, Branch and Price methods require new branching rules and meta–heuristics need their improvement routines extended to explore the larger search area. Additionally, each dedicated driver can have an unique morning and afternoon depot with different start and end times (only a maximum time away from the home base has to be respected). However, the most important difference is that schedules for dedicated drivers have to be created simultaneously and in conjunction with the matching of ad hoc drivers and riders.

Additionally, there are close links to the "crowdshipping" problem mentioned in Section 2.4.2.2. Both settings investigate moving items through a network, either with cheap vehicles that operate along heavily constrained routes, or with expensive vehicles that can be sent anywhere. However, there are some important differences. First, the time windows that restrict parcel pickups are much looser than those on riders, making it easier to find solutions. Additionally, all shipping requests originate from a set of predefined depots, while riders have their own unique (though potentially clustered) origins. Finally, real world crowdshipping settings require that almost all parcels (higher than 99%) be delivered (by any driver); this is much higher than the expected success rate for ride sharing systems.

4.5 Exact Methods for the DRSDD

4.5.1 An Integer Programming Formulation

We now present our extended variant as a formal optimization problem. To avoid duplicate notation, we will reuse notation from Section 2.4.2.1. Consequently, the sets of drivers and riders are denoted by $d \in D$ and $r \in R$ respectively. Each participant $i \in D \cup R$ has a departure window of $[e_i^{Dept}, l_i^{Arr}]$, and an arrival window of $[e_i^{Arr}, l_i^{Arr}]$. Travel time between a participant's origin and destination is given by $\hat{t}_i$, and the time flexibility, t^{Flex}, indicates how much earlier a participant is willing to leave in exchange for being paired with another participant, i.e., $t^{Flex} = l_i^{Arr} - e_i^{Dept} - \hat{t}_i$. We assume that the value given by t^{Flex} is a maximum, and participants prefer to depart closer to l_i^{Dept} (and arrive closer to l_i^{Arr}). Participants are associated with a lead time indicating how early they provided their itinerary, given by H_i. Riders are served by a single driver, through drivers (with sufficient capacity) may serve a group of riders with the same travel plans who make a single booking. Matches are only permitted between drivers and riders if they reduce total cost (assuming unmatched participants would drive themselves), as described in Figure 2.10.

With the preliminaries dispensed with, we now turn our attention to the issues surrounding the use of dedicated drivers. In our computational study, we focus on understanding the connection between budgetary limits and service levels achieved. Hence (unless otherwise noted), the remainder of the chapter will assume the scenario outlined in Section 4.4.3.1, where the operator aims to meet a certain *service level*, $\hat{s}$ at minimum cost (again, the service level is the minimum percentage of riders that must receive a match). We assume the revenue model from Section 4.4.1, where the operator receives a small constant commission ϵ from all participants in a ride sharing match (the remaining savings would be distributed in some manner, although the details of this does not affect our analysis). Furthermore, for riders served by dedicated drivers, the operator again takes a small commission from the fare, ϵ, and puts the rest towards the cost of dedicated drivers. If we assume the rider pays a fee proportional to the length of their trip, there are two relevant costs – one for every mile the vehicle travels empty, and a reduced rate for every mile the vehicle has a customer.

The problem involving the use of dedicated drivers to serve riders can be represented on a graph $G(\mathbf{N}, \mathbf{A})$. We could include the ad hoc drivers, but that would mask the nice bipartite structure present in the matching problem (of course, both driver types must be considered simultaneously to solve the problem to optimality). The graph has a node for each rider $r \in R$ and an arc (i, j) between the nodes for riders i and j represents a dedicated driver leaving the destination of rider i, visiting the origin of rider j, and transporting rider j to his destination (this is referred to as *serving rider j*). Therefore, the travel time associated with arc (i, j), denoted by t_{ij}, is defined to be the time needed to drive from the destination of rider i to the origin of rider j, and then to the destination of rider j. A time T_j is associated with the node for rider j representing the time they will arrive at their destination if served by a dedicated driver; this must fall within the window $[e_j^{\text{Arr}}, l_j^{\text{Arr}}]$. An explanatory diagram is given in Figure 4.4, where we have shown the destination of rider i, and the origin and destination of rider j. Nodes and arcs in G are drawn with solid lines, while others are dashed. Effectively, we have replaced the two links connecting the

destination of rider i with the destination of rider j with a single link, where the travel time is the sum of that on the replaced links. Similarly, let c_{ij} represent the *net* cost associated with a dedicated driver traversing arc (i, j), i.e., it includes the fare paid by the rider. We needn't be so explicit about the arrival time of riders served by ad hoc drivers; as these drivers serve only one rider, any time feasible journey plan will suffice.

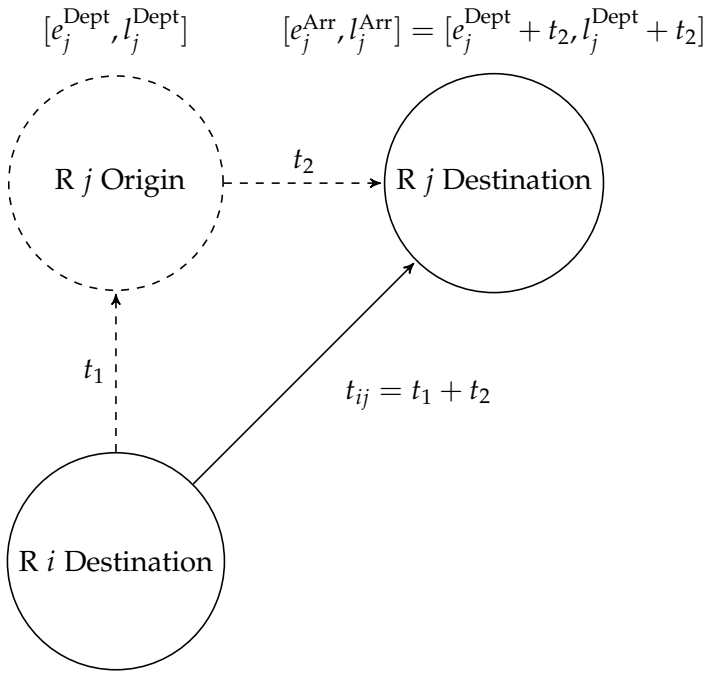


Figure 4.4: Network reduction techniques used

For every dedicated driver $k \in K$, we add two nodes, m_k and a_k to the set $\mathbf{N}$, representing the driver's morning and afternoon depot (they may be unique and distinct, as under a rolling horizon approach). For convenience, we define the sets $M = \bigcup_{k \in K} m_k$ and $A = \bigcup_{k \in K} a_k$. Each dedicated driver has an earliest start and latest finish time, given by e_k and l_k, respectively. The time a dedicated driver actually leaves his starting location is given by T_k^e and the time at which he returns is given by T_k^l. The difference between these two times is controlled by a drive time limit L i.e., $T_k^l - T_k^e \leq L$, $k \in K$ where $T_k^e \geq e_k$ and $T_k^l \leq l_k$. Finally, we assume drivers operate homogeneous vehicles, i.e., there is no difference in travel times or operating cost.

We introduce a binary variable y_{ij}^k, $i, j \in R$ to indicate if dedicated driver k uses arc (i, j), and extend this with binary variables $y_{m_k j}^k$ and $y_{j a_k}^k$ to capture the start and end of their route. Next, we introduce binary variables $y_{m_k a_k}^k$, $k \in K$ to represent that a driver is not used. Finally, we reuse the binary variable x_{ij} to represent the matching between ad hoc driver $i \in D$ and rider $j \in R$.

The following integer program ensures that a minimum number of riders is served, either by an ad hoc driver or by a dedicated driver, at minimum cost:

Integer Program 8:

$$\min \sum_{k\in K}\left(\sum_{j\in R} c_{m_k j} y^k_{m_k j} + \sum_{i\in R}\sum_{j\in R} c_{ij} y^k_{ij} + \sum_{j\in R} c_{ja_k} y^k_{ja_k}\right) - \sum_{i\in R}\sum_{j\in D} \epsilon x_{ij}$$

$$\sum_{j\in R} x_{ij} \leq 1, i \in D \tag{4.1}$$

$$\sum_{i\in D} x_{ij} + \sum_{k\in K}\left(y^k_{m_k j} + \sum_{i\in R} y^k_{ij}\right) \leq 1, j \in R \tag{4.2}$$

$$\sum_{j\in R}\left(\sum_{i\in D} x_{ij} + \sum_{k\in K}\left(y^k_{m_k j} + \sum_{i\in R} y^k_{ij}\right)\right) \geq n^{\min} \tag{4.3}$$

$$\sum_{i\in R\cup M} y^k_{ij} - \sum_{i\in R\cup A} y^k_{ji} = 0, j \in R, k \in K \tag{4.4}$$

$$\sum_{j\in R} y^k_{m_k j} + y^k_{m_k a_k} = 1, k \in K \tag{4.5}$$

$$T_j \geq e^{\text{Arr}}_j,\ j \in R \tag{4.6}$$

$$T_j \geq \sum_{k\in K}(T^e_k + t^k_{m_k j}) y^k_{m_k j},\ j \in R \tag{4.7}$$

$$T_j \geq (T_i + t_{ij}) \sum_{k\in K} y^k_{ij},\ j \in R,\ i \in R \tag{4.8}$$

$$T_j \leq l^{\text{Arr}}_j, j \in R \tag{4.9}$$

$$T^e_k \geq e_k,\ k \in K \tag{4.10}$$

$$T^l_k \geq \sum_{j\in R}(T_j + t^k_{ja_k}) y^k_{ja_k},\ k \in K \tag{4.11}$$

$$T^l_k \leq l_k,\ k \in K \tag{4.12}$$

$$T^l_k - T^e_k \leq L,\ k \in K \tag{4.13}$$

$$x_{ij} \in \{0,1\}, i \in D, j \in R$$

$$y^k_{ij} \in \{0,1\}, i \in R \cup m_k, j \in R \cup a_k$$

The objective attempts to minimizes the provider's net cost, i.e., operational expenses less revenue raised (recall that the $c_{ij},\ i,j \in R \cup M \cup A$, parameters include the rider's fare). Constraints (4.1) ensure that an ad hoc driver does not serve more than one rider. Similarly, constraints (4.2) ensure that a rider is not served by more than one driver, including dedicated drivers. Constraint (4.3) ensures that the required minimum number of riders is served. Constraints (4.4) ensure that if a dedicated driver enters a node, he also leaves it. Constraints (4.5) ensure that a dedicated driver leaves his home base. Constraints (4.6) – (4.9) ensure consistency of the arrival times of riders (served by dedicated drivers) at their destinations. Constraints (4.10) – (4.12) enforce the relevant time windows for dedicated drivers and constraints (4.13) ensure that a dedicated driver does not exceed their shift time limit.

Although constraints (4.7), (4.8), and (4.11) are nonlinear, it is straightforward to linearize them. This is necessary to obtain an integer program that can be solved by any of the available commercial integer programming solvers.

$$T_j \geq T_k^e + t_{m_k j}^k - M(1 - y_{m_k j}^k), \quad j \in R, /k \in K \tag{4.7'}$$

$$T_j \geq T_i + t_{ij} - M(1 - \sum_{k \in K} y_{ij}^k), \quad i, j \in R \tag{4.8'}$$

$$T_k^l \geq T_j + t_{ja_k}^k - M(1 - y_{ja_k}^k), \quad j \in R, /k \in K \tag{4.11'}$$

Although our objective is to minimize the net cost faced, it is not difficult to modify the above integer program to serve the maximum number of riders (subject to a budget constraint on the cost incurred by dedicated drivers).

4.5.2 Problem Structure

Broadly speaking, this problem contains a special structure which allows it to be split into two parts. The first involves finding a bipartite *matching* between ad hoc drivers and riders and the second creates *routes* for dedicated drivers to undertake (in both cases, all relevant constraints must be respected). The first problem is both algorithmically and computationally simple, whether done for all riders or only those not served by a given set of routes. However, optimally scheduling dedicated drivers is a challenging task, even if the problem size is reduced by matching some riders with ad hoc drivers. Of course, care must be taken when splitting the problem to ensure the optimal solution can still be obtained. We will exploit this structure below, both in our Branch and Price algorithm and our Variable Neighborhood Search meta–heuristic.

4.5.3 A Branch and Price Algorithm for the DRSDD

As discussed in Section 2.3.1.3, Branch and Price approaches are able to (optimally) solve larger routing problems than any other method, so we present such an algorithm for our problem. As alluded to above, this actually involves two sub–problems; a bipartite matching problem involving ad hoc drivers, denoted by MSP and an elementary shortest path problem with time windows involving dedicated drivers, denoted by ESSPTW . We will make the same assumptions as Section 4.5.1 around our choice of revenue model, objective and operational constraints. We will briefly introduce some notation, then detail the Master Problem (MP) and the two sub–problems.

First, consider the MSP, where the set of known feasible matchings will be denoted by $p \in P^m$ and where a^p is a binary vector containing values representing the pth (feasible) matching. Specifically, the jth element of a^p, given by a_j^p, is one if rider j is matched with an ad hoc driver in the pth matching and zero otherwise. λ^p is a binary variable indicating if matching $a^p, p \in P^m$ is selected in the MP.

Next, we consider the ESSPTW, which is duplicated for each dedicated driver. Let the set of known feasible routes for driver $k \in K$ be denoted by V^k and let $V^A = \bigcup_{k \in K} V^k$. Again,

let $a^{v,k}, v \in V^k$ be a binary vector containing values which represent the vth (feasible) route for driver k, where the jth element of $a^{v,k}$, given by $a_j^{v,k}$, is one if rider j is served by the dedicated driver and zero otherwise. Again, $\lambda^{v,k}$ is a binary variable controlling if route $a^{v,k}, v \in V^k$ is selected by the MP.

A solution to the MSP or ESSPTW (i.e., a matching $p \in P^m$ or route $v \in V^k, k \in K$) has an associated column in the MP, with a cost given by, c^p or $c^{v,k}$ respectively. As the objective concerns the minimization of the net cost to the provider, and the commission from matches represents a positive revenue, $c^p < 0$ while $c^{v,k} > 0$ (unless the driver isn't used, in which case $c^{v,k} = 0$). Similarly, each solution has an associated number of riders served, given by b^p and $b^{v,k}$. Specifically, if e_1 is the unit vector of ones with cardinality $|R|$, then $b^p = e_1^{\text{T}}.a^p$ and $b^{v,k} = e_1^{\text{T}}.a^{v,k}$. We also include a virtual variable, z, representing the difference between the required minimum number and actual number of riders served. This variable is added to ensure the minimum service constraint can always be satisfied (avoiding infeasibility), but is heavily penalized in the objective function.

Of course, generating the sets $P^m$and $V^A$is computationally intractable, so we work with reduced sets $\hat{P}^m \subset P^m$, $\hat{V}^k \subset V^k$ and $\hat{V}^A \subset V^A$ representing the sets of currently known solutions. Consequently, we also use the term *Restricted* Master Problem (RMP).

4.5.3.1 Restricted Master Problem

Before presenting the RMP, we will give a brief overview. It can be viewed a set covering problem, where we need to select one column from P^m (i.e., one matching with ad hoc drivers) and one column from each of the sets $V^k, k \in K$ (i.e., one route for each dedicated driver). Naturally, each rider can be served at most once. Additionally, there is also a generalized covering constraint enforcing the minimum service level. Once optimality is achieved (or almost achieved) for the current set of columns, we use our sub–problems to generate new matchings and routes.

Restricted Master Problem:

$$\min \sum_{p \in \hat{P}} c^p \lambda^p + \sum_{k \in K} \sum_{v \in \hat{V}^k} c^{v,k} \lambda^{v,k} + Mz$$

$$\sum_{p \in \hat{P}} a_j^p \lambda^p + \sum_{k \in K} \sum_{v \in \hat{V}^k} a_j^{v,k} \lambda^{v,k} \leq 1, j \in R \tag{4.14}$$

$$\text{s.t.} \sum_{p \in \hat{P}} b^p \lambda^p + \sum_{k \in K} \sum_{v \in \hat{V}^k} b^{v,k} \lambda^{v,k} + z \geq n^{\min} \tag{4.15}$$

$$\sum_{p \in \hat{P}} \lambda^p = 1 \tag{4.16}$$

$$\sum_{v \in \hat{V}^k} \lambda^{v,k} = 1, k \in K \tag{4.17}$$

$$\lambda^p \in \{0, 1\} \tag{4.18}$$

$$\lambda^{v,k} \in \{0, 1\} \tag{4.19}$$

$$z \in \mathbb{Z}^+ \tag{4.20}$$

The objective is designed to match the reduced cost calculation in the RMP. Constraints (4.14) ensure no rider is serviced by more than one driver. Constraint (4.15) ensures that (at least) the minimum number of riders are served, (noting previous discussions regarding how serving riders normally represents a net cost, we would expect this constraint to hold at equality in most instances). Constraints (4.16) and (4.17) are the convexity constraints. Finally, constraints (4.18)–(4.20) enforce integrality for our decision and virtual variables.

Before presenting our sub–problems, we must define the relevant dual variables; let constraints (4.14), (4.15), (4.16) and (4.17) have duals γ_i, $i \in R$, γ^0, β^m and $\beta^k, k \in K$ respectively. Broadly speaking, the dual variable γ_i represents how efficiently rider i is served in the current RMP solution and the likelihood that a new matching or route involving this customer will improve the RMP. Similarly, the dual coefficient γ^0 is associated with the minimum service constraint and reduces arc costs in the sub–problems if more riders should be included; alternatively, the magnitude of γ^0 indicates the additional work required to satisfy this constraint. Finally, β^m and $\beta^k, k \in K$ measure how effective and efficient the current matching and vehicle routes are.

4.5.4 Matching sub–problem

Our first sub–problem, associated with matching ad hoc drivers with riders, is largely unchanged from that presented in Section 2.4.2.1. It is still formulated as a weighted bipartite matching problem on the sets D and R with a match between driver i and rider j represented by $x_{ij}, i \in D, j \in R$. However, the associated arc has a weight (or reduced cost) of $\delta^p_{ij} = c_{ij} - \gamma_j - \gamma^0$.

Matching Sub Problem (MSP)

$$\min \sum_{i \in D} \sum_{j \in R} \delta^p_{ij} - \beta^m \tag{4.21}$$

$$\text{s.t.} \sum_{i \in D} x_{ij} \leq 1, j \in R \tag{4.22}$$

$$\sum_{j \in R} x_{ij} \leq 1, i \in D \tag{4.23}$$

$$x_{ij} \in \{0, 1\} \tag{4.24}$$

As this represents a simple bipartite matching problem like that given in Section 2.4.2.1, we will not explain it again.

4.5.4.1 Elementary Shortest Path with Time Windows Sub Problem (ESPTW) for the kth dedicated driver

Our second sub–problem involves creating routes for dedicated drivers and can be seen as an elementary shortest path problem with time windows. We make use of the same network given in Section 4.5.1, with decision variables $y_{ij}, i \in R \cup \{m_k\}, j \in R \cup \{a_k\}$, still indicating if dedicated driver k services rider i immediately before rider j (again, this problem is duplicated for each dedicated driver). However, the associated weights are given by $\delta^k_{ij} = c_{ij} - \gamma_j - \gamma^0, i \in R \cup \{m_k\}, j \in R$ and $\delta^k_{ij} = c_{ij}$ if $j \in \{a_k\}$.

$$\min \sum_{i \in R \cup m_k} \sum_{j \in R \cup \{a_k\}} \delta^k_{ij} y_{ij} - \beta^k$$

$$\text{s.t.} \sum_{j \in R \cup \{a_k\}} y_{ij} - \sum_{j \in R \cup \{m_k\}} y_{ji} = 0, i \in R \tag{4.25}$$

$$\sum_{j \in R \cup \{a_k\}} y_{m_k, j} = 1, \tag{4.26}$$

$$\sum_{j \in R \cup \{m_k\}} y_{j, a_k} = 1, \tag{4.27}$$

$$T_j \geq (T_i + \hat{t_{ij}}) y_{ij}, i \in R \cup m_k, j \in R \cup \{a_k\} \tag{4.28}$$

$$T_j \geq e_j, j \in R \cup m_k \tag{4.29}$$

$$T_j \leq l_j, j \in R \cup a_k \tag{4.30}$$

$$y_{ij} \in \{0, 1\}, i \in R \cup \{m_k\}, j \in R \cup \{a_k\} \tag{4.31}$$

$$T_j \geq 0, j \in R \cup \{a_k\} \tag{4.32}$$

$$T_{m_k} = 0 \tag{4.33}$$

As most of these constraints have a matching counterpart in the formulation from Section 4.5.1, we will focus on the parts specific to Branch and Price. The objective minimizes the path length between the morning and afternoon depots and adds a duality constant, representing how effectively the current vehicle is used. Constraints (4.25)–(4.27) conserve the flow of dedicated drivers. Constraints (4.28)–(4.30) enforce time windows, while constraints (4.31)–(4.33) set the relevant domains for our variables.

Finally, we note that it is possible to consider the sub–problems for all dedicated drivers simultaneously by simply combining the relevant graphs. However, the presence of different morning and afternoon depots requires indexing the arc variables by the driver, meaning little is gained. Of course, the subproblems are so strongly related that information should be shared, i.e., the final solution to one sub–problem can be used as the initial solution for another.

4.5.4.2 Branching strategies

We will now outline a general branching strategy, which is extended from the approach given for the Team Orienteering Problem in Boussier et al. (2007). It comprises of two parts; the first involves branching on the combined flow of vehicles (where λ^p and $\lambda^{v,k}$ variables are considered simultaneously) and the second branches on the individual flows of vehicles (where $\lambda^{v,k}$and λ^pvariables are considered sequentially).

For the first part, if the solution to RMP is fractional, there is (at least) one rider $i \in R$ who is visited a fractional number of times. We derive two branches, either forbidding or enforcing the service of rider i, by adding one of the constraints given in (4.34) to the RMP. These will remain in the problem for the second part of our branching procedure.

$$\sum_{p \in \hat{P}} a_j^p \lambda^p + \sum_{k \in K} \sum_{v \in \hat{V}^k} a_j^{v,k} \lambda^{v,k} = \begin{cases} 1 & \text{in the branch where rider } i \text{ must be served} \\ 0 & \text{in the branch where rider } i \text{ must } \textit{not} \text{ be served} \end{cases} \tag{4.34}$$

If visiting rider i is forbidden, then the corresponding node is also removed from the two subproblems. If several customers with a fractional value exist, we select the one with smallest value $\gamma_j - f_j$. This rule should penalize the branch where the vertex is forbidden, which is potentially more difficult to solve.

Once each rider is served an integer number of times, we turn our attention to arcs with fractional flow (we consider a single dedicated driver and continue branching until intergrality is achieved). For an arc (i, j), $i, j \in R$, there are two possible scenarios:

- **Scenario One:** If either rider i or j must be served, then there are two possible branches: one where arc (i, j) must be used and one where it is forbidden.
- **Scenario Two:** If neither i nor j is constrained to be served, there are three possible branches. The first branch forbids serving rider i. The second and third branches re-

quires us to serve rider i and either enforce or forbid the use of arc (i, j). In all three cases, arc (i, j) will have integer flow.

Once integer flow for a vehicle has been achieved, it may again become fractional following branching for a subsequent vehicle. In this case we just keep on branching until integrality on all vehicles is achieved. Additionally, we may have to branch on the λ^p variables. This follows the same two scenarios as given above, where arcs represent ride sharing matches. Finally, it may in general be possible that flow through a customer (as considered in the first part) will become fractional – in this case, we just continue adding branches as outlined in (4.34).

While algorithmic outlines of Branch and Price schemes are useful, their actual performance relies on many implementation specific details. One such factor is the amount of solve time allowed for the sub–problems. Better solutions will give variables with a more negative reduced cost in the RMP, which should lead a greater improvement. However, this requires additional computational time and also uses (outdated) dual information from the previous iteration. This suggests an alternative strategy of initially generating new routes that are "good enough" and only investing further computational effort when close to the optimal solution (of the RMP). This can be achieved by using heuristics for the sub–problems or by simply terminating an exact algorithm once an acceptable solution has been found. Similar statements can be made about solving the RMP each iteration; an optimal solution provides the sub–problems with more accurate dual information, but uses a restricted set of columns and requires additional computational effort. Another consideration is the number of solutions to be returned by each sub–problem; returning more may result in fewer iterations, but increases the computational difficulty of the sub–problems and the RMP. Finally, Branch and Price methods can be warm started with solutions found by heuristics (that may require tuning themselves) – but highly optimized solutions may actually increase solve times if they differ greatly to the global optima. We can imagine such a case arising from building dedicated driver schedules around the "wrong" initial matching. For all of these considerations, the correct decision can only be determined through extensive computational experiments which measure run time under different settings (and any findings only hold for instances similar to those trialled).

4.6 A Heuristic Algorithm

It is unlikely that we will be able to solve large-scale instances using the exact methods presented above, so we also present an effective and efficient Variable Neighborhood Search (VNS) meta–heuristic.

4.6.1 Construction

As an overview, our solution construction procedure involves first finding a *maximal* matching between ad hoc drivers and riders and then use a simple insertion heuristic to route

the dedicated drivers amongst the *remaining* riders. Finally, we repeat the procedure with alternative matchings to create a range of initial solutions.

The use of a *maximal* matching (e.g., one that includes as many riders as possible) arises from the observation that using dedicated drivers generally incurs a cost, so it seems sensible to minimize their use. To schedule dedicated drivers amongst the unmatched riders, we use a *sequential* insertion heuristic (for full details, see Algorithm 3 below). Specifically, for each rider we determine the cheapest insertion into the schedule for the current driver and effectuate the cheapest among them. Once there are no more feasible insertions into the route of the current driver, we start a route for a new driver. We continue adding new drivers until the minimum number of riders are served. Unlike most construction heuristics, we choose not to initialize routes with a seed customer. Without seed customers, insertion heuristics tend to myopically cherry pick the "easy" customers near the depot, making later insertions more expensive. However, in this setting, we only need to visit a subset of customers – so cheap initial insertions are actually advantageous. Of course, there are equity issues surrounding this, as discussed in Section 4.4.3.4.

Algorithm 3: Solution Construction Method

```
Input: Empty set of vehicle schedules V, Set of passengers P
Initialize: P̂ ← P
repeat
    InitializeEmptyVehicleSchedule(v)
    repeat
        FindFeasibleInsertion(P̂, v, foundInsert, c1)
        if foundInsert then
            InsertMinimumCostCandidate(v, c1, c2)
            UpdateRemainingRiders(P̂, c1, c2)
        else
            AppendVehicleRoute(V, v)
    until !foundInsert
until |P̂| = 0
ShuffleOrderOfVehicles(V)
return V
```

As discussed in Section 4.5.2, selecting certain initial matchings can hinder the discovery of the optimal solution. To address this, we repeat the whole meta–heuristic process **nRepeats** times using different initial solutions. To generate new matchings, we add a constraint preventing a deterioration in the number of matches and then modify the objective coefficients to favor unused arcs (and discourage those previously seen). The full scheme is given below:

1. We find an *dummy* maximal matching by setting all objective coefficients to the same value and solving as normal. We then add a constraint enforcing the maximum number of matchings for all subsequent times we solve this problem.
2. The first (of **nRepeats**) matchings is found by resetting the objective coefficients to their original values and resolving the problem. Let the optimal matching be denoted by $x^*_{ij},\ i, j \in R$.

3. Set objective weights $\hat{c}_{ij} = 1 - x^*_{ij} + \delta_{ij}$, where δ_{ij} is uniformly distributed in the range $[\delta_{\mathbf{min}}, \delta_{\mathbf{max}}]$.
4. Resolve the resulting matching problem to produce an alternative matching $\hat{x}$, and save this matching. Set $x^*_{ij} = \hat{x}_{ij},\ i, j \in R$.
5. Repeat Steps 3 and 4 until a sufficient number of distinct matchings is produced, or a maximum number of iterations is exceeded (in our computational study, we used **nRepeats** matchings and **nConsMaxIter** iterations, respectively).

If fewer than **nRepeats** initial matchings were generated, we simply reuse existing ones as required. Even if we do generate **nRepeats** unique matchings, we can't guarantee the degree of difference between them. Because of this, we randomly shuffle the order in which the routes are stored in our solution (similar to what was discussed in Section 3.4.3.2). This affects the order in which neighborhoods from Section 4.6.2 discover new solutions; as these neighborhoods always accept the first improving move, the heuristic's trajectory through the solution space will also be altered.

Table 4.1: Construction parameters used

nRepeats	15
nConsMaxIter	500
$\delta_{\mathbf{min}}$	-0.5
$\delta_{\mathbf{max}}$	0.5

4.6.2 Neighborhood Search

Each of our initial solutions is then improved with a Variable Neighborhood Search (VNS) procedure using two types of improvement routines: those which improve the schedules of dedicated drivers, and those which change the type of driver serving a rider. As is typical, our neighborhoods are ordered and are explored in sequence. When exploring a given neighborhood, the first feasible improvement found is accepted and we immediately revert back to the first neighborhood. Within a specific neighborhood, moves are always trialled in the same order. The algorithm stops when our *shaking* procedure occurs twice, without an improvement in the best known solution. An outline of the VNS scheme is given in Algorithm 4 and descriptions of the neighborhoods used are given in the following sections.

4.6.2.1 Improving the Routes of Dedicated Drivers

The neighborhoods given below take the current set of routes for dedicated drivers and attempt to reduce the associated cost:

- **Transfer a single rider within a route:** Take every rider and try to insert them into every other position in the route for the same dedicated driver.

Algorithm 4: Variable Neighborhood Search

```
Input: A feasible initial solution x'
Initialize: noImproveCounter ← 1, x ← x'
repeat
    k ← 1
    h ← 1
    repeat
        repeat
            foundImprov ← ExploreImprovRouteNeighborhood (h, x, x*)
            if foundImprov = TRUE then
                h ← 1
                x ← x*
            else
                h ← h + 1
        until h = 5
        foundImprov ← ExploreChangeDriverNeighborhood (k, x, x*)
        if foundImprov = TRUE then
            k ← 1
            h ← 1
            x ← x*
        else
            k ← k + 1
    until k = 2
    if CostOfSolution(x) < CostOfSolution(x') then
        noImproveCounter ← 1
        x' ← x
    else
        noImproveCounter ← noImproveCounter + 1
    if noImprovCounter < 2 then
        x' ← ShakeSolution (x')
until noImproveCounter = 3
return x'
```

- **Swap a pair of riders within a route:** Take every pair of riders in the same route and try to swap their positions.
- **Transfer one rider between routes:** Take every rider and try to insert them into every position in every other route.
- **Swap a pair of riders between routes:** Take every rider and try to swap their position with all other riders in every other route.
- **2–Opt:** Take every sequence of riders in the same route and try to reverse the order in which the riders in the sequence are served.

Note that none of these neighborhoods changes the number of riders served and none of the neighborhoods allow an increase in the cost of the routes. Of course, it is possible that after the third neighborhood we have an empty route, in which case we simply remove the corresponding vehicle.

4.6.2.2 Changing the Type of Driver Serving a Rider:

To diversify the neighborhood search, we employ neighborhoods that attempt to change the type of driver serving a rider, i.e., a rider served by a dedicated driver will instead be served by an ad hoc driver and vice versa. These neighborhoods are explored after those given in Section 4.6.2.1, but before the shaking step given in Section 4.6.2.3.

- **Changing the type of driver used:** For each ad hoc driver d currently matched with a rider r, we create the set of riders R_d which are currently served by a dedicated driver, but could be served by driver d. For each rider $\hat{r} \in R_d$, we determine if it is feasible for the dedicated driver to serve r instead of rider $\hat{r}$ and if the cost of routing the dedicated driver will decrease. We perform the swap as soon as we identify a pair of riders r and $\hat{r}$ for which both conditions are satisfied.
- **Changing the ad hoc driver used:** Alternatively, for each ad hoc driver d currently not matched, we create the set of riders S_d which are currently served by another ad hoc driver, but which could be served by driver d. If $S_d \neq \emptyset$, we choose one of these riders randomly and perform the swap. Even though such a swap does not affect the cost associated with employing dedicated drivers, it perturbs the solution and allows the previous neighborhood to discover new solutions. Consequently, reverting straight to the interior neighborhoods (that improve the routes of dedicated drivers) would be pointless – so a successful iteration of this neighborhood is always followed by a call to the first neighborhood described in this list. Note that because the initial matching has maximum cardinality, there is no rider $\bar{r} \notin S_d$ who can feasibly be matched with driver d.

4.6.2.3 Shaking

Even though swapping drivers creates diversity, it is sometimes advantageous to modify a solution more drastically, to allow a fuller exploration of the solution space. Shaking is performed once a solution is (locally) optimal with respect to all neighborhoods. After

shaking, the other neighborhoods are applied again in the same manner. We stop when we shake a solution twice without achieving an improvement in the best known solution.

Our shaking procedure uses the concept of "ruin and recreate", where we destroy routes for dedicated drivers by deleting riders and then rebuild them by inserting riders. Destruction is controlled by two parameters: $0 \leq$ **deleteFraction** ≤ 1, which specifies the fraction of riders (served by dedicated drivers) that are to be deleted, and $0 \leq$ **probLargest** ≤ 1, which specifies the probability that the (next) rider to be deleted gives the largest reduction in routing cost (otherwise, they are chosen at random). Rebuilding is controlled by two parameters: $0 \leq$ **probIncludeDeleted** ≤ 1, which specifies the probability that we allow riders which were deleted in the destruction phase to be considered for reinsertion, and $0 \leq$ **probLeast** ≤ 1, which specifies the probability that we perform the cheapest feasible insertion (rather that the first feasible insertion). The parameters values used were selected through a small pilot study and are listed in Table 4.2.

Table 4.2: Shaking parameters used

deleteFraction	0.3
probLargest	0.9
probIncludeDeleted	0.5
probLeast	0.9

4.6.2.4 Adaption for Different Objectives

In the case we simply wish to serve as many customers as possible (given some fixed budget), we only need to slightly modify the search procedure given above. First, we note that reducing the routing cost is still very important, as it increases the budget available for the insertion of new customers. For this reason, we still use a minimum cost objective when using our neighborhood search routines, but attempt the insertion of a new customer after every 20 improving moves. Specifically, we attempt to insert all riders (who are not currently served by any driver) into every position in the route of every dedicated driver and select the cheapest feasible insertion that does not violate the budget constraint.

When evaluating two solutions in the shaking step, we use a lexicographical ordering based on the maximum service objective and the minimum cost objective. As an example, suppose we have two solutions (each consisting of a matching and a set of routes). If one solution serves a greater number of riders, then it is said to have a higher objective value, regardless of the cost. If both serve the same number of riders, the cheaper one is better (at least with respect to the objective value). If we considered only the maximum service objective during the shaking step, we would likely terminate the search earlier and have less opportunity to improve known solutions.

4.6.3 Maximum Service Objective

We will now more fully discuss the idea of simply maximizing the number of riders served. The proposal does have merit; it would avoid allegations of profiteering, allow maximum involvement and would strengthen the case for public subsidies. However, its effect on the problem's computational difficulty is unclear. Given a solution, we can slightly perturb it without violating the budget constraint (e.g., by swapping the order of two riders) and without changing the number if riders served; consequently the objective space is comprised of a series of flat plateaus. Alternatively, we could say this structure arises from the homogeneity of the objective coefficients. We would expect this clustering to occur around all solutions, even globally optimal ones. Additionally, we would expect there to be multiple clusters of global optima, potentially with significantly different solutions. Flat solution spaces can be difficult to explore (as there's no objective gradient to guide the search), but the presence of multiple clusters of optimal solutions should make it easier to find one.

Of course, solutions under both the minimum cost and maximum service objectives require efficient routing decisions, as these either reduce the total cost or allow greater service from the same fixed budget. Consequently, solutions that perform well under one objective likely perform well under the other. To see this, suppose we have a solution S_1 which is optimal under the first objective, i.e., it meets some service level $\hat{s}$ at minimum cost. If we impose a maximum service objective and enforce a budget constraint equal to the minimum cost, then S_1 is not guaranteed to remain optimal under the new objective. That is, there may be multiple solutions with the same cost that serve different numbers of passengers. However, S_1 is likely to be a high quality solution in most real world cases. Similarly, given a solution S_2 which is optimal under a maximum service objective and satisfies some budget, we can add a constraint preventing a reduction in the service level and switch to an objective that minimizes cost. We expect S_2 to be of a high quality (as cheap routes enable the best effect from limited funds), but can't guarantee optimality (due to the "many–to–one" mapping between the two objective spaces).

In this section we detail a preprocessing procedure that exploits the presence of multiple optimal solutions (under a maximum service objective) to reduce the network size and the computational effort required to find a solution. This procedure consists of two rules which reduce the solution space by fixing matches that most likely exist in (at least) one optimal solution. Once a matching is fixed, the participants are removed from the network and not given further consideration. The application of these rules can prevent the discovery of an optimal solution, but we have strong empirical evidence suggesting that such cases are very rare (see Section 4.8.1.1), indicating that this procedure represents an effective way to focus the search. Our two rules are presented below:

Preprocessing Rule One: If an ad hoc driver may only service one rider, they are matched together.

Preprocessing Rule Two: If a rider may only be serviced by one ad hoc driver (and any number of dedicated drivers), they are matched together.

To see that the first rule can result in suboptimal solutions, recall that earlier discussion

concerning the network in Figure 4.3 showed that if the service provider can reinvest the fares earned by dedicated drivers, the addition of certain riders to a dedicated driver's route can actually reduce overall costs (i.e., offer a marginal profit). Under a fixed budget, it is conceivable this additional income would allow a hypothetical fourth customer to be served (where there would otherwise be insufficient funding to do so). If the first rule was followed, then the ad hoc driver and rider R2 would have been paired together, preventing the service of the new customer and excluding the optimal solution. We note this is a consequence of allowing rider fares to fund subsequent dedicated drivers, and could be avoided by imposing the budget on the *operational* cost (and not the *net* cost). Alternatively, the service provider may feel it is good practice to involve as many ad hoc drivers as possible (even at the expense of optimality), in order to maximize engagement and avoid profiteering allegations.

To see that the second rule can yield suboptimal solutions, consider the networks given in Figure 4.5. Assume that a dedicated driver (who leaves from the depot) is able to serve riders R2 and R3 in a single shift, but if they serve R1, they may not serve any other rider. Also, assume that the ad hoc driver can serve either rider R1 or R2. Under these constraints, there is a single optimal solution, as given in Figure 4.5(a). However, the application of our second rule would force the pairing of the ad hoc driver and rider R2, which leads to the sub–optimal solution given in Figure 4.5(b). As our rules would finalize these matches at the start of the process, our search routines would not have an opportunity to rectify this mistake. However, Section 4.8.1.1 presents computational experiments showing that in practice, this occurs rarely. [1]

The above mentioned caveats notwithstanding, we will use the networks given in Figure 4.6 to illustrate the effectiveness of these rules. If we consider the first network in Figure 4.6(a), there are clearly three optimal matchings. The application of Rule Two matches the first driver with the first rider, and subsequent application of Rule One matches the second driver and second rider. In larger problems, we may need to apply the rules iteratively, and will likely obtain a remaining sub–network to which we can apply our VNS meta–heuristic (again, matched participants are excluded from further consideration). In the network given in Figure 4.6(b), even after pairing the third and fourth drivers with the fourth and fifth riders respectively, we obtain a sub–network that our preprocessing rules cannot reduce any further.

Of course, the order in which the two rules are used and the sequence in which participants are considered affects which pairs are found. To exploit this, when we choose to use a maximum service objective, we slightly modify the construction of our initial solutions. We always apply the preprocessing rules identified above to the original network (for each repeat, before doing anything else), but we consider participants in a random order (leading to different sub–networks being discovered). We then find a maximum cardinality matching between riders and ad hoc drivers on the remaining sub–network, where

[1] An interesting variant of this rule was proposed by a Doctoral Examiner. Specifically, set $x_{ij} = 0, i \in D, j \in R$ if rider j can be served by other ad-hoc drivers, and there exists another rider $k \in R$ that can only be served by ad hoc driver i. Essentially, this rule means that if an ad hoc driver has some riders that only they can serve, then they must serve one of them; but which one is decided in the main optimization stage. This variant is weaker, (that is, it removes fewer feasible solutions), but is guaranteed to never remove the optimal solution. In certain settings, you may need to ensure that rider k can never represent a net profit if served by a dedicated driver.

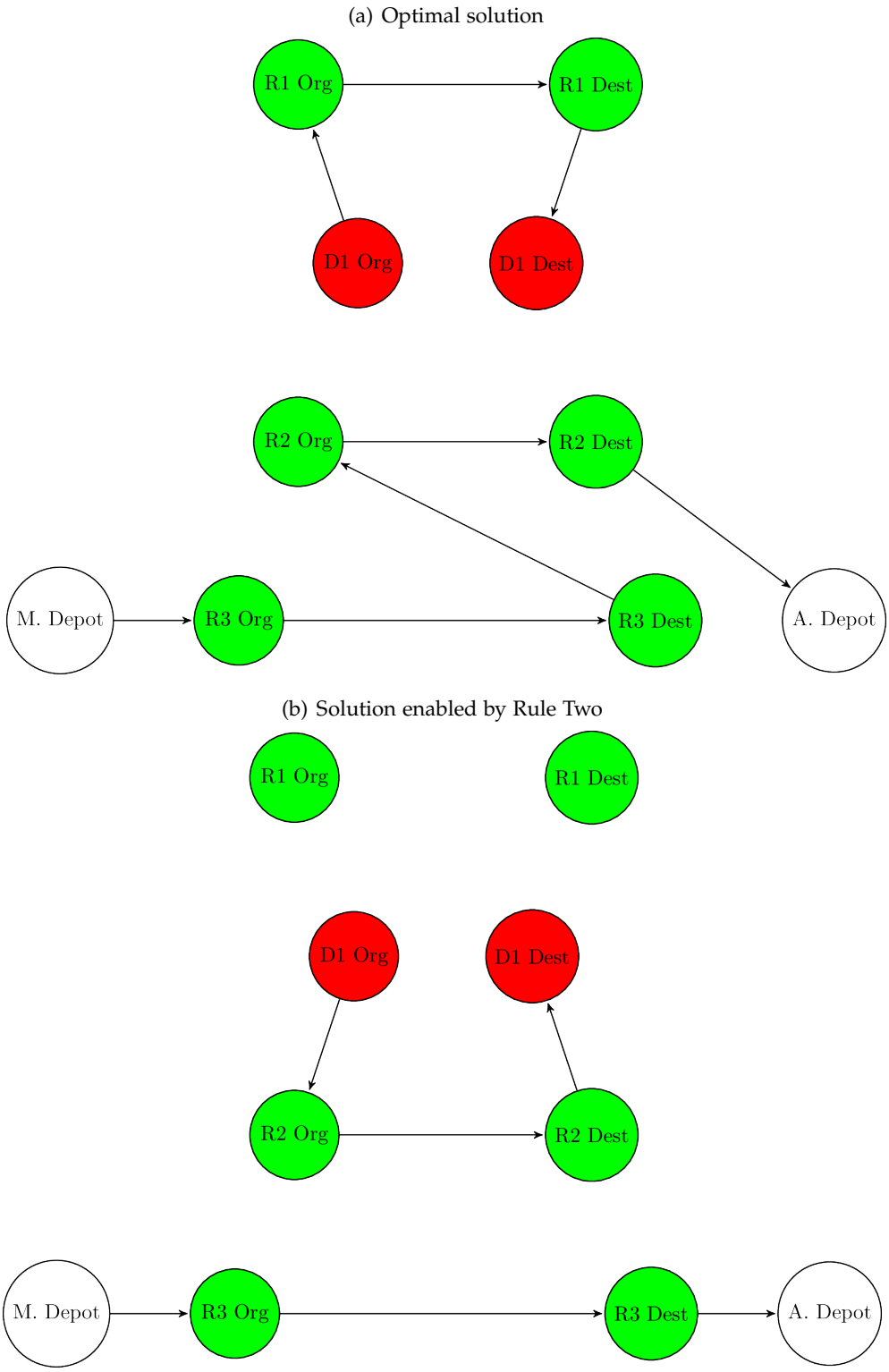


Figure 4.5: Counter example for Rule Two

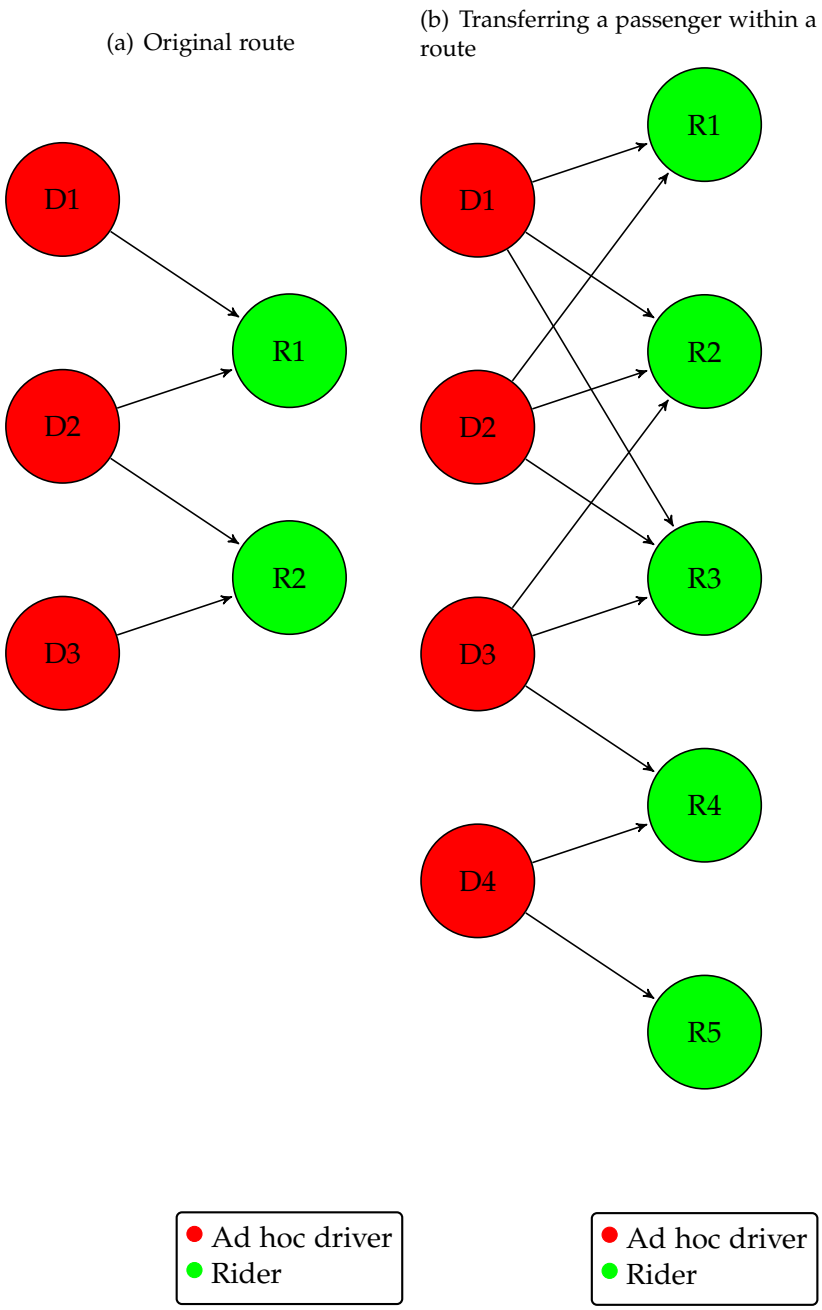


Figure 4.6: Moving passengers within a route

the objective function is still modified based on previous matchings, as discussed in Section 4.6.1. Routes for dedicated drivers are then constructed using the insertion heuristic also outlined in Section 4.6.1.

Our preprocessing rules will certainly preclude the discovery of certain solutions by removing them from the solution space, but will (almost) never remove all optimal solutions. This gives rise to an interesting trade–off in our heuristic. Essentially, we are hoping that by restricting ourselves to a smaller, less diverse solution space (where fewer decisions must be made), we can focus our search in an efficient manner (and find an optimal solution faster). Computational results showing the benefits of this procedure are given in Section 4.8.1.1.

Of course, these network reduction techniques can also be used to speed up the IP given in Section 4.5.1. Indeed, as the IP tends to have a solve time that grows faster than the heuristic's (and becomes intractable with a smaller number of participants), these rules may have a greater effect when applied to our IP.

4.7 Instance Generation

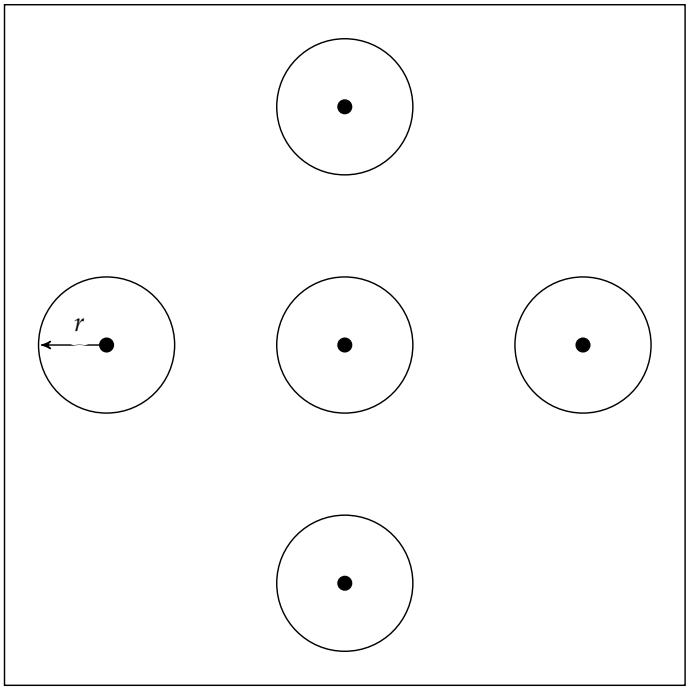
To contextualize the potential costs and benefits associated with dedicated drivers, we need to understand the characteristics of the network being studied. We will now explain how our instances were constructed and the steps we took to mimic real life networks.

The service area of the ridesharing system will be a rectangular geographical region of length l and width w. Specifically, the region used in our computational experiments is a 25 kilometer $\times$ 25 kilometer square, and our vehicles are assumed to travel at a constant speed of 50 kilometers per hour (with negligible time spent collecting and delivering passengers).

To be able to analyze the impact of participants' travel patterns, we investigate three different distributions of participants' origins and destinations: uniform random (in the service area), centered around five hubs and centered around two hubs (see Figure 4.7). In the two latter settings, a participant's origin and destination location is chosen (uniformly) at random within radius r of a hub, with *different* hubs for the origin and destination. When using hubs, travel between each pair of hubs is equally likely. For ease of reference, settings involving hubs may be said to include "corridors", because in the real world such settings typically have transport corridors connecting the hubs.

Time windows for a participant $i \in R \cup D$ are generated as follows. We draw the latest arrival time, l_i^{Arr} at the destination from a truncated normal distribution with a mean of 240 minutes and a standard deviation of 45 minutes. Times outside of this four and a half hour period (i.e., more than 3 standard deviations from the mean) are discarded, and a new one is generated. The implied latest departure time is given by $l_i^{\text{Dept}} = l_i^{\text{Arr}} - \hat{t}_i$ (again, $\hat{t}_i$ is the travel time from the participant's origin to their destination). Similarly, the implied earliest arrival and earliest departure times are given by $e_i^{\text{Arr}} = l_i^{\text{Arr}} - t_i^{\text{Flex}}$ and $e_i^{\text{Dept}} = l_i^{\text{Dept}} - t_i^{\text{Flex}}$, respectively, where t_i^{Flex} is the participant's travel flexibility, i.e., will-

(a) Five Hubs



(b) Two Hubs

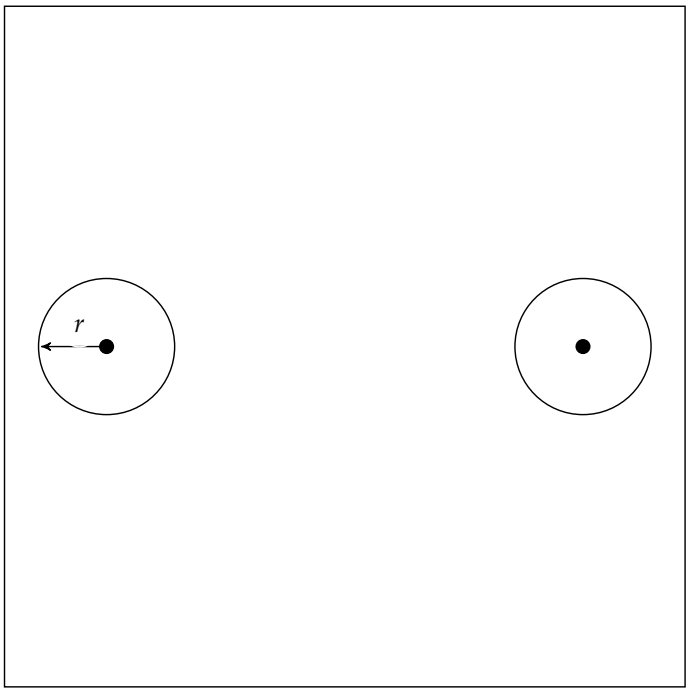


Figure 4.7: Diagram of networks with hubs

ingness to deviate from their desired travel times. t_i^{Flex} is drawn from a normal distribution with a parameterized mean (given below) and a standard deviation of 4 minutes; this is again truncated beyond 3 standard deviations.

As we are most interested in exploring links between the level of service provided and the expense of operating dedicated drivers, we have chosen to simplify the cost structure used. We assume the service provider must pay some flat fee per mile traveled by a dedicated driver, and assume the provider simply wants to minimize this (subject to some service constraint). For simplicity, we assume this is $1 per mile. We neglect any consideration of commissions received, as when a minimum service level is imposed, all feasible solutions likely receive the same amount of commissions (assuming only the minimum number of riders are served).

A class of instances is defined by three parameters, whose definitions and values used in the computational study are given in Table 4.3. With respect to participants, we note two things. First, each is equally likely to be a driver or a rider. Secondly, if two instances differ only in the number of participants, then the set of participants in the smaller instance is a proper subset of those in the larger instance, e.g., to create instances with more participants, we keep those already present and only generate the number required to makeup the shortfall.

Table 4.3: Parameters defining classes of instances

Parameter	Definition	Values used
Number of Participants:	The number of participants.	$100, 200, \ldots, 500$ participants
Spread of Participants:	Dispersal of origins and destinations.	Uniform random, 2 hubs, 5 hubs
Mean Flexibility:	The mean time flexibility of participants.	24 and 36 minutes
Number of Instances:	The number of instances in an instance class.	10 instances

To gain an understanding of the trade–off between achieving a target service level and the associated cost of employing dedicated drivers, we explore three target service levels: 90%, 95%, and 98%. Of course, when changing the required service level, all other instance characteristics are held constant.

4.8 Computational Study

4.8.1 Heuristic Validation

To be able to meaningfully investigate the performance of a ridesharing system, we need to be confident that our heuristic performs well. Illustrating this was done in two stages; we first tuned the various heuristic parameters used by our algorithm, and then compared its performance against exact solution methods. Note that we used the same set of moderate sized test problems for both stages.

To conduct our pilot study, we identified the parameters we believed to be most important to overall heuristic performance. These have been introduced in Sections 4.6.1 and 4.6.2.3

and are repeated in the Table 4.4 below. Please note a bold value indicates what was ultimately selected.

Table 4.4: Values of heuristic parameters trialed

Parameter	Values used
nRepeats	10, **15**, 20
nConsMaxIter	100, **500**, 1000
deleteFraction	0.1, **0.3**, 0.5, 0.7, 0.9
probLargest	0.7, **0.9**, 1
probIncludeDeleted	0.3, **0.5**, 0.7
probLeast	0.7, **0.9**, 1

When reviewing the performance results, there were two important considerations: the time required for completion, and the quality of the best solution found (measured by the objective value). Not surprisingly, we found that the **nRepeats** parameter had the biggest impact on run time, and values above 15 offered no improvement in solution quality. The value of **nConsMaxIter** did not meaningfully affect run time (as the construction phase is very fast) and values above 500 did not improve the final solution. Extreme values of **deleteFraction** significantly impacted run time, as this controls the fraction of customers removed at each shaking step. Removing too many customers destroys any desirable structure present, worsening the solution and increasing the time needed achieve convergence; conversely, removing too few customers means we only explore a small fraction of the solution space and quickly converge to a low-quality solution. We found that a value of 0.3 gave the best solutions, and gave shorter run times than higher values. We found the last three parameters had only a modest impact on run time and performance. This is because these parameters control which customers are selected for removal/insertion in the shaking step; this is done to introduce diversity into the algorithm, and "wrong" decisions here can easily be corrected through subsequent improvement steps.

For both the tuning runs (above) and the validation runs (below), we created instances with 50 participants and a minimum service level of 98% (as this arguably represents the toughest scenario); all parameters other than the number of participants were as described in Table 4.3.

Next, we want to compare the quality of our heuristic solutions with those obtained by using the IP formulation presented in Section 4.5.1. For these problems, the maximum number of vehicles available is set to one more than the number needed in the heuristic solution. Although this restriction could theoretically remove the optimal solution (in case the optimal solution used two more vehicles than the heuristic solution), we feel this will happen very rarely. Additionally, allowing more vehicles would increase the solution space and slow down solve times, limiting the size of problems we can study. The results can be found in Table 4.5. For clarity, as these experiments were done with a Minimum Cost Objective, we did not apply our preprocessing techniques discussed in Section 4.6.3; for reasons given above, we believe this is only applicable under a Maximum Service Objective.

We see that the heuristic performs very well, with the average difference in objective value

Table 4.5: Comparison of heuristic solutions and optimal solutions

	Uniform Distribution	Five Hubs	Two Hubs
Average difference in objective value	0.84%	0.72%	0.04%
Maximum difference in objective value	2.04%	3.15%	0.28%
Average difference in vehicles used	0.3	0.3	0.1
Maximum difference in vehicles used	1	1	1
Instances where Opt was achieved	2/10	5/10	8/10

being under 1% in all configurations and being at most 3.15%. The heuristic achieved optimality in half of the thirty instances and used an extra vehicle (compared to the IP) five times. We believe this shows that our heuristic approach produces high quality solutions and that using it to investigate characteristics of ridesharing systems is appropriate.

4.8.1.1 Maximum service objective

In Section 4.6.3 we introduced a preprocessing procedure for the case with a maximum service objective and claimed that it could significantly reduce network size, but may (in very rare instances) remove all optimal solutions. To show the first part of our claim, we apply this procedure to a wide range of networks and count the pairs of participants removed. Our instances were formed using the methods and parameters given in Section 4.7, except that the number of participants came from the set $\{50, 100, 150, \ldots, 1000\}$, so we could judge the procedure's performance on larger instances. We repeated the analysis **nRepeats** times for each instance, shuffling the order in which participants are considered. Results are given in Table 4.6, with values averaged across all repeats and parameter values (other than the number of participants).

Table 4.6 shows that our rules can remove a large proportion of participants and allow a significant reduction in our heuristic's run time. It appears that the number of participants removed eventually levels out at around 65 – 75 pairs. This is probably because once a network reaches a certain size, any new participant is as likely to be a candidate for removal as it is to block another participant being removed. The particular value converged to is a function of the various network characteristics (such as the length of time windows, the distribution of origins and destinations, etc).

As discussed previously, there is a fair concern that our procedure will remove the optimal solution. One way to test this is to run our meta–heuristic both *with* and *without* the preprocessing procedure, and check for a difference in the objective values. For this purpose, we will select the best value obtained over all 15 repeats. To perform this experiment, we need a budgetary constraint (note this was not required when performing the preprocessing procedure). To find our budget, we run the meta–heuristic with a minimum cost objective and increase the cheapest cost found by 5% (we use a service level of 90% and restart the search 15 times). After reverting to a maximum service objective, we found that there were only two instances in which applying our preprocessing procedure was associated with our meta–heuristic finding a worse solution. In both cases, the instances had 550 customers and the difference between the objectives was a single matching. Of course, we don't know if the preprocessing truly removed the optimal solution (simply removing most occurrences would make it harder for the heuristic to discover the optimal solution).

Table 4.6: Average number of pairs of participants removed from the problem

Number of participants	Matches found	Percent of participants matched	Number of iterations	Reduction in solve time
50	8.10	32.40%	1.30	47.50%
100	20.55	41.10%	1.75	50.40%
150	35.00	46.67%	2.55	47.22%
200	51.10	51.10%	3.60	59.40%
250	62.35	49.88%	4.20	64.32%
300	65.80	43.87%	4.45	56.25%
350	66.65	38.09%	4.35	39.46%
400	65.65	32.83%	4.50	42.35%
450	68.95	30.64%	4.65	45.21%
500	65.15	26.06%	3.95	26.71%
550	74.05	26.93%	5.05	28.19%
600	69.70	23.23%	4.15	34.20%
650	69.00	21.23%	4.05	29.15%
700	63.30	18.09%	3.10	34.62%
750	62.05	16.55%	3.00	20.37%
800	62.75	15.69%	3.05	16.93%
850	63.95	15.05%	3.10	26.86%
900	63.55	14.12%	3.05	19.73%
950	65.40	13.77%	3.05	21.70%
1000	67.60	13.52%	2.95	13.86%

Considering there were 1200 instances, a small deterioration in the objective value of two may be a fair price for such a significant reduction in solve time.

This is the last time we will consider a maximum service objective and for the rest of the chapter we will assume our intention is to minimize net cost.

4.8.2 Computational Results

We start by determining for each of the different network configurations what fraction of riders can be served by ad hoc drivers. This gives an initial indication of the "effort" required to attain a certain service level. The results can be found in Table 4.7, where the results represent averages over the ten instances formed by this combination of parameters. The number in parenthesis indicates how many riders this represents, with the remainder requiring service by a dedicated driver (for reference, Table 4.8 gives the minimum number of riders that must be served at each service level).

Table 4.7: Fraction of riders that can be served by ad hoc drivers

Num Participants	Num Riders	Uniform Dist	Five Hubs	Two Hubs
100	50	34.20% (17.10)	58.80% (29.40)	85.20% (42.60)
200	100	52.20% (26.10)	74.70% (37.35)	90.10% (45.05)
300	150	58.87% (29.43)	83.40% (41.70)	92.80% (46.40)
400	200	64.10% (32.05)	88.20% (44.10)	94.05% (47.03)
500	250	68.28% (34.14)	90.92% (45.46)	95.52% (47.76)

Table 4.8: Minimum number of riders that must be served

Num Participants	Num Riders	Service Level		
		90%	95%	98%
100	50	45	48	49
200	100	90	95	98
300	150	135	143	147
400	200	180	190	196
500	250	225	238	245

As expected, the fraction of riders that can be served by ad hoc drivers increases when there are more riders and journeys are concentrated along common corridors (i.e., when participants travel between hubs). What may be more surprising is the magnitude of the difference caused by these corridors. With 500 participants and a service level of 95%, we do not need any dedicated drivers when journeys are between two hubs, but we need to serve more than 25% of riders with dedicated drivers when journeys are uniformly distributed.

Next, in Table 4.9, we present performance statistics relating to the use of dedicated drivers in the different ridesharing system configurations. Specifically, we give the total cost (and in parentheses the per-rider cost), the number of dedicated drivers required, and the percentage of riders served by a dedicated driver. Occasionally, no dedicated drivers were

needed to serve riders; Table 4.10 gives the number of instances (out of 10) for which this happened.

Table 4.9: Performance statistics of dedicated drivers

Uniform Distribution									
	Total Cost (Per–rider Cost)			# Ded Drivers			% Served by Ded Drivers		
	90 %	95%	98%	90 %	95%	98%	90 %	95%	98%
100	527.11 (18.90)	622.21 (20.15)	658.27 (20.65)	5.90	6.90	7.10	62.00%	64.38%	65.10%
200	631.92 (16.73)	769.00 (17.97)	862.89 (18.85)	7.30	9.10	10.10	42.00%	45.05%	46.73%
300	728.62 (15.56)	942.18 (17.20)	1059.62 (18.02)	8.60	10.90	12.20	34.59%	38.25%	39.93%
400	759.99 (14.62)	991.33 (16.00)	1158.35 (17.04)	8.60	11.50	13.70	28.78%	32.53%	34.59%
500	754.09 (13.84)	1037.52 (15.37)	1246.87 (16.74)	8.90	12.10	14.00	24.13%	28.28%	30.33%
Five Hubs									
	Total Cost (Per–rider Cost)			# Ded Drivers			% Served by Ded Drivers		
	90 %	95%	98%	90 %	95%	98%	90 %	95%	98%
100	224.59 (14.32)	289.70 (15.55)	312.83 (15.94)	3.10	4.00	4.30	34.67%	38.75%	40.00%
200	193.16 (12.56)	279.21 (13.74)	336.72 (14.45)	3.00	4.00	5.00	17.00%	21.37%	23.78%
300	120.15 (11.74)	244.67 (13.53)	323.63 (14.68)	2.00	4.00	5.00	7.33%	12.52%	14.90%
400	58.51 (12.13)	169.73 (12.40)	276.56 (14.14)	1.29	2.80	4.80	2.11%	7.16%	10.00%
500	47.08 (10.56)	121.46 (11.40)	229.76 (13.08)	1.50	2.10	3.90	0.40%	4.50%	7.22%
Two Hubs									
	Total Cost (Per–rider Cost)			# Ded Drivers			% Served by Ded Drivers		
	90 %	95%	98%	90 %	95%	98%	90 %	95%	98%
100	49.62 (22.03)	110.31 (20.23)	137.97 (21.47)	1.10	1.90	2.60	5.33%	11.25%	13.06%
200	140.35 (23.39)	130.55 (23.57)	184.51 (22.96)	1.67	1.89	3.00	2.00%	5.26%	8.06%
300	122.07 (26.76)	179.29 (24.06)	191.47 (23.38)	1.50	2.33	2.80	0.67%	2.94%	5.31%
400	107.59 (26.90)	136.17 (24.12)	195.89 (24.47)	1.00	1.83	2.60	0.22%	1.74%	4.03%
500	25.52 (25.52)	237.39 (23.00)	176.43 (24.03)	1.00	3.00	2.22	0.04%	0.80%	2.65%

Table 4.10: Number of instances for which no dedicated drivers were needed

	Uniform Distribution			Five Hubs			Two Hubs		
	90%	95%	98%	90%	95%	98%	90%	95%	98%
100	-	-	-	-	-	-	-	-	-
200	-	-	-	-	-	-	7	1	-
300	-	-	-	-	-	-	8	4	-
400	-	-	-	3	-	-	9	4	-
500	-	-	-	8	-	-	9	8	1

For every system configuration, there is a density after which the number of required dedicated drivers to guarantee a certain service level starts to decrease. However, we observe that when the participants are uniformly distributed, this density has not yet been reached even when there are 500 participants (the number of required drivers is still going up). However, in the configuration with five hubs, even when the required service level is 98%, when the number of participants surpasses 300, the number of dedicated drivers needed starts to decrease (and so do the total costs). It is also interesting to observe that the per-rider cost decreases when the number of participants increases. When the number of participants is small, the riders will be more dispersed and a dedicated driver has to drive a larger distance between consecutive riders served, which leads to an increase in the per-rider cost.

In Table 4.11, we show the average cost incurred per rider served by a dedicated driver, and the average distance between a rider's origin and destination depending on the type of driver who serves them. As the cost of employing a dedicated driver is proportional to the distance they travel, the difference between the per-rider cost and the average distance between a rider's origin and destination when served by a dedicated driver estimates the distance a dedicated driver travels empty (between serving riders). For completeness sake, we also give the average origin-destination distance over all riders (served and unserved).

We see that riders with longer journeys tend to be served by ad hoc drivers, while riders with shorter journeys tend to be served by dedicated drivers. This is because shorter journeys are both cheaper and allow dedicated drivers to serve more riders in a given time frame. Also, as the magnitude of the cost savings increase with trip length, matching riders with longer journeys with ad hoc drivers also benefits the participants. We observe too that the average origin-destination distance of riders served by a dedicated driver decreases when the density increases, because there are more choices and the dedicated drivers can more easily select riders with short origin-destination distances. We also observe that, as expected, the average origin-destination distance for riders served by dedicated drivers increases as the required service level increases, because the dedicated drivers are forced to serve more expensive riders.

The per–rider costs reflect the observations made above, except for the configuration with two hubs. There are several reasons for this. First, when a dedicated driver serves only a few riders (which happens frequently in the configuration with two hubs, especially with a large number of participants), the distance from and to the home base of a dedicated driver starts to have a disproportionately large impact on the total cost and thus the per–rider cost (note the home base of the dedicated drivers is not in one of the hubs). Additionally, at higher service levels, some riders can only be served by a dedicated driver

Table 4.11: Average distance between a rider's origin and destination

	Avg Cost per Rider			Avg O/D Dist Ad hoc			Avg O/D Dist Ded Driver			Avg O/D Dist
	90 %	95%	98%	90 %	95%	98%	90 %	95%	98%	
Uniform Distribution										
100	18.90	20.15	20.65	12.43	12.41	12.38	11.22	12.14	12.50	12.73
200	16.73	17.97	18.85	13.45	13.42	13.47	9.96	10.91	11.62	12.85
300	15.56	17.20	18.02	13.31	13.43	13.41	9.49	10.59	11.33	12.84
400	14.62	16.00	17.04	13.52	13.75	13.69	8.81	9.84	10.87	12.95
500	13.84	15.37	16.74	13.59	13.78	13.74	8.23	9.48	10.59	13.00
Five Hubs										
	Avg Cost per Rider			Avg O/D Dist Ad hoc			Avg O/D Dist Ded Driver			Avg O/D Dist
	90 %	95%	98%	90 %	95%	98%	90 %	95%	98%	
100	14.32	15.55	15.94	10.15	10.15	10.13	9.67	10.11	10.30	10.29
200	12.56	13.74	14.45	10.37	10.37	10.43	9.28	9.84	10.04	10.44
300	11.74	13.53	14.68	10.27	10.33	10.30	8.20	9.19	10.02	10.37
400	12.13	12.40	14.14	10.26	10.31	10.34	8.16	8.78	9.68	10.34
500	10.56	11.40	13.08	10.30	10.38	10.41	8.32	8.14	9.32	10.39
Two Hubs										
	Avg Cost per Rider			Avg O/D Dist Ad hoc			Avg O/D Dist Ded Driver			Avg O/D Dist
	90 %	95%	98%	90 %	95%	98%	90 %	95%	98%	
100	22.03	20.23	21.47	15.06	15.08	15.10	13.54	14.01	14.06	14.99
200	23.39	23.57	22.96	15.04	15.08	15.10	13.76	13.67	13.98	15.01
300	26.76	24.06	23.38	15.05	15.08	15.11	13.32	13.71	13.58	15.03
400	26.90	24.12	24.47	15.05	15.08	15.10	13.45	13.40	13.52	15.05
500	25.52	23.00	24.03	15.04	15.06	15.09	12.75	13.72	13.31	15.04

by first traveling empty between the hubs, which leads to a, relatively speaking, huge extra cost. Finally, note that the average journey length is greater for the "Two Hub" system (compared with the "Five Hub" system), which naturally leads to higher costs.

To provide additional insight into the consequences of imposing a minimum service requirement, Table 4.12 gives the reduction in total cost resulting from lowering the required service level. For some instances, there was no need for dedicated drivers for both levels of required service, or no need for dedicated drivers in the lower level of required service. These instances were excluded when calculating the percentage cost reduction; in parentheses we show the number of instances used in the calculation.

Table 4.12: Decrease in total cost from lowering the required service level

	Uniform Distribution		Five Hubs		Two Hubs	
	98% → 95%	95% → 90%	98% → 95%	95% → 90%	98% → 95%	95% → 90%
100	5.51% (10)	15.38% (10)	7.53% (10)	22.98% (10)	21.29% (10)	57.29% (10)
200	11.02% (10)	18.00% (10)	17.25% (10)	31.39% (10)	42.24% (9)	57.33% (3)
300	11.10% (10)	22.34% (10)	25.33% (10)	53.38% (10)	41.77% (6)	69.96% (2)
400	14.36% (10)	23.50% (10)	39.84% (10)	71.44% (7)	59.38% (6)	71.93% (1)
500	15.93% (10)	27.57% (10)	48.56% (10)	76.75% (2)	46.97% (2)	93.30% (1)

It appears that decreasing the service level is especially beneficial for configurations that are naturally conducive to ridesharing. However, this may be partly the result of the fact that the costs associated with dedicated drivers is much smaller and even small savings may represent a large relative change.

For an alternative perspective, we report in Tables 4.13 and 4.14 the effect of lowering the required service level on the per-rider cost and on the average origin-destination distance

of the riders served by a dedicated driver. Again, if lowering the service level meant dedicated drivers were no longer required, then the instance was excluded from the analysis.

Table 4.13: Effect of lowering $\hat{s}$ on the average per-rider cost

	Uniform Distribution		Five Hubs		Two Hubs	
	95% → 90%	98% → 95%	95% → 90%	98% → 95%	95% → 90%	98% → 95%
100	2.42% (10)	6.17% (10)	2.48% (10)	7.95% (10)	5.96% (10)	-12.06% (10)
200	4.68% (10)	6.93% (10)	4.84% (10)	8.53% (10)	-8.44% (9)	5.53% (3)
300	4.55% (10)	8.91% (10)	8.09% (10)	13.44% (10)	-0.60% (6)	1.70% (2)
400	6.02% (10)	8.69% (10)	12.20% (10)	3.87% (7)	1.33% (6)	1.75% (1)
500	7.14% (10)	10.12% (10)	12.76% (10)	6.15% (2)	2.44% (2)	6.20% (1)

Table 4.14: Effect of lowering $\hat{s}$ on the average OD distance of riders served

	Uniform Distribution		Five Hubs		Two Hubs	
	95% → 90%	98% → 95%	95% → 90%	98% → 95%	95% → 90%	98% → 95%
100	2.88% (10)	7.63% (10)	1.83% (10)	4.38% (10)	0.33% (10)	3.42% (10)
200	6.19% (10)	8.61% (10)	2.04% (10)	5.57% (10)	1.78% (9)	0.59% (3)
300	6.49% (10)	9.61% (10)	8.61% (10)	10.64% (10)	-0.03% (6)	2.16% (2)
400	8.92% (10)	10.61% (10)	9.43% (10)	10.23% (7)	1.88% (6)	1.92% (1)
500	8.73% (10)	13.84% (10)	11.85% (10)	-0.57% (2)	0.25% (2)	6.22% (1)

We see that lowering the required service level leads to proportionally larger reductions at higher densities. This is because fewer riders are served, so a small reduction can give a large proportional decrease. The trend is noisier for the instances with two hubs, because dedicated drivers are often no longer needed, reducing the number of instances participating in the analysis.

Next, we investigate the effect of increasing the flexibility of participants; more specifically, we increase the average time flexibility of participants from 24 minutes to 36 minutes. The results are reported in Table 4.15, which shows the percentage reduction in the total cost, the number of dedicated drivers required, and the fraction of riders served by dedicated drivers. The increased flexibility resulted in instances where dedicated drivers were no longer required. These instances were not included in the calculation of the percentage reduction. Therefore, we report in parentheses the number of instances included in the calculation of a statistic. Finally, an entry "N/A" indicates that no dedicated drivers were needed in any instance with increased flexibility.

We see that the reduction in total cost increases with density for the configuration with random journeys and for the one with five hubs (but is tapering off). As expected, the extra flexibility can be exploited more effectively when the number of participants is higher, but only up to a point. The situation is different for the configuration with two hubs. This is an environment that is conducive to ridesharing and relatively little effort is required to attain a certain service level. In such environments, the extra flexibility has a much smaller (if any) impact. A similar pattern emerges for the reduction in the number of dedicated drivers needed.

Table 4.15: Percentage reduction resulting from greater flexibility

Uniform Distribution									
	Total Cost			Num of Ded Drivers			% Served by Ded Drivers		
	90 %	95 %	98 %	90 %	95 %	98 %	90 %	95 %	98 %
100	24.71% (10)	22.11% (10)	21.74% (10)	30.90%	24.94%	18.21%	18.31%	16.44%	15.90%
200	37.13% (10)	33.55% (10)	31.60% (10)	35.63%	35.16%	34.34%	27.44%	24.10%	22.47%
300	42.33% (10)	38.76% (10)	36.35% (10)	39.56%	38.93%	34.32%	31.98%	27.22%	25.34%
400	44.67% (10)	39.71% (10)	37.31% (10)	42.30%	37.45%	37.23%	32.87%	27.53%	25.08%
500	45.28% (10)	39.74% (10)	37.64% (10)	40.42%	39.64%	31.83%	33.90%	27.32%	24.74%
Five Hubs									
	Total Cost			Num of Ded Drivers			% Served by Ded Drivers		
	90 %	95 %	98 %	90 %	95 %	98 %	90 %	95 %	98 %
100	32.94% (10)	28.54% (10)	26.86% (10)	32.50%	26.33%	23.33%	31.23%	26.05%	24.70%
200	63.73% (10)	48.34% (10)	41.13% (10)	53.33%	42.00%	36.17%	59.91%	44.80%	38.95%
300	80.93% (4)	66.95% (10)	56.14% (10)	58.33%	57.67%	50.33%	79.43%	61.08%	49.42%
400	N/A	71.24% (7)	61.46% (10)	N/A	57.14%	61.50%	N/A	68.15%	56.74%
500	N/A	77.99% (4)	68.51% (9)	N/A	52.08%	58.52%	N/A	74.89%	63.85%
Two Hubs									
	Total Cost			Num of Ded Drivers			% Served by Ded Drivers		
	90 %	95 %	98 %	90 %	95 %	98 %	90 %	95 %	98 %
100	51.09% (2)	45.47% (9)	42.22% (10)	25.00%	31.48%	46.67%	50.00%	52.90%	47.90%
200	20.09% (1)	42.04% (5)	37.29% (8)	33.33%	17.33%	48.96%	25.00%	44.20%	44.12%
300	12.68% (1)	22.23% (4)	24.56% (7)	0.00%	20.83%	38.10%	12.50%	20.59%	27.96%
400	24.98% (1)	29.07% (4)	25.87% (7)	0.00%	16.67%	20.24%	25.00%	25.83%	27.17%
500	0.00% (1)	19.70% (2)	19.58% (8)	0.00%	25.00%	15.62%	0.00%	20.00%	23.81%

4.9 Conclusion

We have presented the issues surrounding reliability in Dynamic Ride Sharing (DRS) schemes and explained how this can keep participation rates from reaching sustainable levels. We have shown how a return trip can be guaranteed to participants while keeping the problem polynomially solvable with minimal assumptions, reducing solution times and improving the quality of matches. We then proposed a setting where the service provider hires their own drivers to serve riders who would otherwise remain unmatched and discussed various pricing methods to determine an appropriate fare. We considered supplementary revenue sources to offset the cost of these drivers and how these are affected by the service provider's operational objectives. Finally, we discussed how the operator could design the system to encourage desirable behavior from participants.

We then proposed three powerful solution methods capable of handling this problem; an Integer Programming formulation, a Branch and Price algorithm and a Variable Neighborhood Search (VNS) meta–heuristic. Two of these approaches were used in an computational study to illustrate the potential benefits of dedicated drivers across a range of networks. We are likely to see an increase of such innovative delivery models in the future and developing the optimization technology to support and analyze such models will be critical for their success.

Chapter 5

Estimating the Robustness of Logistic Schedules to Future Insertions

5.1 Introduction

Worldwide, many successful delivery firms invest in the optimization of routes used by their delivery vehicles. In some industries (such as food manufacturing) customers want regular deliveries (as goods have a limited shelf life) at recurring times (to make planning easier). As a consequence, delivery firms serving this industry may decide to run the same fixed routes on a regular basis. Once designed, these routes can only be changed during infrequent, periodic reviews, which may only occur annually. However, between such reviews businesses still attract new customers, who must be served without changing existing routes, i.e., they may only be added through a simple *insertion* mechanism. The challenge faced by fleet managers is to design low cost routes that can still accept these new customers. In this chapter, we formulate this problem and present a heuristic designed to create routes under this uncertainty. It is an analytical geometric process that returns a numerical robustness "score" proportional to the area from which new customers can be accepted. An important feature of the score is its efficiency; it is computed without simulation or statistical analysis, allowing for computationally inexpensive evaluation. We present experiment results showing the correlation between a solution's score and its robustness properties. Finally, we show the potential profitability benefits of selecting robust solutions and that our estimation techniques can do this with greater accuracy than natural alternative strategies.

5.2 Motivation

Finding efficient routes for delivery vehicles is a classical Operations Research problem and many practitioners are hired by delivery firms seeking greater operational efficiency. Naturally, these practitioners find many real world complexities not considered in the standard theoretical formulation. One such issue surrounds continuity of service at regular customers, i.e., if a customer receives recurring deliveries, these should arrive at approximately the same time on the same day. In some cases, delivery time windows may even be specified as part of a contract. This consistency helps establish regular routines and lets drivers form a working relationship with clients. Unfortunately, this directly conflicts with another common real world complication: a dynamic customer set. As new customers arrive (and old ones leave), the disruption to existing routes should be kept minimal, even if operational savings can be made through significant reworking of the existing

routes.

In order to accept a new customer, no re-ordering of existing customers or shifting of current time windows is permitted; the customer can only be accepted as a deviation within the existing route. Naturally, vehicle capacity constraints and customer time window constraints must also be respected. In other words, new customers may only be added through a simple insertion operator. This scenario is actually quite common and was motivated through discussions with fleet managers at a large bakery, where delivery routes are only reviewed on an annual cycle. The problem we address is to create solutions where the ability to insert new customers is maximized. For convenience, this is referred to as the Vehicle Routing Problem with Future Insertions (VRPFI).

Obviously, there has been a large amount of study on designing efficient, low cost routes and any proposed solution method should fully utilize advances in this area. This could be achieved in combination with a meta–heuristic solver, where thousands of solutions may need evaluation. Consequently, any solution method must be computationally fast. Unfortunately, this is made difficult by the vastness of the solution space for this problem – we need to consider all possible schedules combined with all possible permutations of new customers. For this reason, we focus our study on heuristic methods. Additionally, the proposed method must allow for the easy comparison of different solutions. To achieve this, our method returns a simple numerical score, where solutions with a greater score should be able to accept more insertions.

There are two ways in which our heuristic could be used to guide a meta–heuristic solver. First, we note most meta–heuristics involve the insertion and removal of customers selected via some metric; this is often the resulting change in cost, but could be the change in our numerical score. Secondly, most meta–heuristics maintain a pool of solutions and must choose between them (either for further consideration, or to return to the user). Again, this is often done on objective value, but could be influenced by our heuristic.

To validate the performance of our heuristic, we need some way to accurately assess the true robustness of solutions. For this purpose, we developed a simulation environment in which a solution's ability to accept new customers is tested over a range of scenarios. We will present empirical evidence showing a correlation between the heuristic score and simulated performance. We will also present computational experiments showing that solutions selected by our heuristic outperform those chosen by current benchmark strategies.

5.3 The Vehicle Routing Problem with Future Insertions (VRPFI)

In this section we formally present the Vehicle Routing Problem with Future Insertions (VRPFI) and discuss some of the related issues.

5.3.1 Problem description

As discussed above, the VRPFI concerns a large delivery firm with a given set of vehicles and a partially known set of customers, who must create a set of delivery routes (one per

vehicle) which satisfy (known) customer demand. These must also satisfy time window constraints and vehicle capacity constraints. More formally, let a solution S represent a set of routes, with a cost denoted by $C(S)$. Given an ordered list of new requests N, define an operator $I(S, N)$ which produces a new solution by sequentially considering the requests in N, and inserting each into the position in a route in S which is feasible, and which increases the cost by the least. If a request in N cannot be feasibly inserted into any route, or does not offer a positive net return, it is rejected (requests can never be rejected voluntarily otherwise). We wish to find a solution S^* for which, over all possible realizations of N, we can *in expectation* maximize some objective; we discuss alternative objectives below. For further details on the exact network used and the generation of new customers, see Section 5.5.

5.3.2 Choice of objectives

Broadly speaking, there are two objectives the delivery firm may choose: maximizing the number of new customers accepted and maximizing their net profit.

5.3.2.1 Maximize number of new customers accepted

It is possible the delivery firm simply wants to maximize the number of new customers they are able to accept. This objective is simple, easy to understand and allows the firm to engage with the largest number of new customers (increasing potential growth). However, this ignores some key pricing issues.

5.3.2.2 Maximize net profit

Naturally, not all new customers are equally desirable; some require large detours for a small quantity of product, while others involve the reverse. Of course, such statements depend on the existing schedule and expectations around future customers, i.e., the insertion of a customer may only be worthwhile if we expect future customers in the same area. Although the former is known, statements about the second are very difficult to make in general. A compromise is to only allow insertions which immediately offer a positive return (although this prevents unprofitable insertions made profitable by subsequent customers or alternate routing decisions).

5.3.3 Trade off with initial schedules

Although it's reasonable to expect some correlation between a solution's initial cost (measured by distance traveled) and its ability to accept future insertions, there is still likely to be some trade–off. Low cost schedules involve efficient routing decisions which increases the slack available in the schedule for future insertions. However, this ignores other factors, such as the impact of time windows and the distribution of spare capacity between

vehicles. Hence, to accept more future insertions, it may be necessary to initially choose a sub–optimal solution and hope that the revenue obtained from additional new customers outweighs the expense incurred. However, this requires detailed knowledge about the expected customer set and complex statistical analysis. Additionally, many fleet managers would be hesitant to accept a significantly more expensive solution on the promise of a future pay–off. Therefore, a reasonable comprise may be to restrict the difference in initial cost compared to the best known solution by a certain amount, say 5%–10%. In a similar vein, it may be desirable to reject a customer that could feasibly be served (either when designing the initial solution or when later accepting new customers), if we expect a more profitable customer to arrive soon. However, very few industry clients are willing to accept this, so we do not consider this variant. Note we still only accept customers that increase net profit, but can't reject a customer for not being sufficiently profitable.

5.3.4 Estimating the customer set

In practice, the set of possible or likely new requests N is not known in advance, although there are various ways to estimate this. One such way is to model customer characteristics through statistical distributions calibrated from historical data, which can be used to generate a pool of new customers. This can then be used to produce a suite of test scenarios that can be used in a simulator to measure the performance of different solutions (Hvattum et al., 2006; Ichoua et al., 2006). The solution with the best performance across all scenarios can be selected. We would expect such an analysis to accurately identify the best solution, albeit after significant computational effort. This level of computation is impractical in a meta-heuristic setting, where thousands of solutions must be evaluated (Ropke and Pisinger, 2006a). An alternative is stochastic modeling, solved by approximate dynamic programming methods (Powell and Topaloglu, 2005; Godfrey and et al., 2002). However, these methods are limited to those instances where the required statistical analysis can be performed.

5.4 A Heuristic Approach

5.4.1 Overview

Due to the limitations of simulation and stochastic based techniques, we will now present a heuristic algorithm for the VRPFI. Our method involves calculating a numerical robustness "score" for each solution, based on the size of the region that future customers may feasibly appear in. Here, higher scores indicate a solution is likely to be more robust (that is, accept more new customers). Determining the score involves calculating the area of many ellipses, so we denote the score as the Ellipse Area Score, (EAS), and give more details below. It is important to stress that the EAS only represents a likelihood and a higher score cannot guarantee greater robustness. However, computational experiments presented later show it outperforms current benchmark methods.

5.4.2 Score calculation

To calculate the EAS for a given solution, first consider a pair of adjacent customers in a route, denoted as customers i and $i+1$. The probability that a new customer can be feasibly inserted is a function of how much *slack time* there is at node $i+1$ (slack time is how long the arrival at that node can be delayed without making it or any later visits infeasible). Specifically, consider an ellipse, where the foci are given by costumers i and $i+1$ and the maximum distance from these is the travel distance between the foci plus the distance that can be traveled in the slack time associated with customer $i+1$ (Figure 5.1). Any new customer inside this shape can be feasibly inserted (at least with respect to time constraints) – and a larger ellipse indicates greater potential for future insertion, making it a good estimator of robustness. The EAS for a route is the sum of areas of all ellipses for all adjacent locations, as shown in Figure 5.2. Similarly, the score for a solution is the sum of scores for all constituent routes. Note that there will be some overlap between adjacent ellipses in the same route (and the area will be counted twice), but for highly constrained routes with little slack time, this effect should be small. Conversely, overlap between ellipses from different routes likely cover distinct time periods, making it appropriate to count these areas twice.

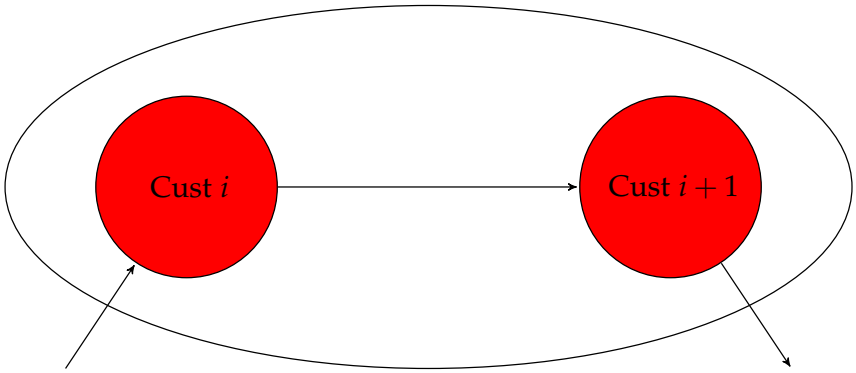


Figure 5.1: Area in which a new customer can be inserted for a given slack time

5.4.3 Extensions

5.4.3.1 Including Vehicle Capacity

To incorporate the effect of vehicular capacity, assume the quantity demanded at the new customer is drawn from the discrete set $q \in Q = \{q_{\min}, q_1, \ldots, q_{\max}\}, Q \subset \mathbb{K}^+$. We then generate a series of ellipses as described above, one for each $q \in Q$, provided the vehicle has enough spare capacity to serve a customer with demand q. When calculating the EAS , the area of an ellipse is weighted by the probability that quantity will be demanded. Vehicles with greater spare capacity will have a larger set of associated ellipses, giving a higher score. Note that all ellipses for a given customer pair (each representing a different quantity) are of the same size, although this assumption is removed in Section 5.4.3.2.

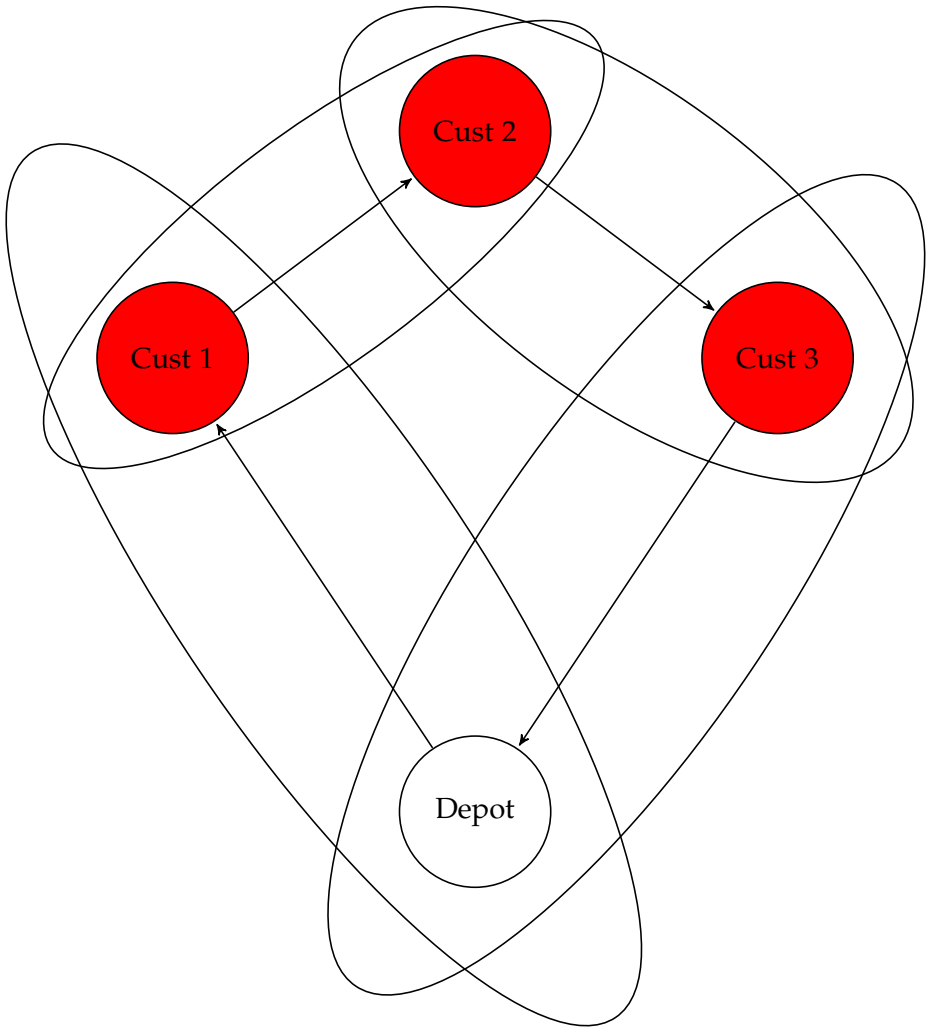


Figure 5.2: Score calculation for a complete route

5.4.3.2 Enforcing Profitable Insertions

We will now consider the requirement that a customer must increase the net profit. Specifically, we assume the profit is given by some reward (proportional to the quantity $q \in Q$ demanded) less a distance based travel cost. Consequently, each value of q implies a maximum permissible increase in distance, beyond which the insertion is not profitable. We repeat the analysis as given above, generating an ellipse for each $q \in Q$, except the maximum permitted distance from the foci of the ellipse is the minimum of that implied by the slack time, or by the quantity demanded. Of course, we only consider demand levels the vehicle has enough spare capacity to serve. An example is shown in Figure 5.3. The EAS of a solution is found by summing the area of all constituent ellipses across all routes.

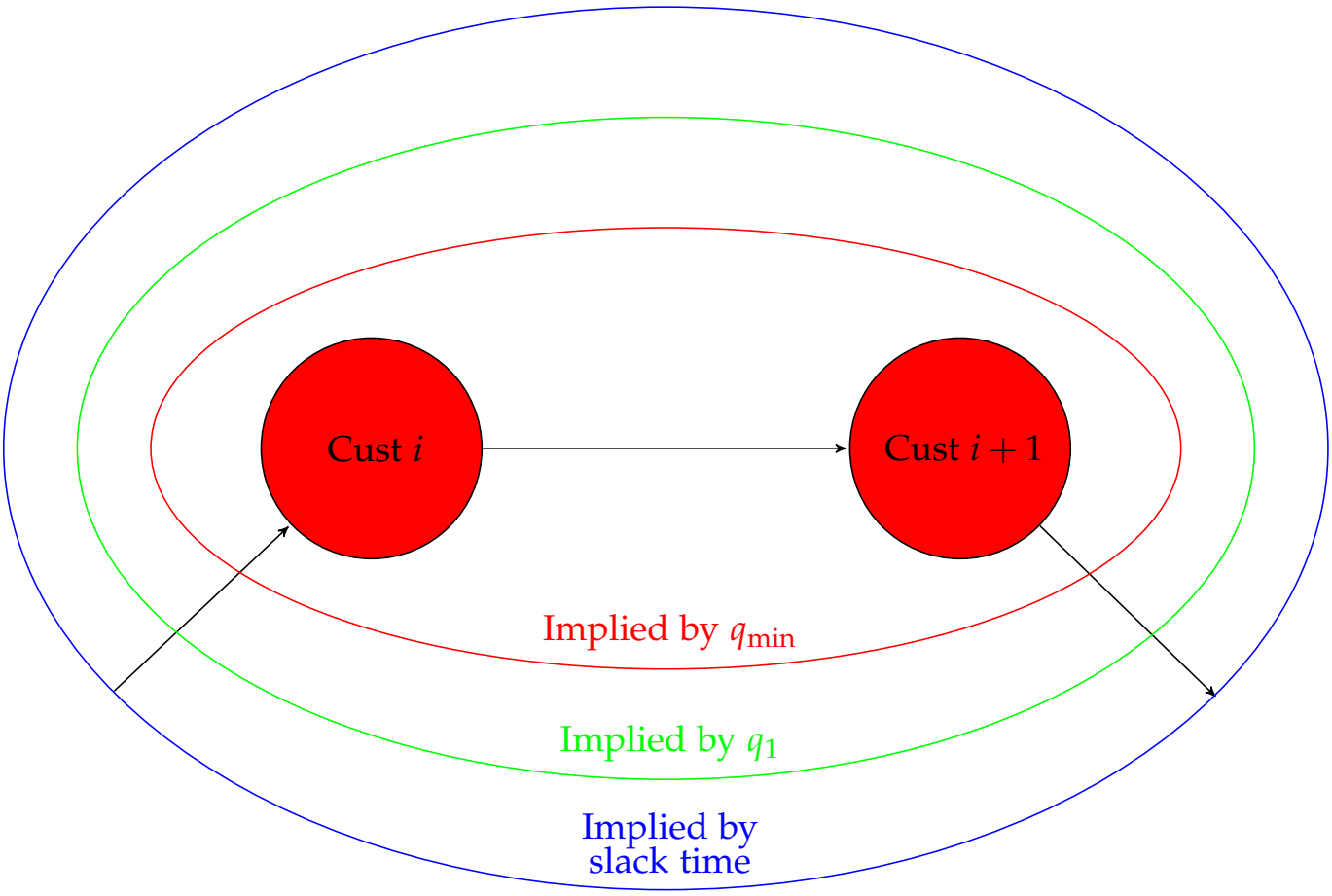


Figure 5.3: Areas in which insertions are profitable

5.4.4 Marginal gain in ellipse area

Broadly speaking, a solution's EAS depends on how the available slack time and spare capacity is distributed amongst the vehicles. It is interesting to look at the *change* in an ellipse's area caused by increasing (or decreasing) the permitted radius of an ellipse by

a single unit, which mirrors shifting slack time between vehicles. Figure 5.4 shows the *marginal* increase in the area of an ellipse by increasing the permitted radius by one unit; while the exact shape depends on the ellipse under consideration, the general pattern still holds. Although the eventual super linear increase is not surprising, the initial dip could be. This dip is related to the eccentricity of the ellipse and is more pronounced for ellipses with greater eccentricity. Note that for a given ellipse, if we decrease the permitted radius (by reducing the slack time at the second customer), then eccentricity will always rise. As near–capacity schedules have small amounts of slack time, we would expect ellipses in these schedules to be in the first half of the curve, where the best trade–off is less obvious.

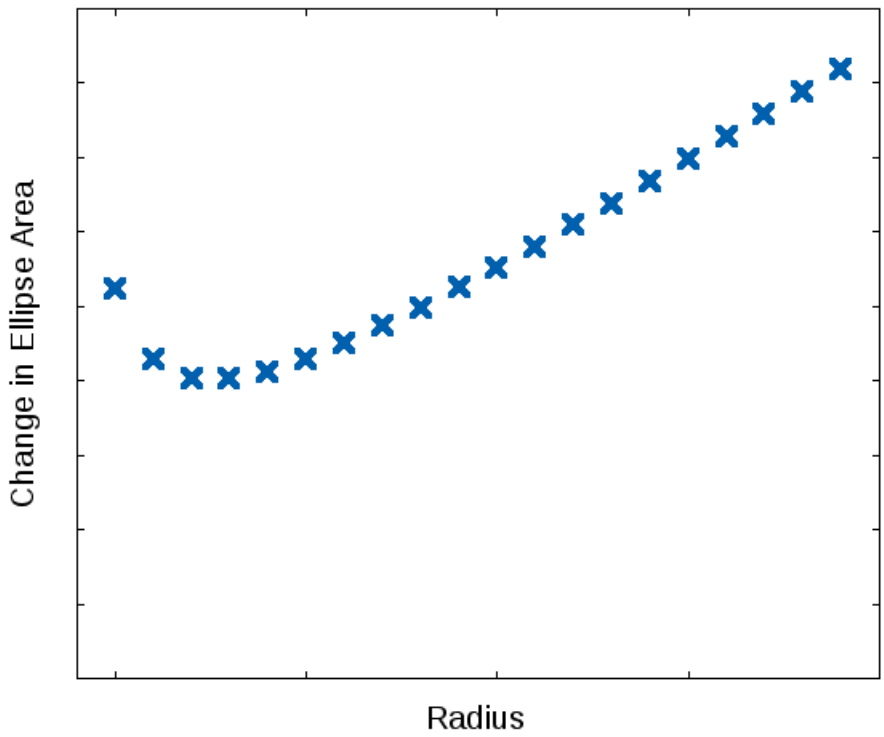


Figure 5.4: Change in area of ellipse per extra unit of slack time

5.5 Instance generation

5.5.1 Test Instances

To conduct our computational experiments demonstrating the benefits of robust solutions, we need to select a bank of problem instances. We chose not to create our own instances, partly so we knew what objective values represent a "high quality" solution, but also because we believe the trend for authors to use their own instances reduces comparability across studies and should only be done when necessary. We decided to use the ubiquitous Solomon instances (Solomon, 1987), which have been extensively studied elsewhere, meaning we can have confidence in the best–known solutions. Additionally, the variation in customer dispersal, time window length and planning horizon mimics features from industrial data sets.

Solomon instances involve 100 customers in a 100 unit square, with a depot positioned at $(35, 35)$. We assume Euclidean distances between locations and vehicles move at one unit of distance every time unit. Each instance is identified by an alphanumeric code indicating key characteristics. Customers may be dispersed (**R**andomly, in **C**lusters, or both **R**andomly and in **C**lusters). The planning horizon may be short (at 230 time units, where vehicles have a capacity of 200 and customers' time windows are 30 time units long), indicated by a numerical code of 1 or long (at 1000 time units, where vehicles have a capacity of 1000 and customers' time windows are 120 time units long), indicated by a numerical code of 2. Finally, each instance has a two digit identifier. As an example, instance R101 would refer to the first instance containing randomly dispersed customers with short time windows and instance RC207 would refer to the seventh instance containing both clustered and randomly dispersed customers with long time windows. We may also refer to a class of instances by replacing the numerical identifiers, i.e., R 1XX would refer to all instances containing randomly dispersed customers with short time windows.

Time windows were imposed on a percentage of customers drawn from the set $\{25\%, 50\%, 75\%, 100\%\}$. The specific customers selected and the length of the time windows are chosen randomly and independently. For the R XXX and RC XXX classes, the temporal center of these windows was chosen randomly; for the C XXX instances, a 3–opt algorithm was run and windows were centered on the service time. The latter approach was chosen to facilitate investigation in "cluster–first, route–second" approaches and represented a reasonable decision at the time. However, the time window placements mean there is unlikely to be multiple, high quality solutions, preventing a comparative analysis. For this reason, we excluded the C XXX class from further study.

5.5.2 Generation of new customers

The generation of new customers is performed independently for each individual instance and is done by fitting statistical distributions to the traits of existing customers. The x and y coordinates, earliest service time and quantity demanded by new customers are uniformly distributed between the minimum and maximum values seen in the instance. All new customers have time windows, and these mirror those already in the instance, e.g., are either 30 or 120 time units long.

5.5.3 Solution Generation

For each instance, we generated 600 solutions using the modern VRP solver *Indigo* (Kilby and Verden, 2011), which uses a combination of Large Neighborhood Search and Constraint Programming (a full discussion is given below). From this pool, we selected the 20 unique solutions with the best objective values and used these for our computational experiments (again, the objective was to minimize total distance traveled).

A step–by–step overview of Indigo, taken from (Kilby and Verden, 2011), is given below:

Overview of Indigo:

1. Create initial solution S
2. Choose a "destroy" method d
3. Create S' by removing customers from S according to method d
4. Choose an "insert" method i
5. Create solution S'' from S' by inserting customers according to method i
6. If the acceptance method accepts solution S'', then replace S with S''
7. If iterations remain, return to line 2

The use of Indigo is characterized by several decisions made by the user, i.e., the initial construction heuristic at line 1, the "destroy" and "insert" methods at lines 2 and 4, the acceptance methods at line 6 and the total number of iterations available at line 8. We will now discuss each of these in turn.

We use the four "destroy" methods outlined in (Pisinger and Ropke, 2007); these involve removing random customers, the most expensive customers, related customers and whole clusters of customers. The removal method is chosen adaptively (with probabilities based on past performance) and 20 customers are removed. Solutions are then rebuilt through insertion heuristics that find the customer/position pair with the most favorable score under one of three metrics. The first metric is simply the resulting increase in objective value (referred to as *minins*). This is extended by the *2–regret* metric which calculates the value under the *minins* metric for all customer/position pairs and selects the customer with the greatest difference between their two best positions (in different routes), as discussed in Section 2.3.3.1. The customer is of course inserted in the position giving the smallest increase in objective value. Finally, we define a *robustness* metric, where for each customer/-position pair, we calculate the percentage reduction in the EAS associated with the insertion (and select the pair giving the smallest reduction). Each metric was used exclusively to generate 200 solutions. If (following the completion of an iteration) a solution has an objective value within 2% of the best solution discovered by the heuristic so far, we invest further effort in an attempt to improve it. Specifically, we run the *Or–opt* improvement routine from Section 2.3.3.2, with a maximum chain length of 10 customers. We use a simulated acceptance method where the temperature gradient is 0.99975 and the initial probability is chosen such that there is a 50% chance of accepting an increase of 5%. The search was terminated after 20,000 iterations.

5.6 Computational Study

5.6.1 Indigo Validation

As discussed above, an advantage of using well studied instances is that we can benchmark our solutions against the current best known solutions. Indeed, such analysis is necessary as the robustness characteristics of low quality solutions are likely quite different (and are naturally of less interest to fleet managers and real world practitioners). We present the best solution found by Indigo as a fraction of the best known solution, averaged across each class, in Table 5.1. Of course, the requirement to generate 20 solutions

will lower the overall quality, so we also present the average ratio for the 20th best solution in parentheses.

Table 5.1: Differences compared to the best known objective value

Instance class	Fraction of best known objective value		
	Min	Avg	Max
RC 1XX	0.91 (0.94)	0.95 (0.99)	1.02 (1.04)
RC 2XX	0.99 (1.01)	1.00 (1.03)	1.01 (1.05)
R 1XX	0.94 (0.95)	0.95 (0.98)	0.97 (1.01)
R 2XX	0.99 (1.01)	1.01 (1.03)	1.05 (1.07)

Table 5.1 shows that all solutions found were of high quality, with a maximum deviation of 7% for the 20th best solution. Importantly, solutions for each instance had a similar cost, meaning any difference in robustness properties can't be due to solution quality. Note that in the cases where we beat the best known solution, this was a consequence of using more vehicles than were strictly required. We accept this may introduce doubt regarding the validation process, so we repeated the analysis using the minimum number of vehicles and present the results in Table 5.2.

Table 5.2: Difference from best known objective values with minimum vehicles

Instance class	Fraction of best known objective value		
	Min	Avg	Max
RC 1XX	1.01 (1.03)	1.02 (1.05)	1.03 (1.07)
RC 2XX	1.00 (1.02)	1.02 (1.04)	1.03 (1.06)
R 1XX	1.00 (1.02)	1.01 (1.03)	1.04 (1.05)
R 2XX	1.02 (1.04)	1.04 (1.06)	1.05 (1.07)

Table 5.2 shows that Indigo can reliably produce solutions comparable to the best known and that any observed robustness traits do not result from poor quality solutions.

5.6.2 Solution Analysis

Once we had our pool of solutions, we then attempted the insertion of 2000 independent sets of 20 new customers. Successfully inserted customers were left in the solution while their set was under consideration, but were removed afterwards. For each of the 20 solutions used, we calculate the proportion of new customers that could be inserted, denoted as the *simulation score*, where a score of 1 indicates all new customers were inserted.

This score allows us to motivate the discussion by showing the potential benefit of selecting the "right" initial solutions. For each *instance*, we can calculate the ratio of the highest to lowest simulation score, presented across the instance classes in Table 5.3. Over all instances, this ratio had an average value of 1.23, demonstrating that even amongst good, low cost solutions, some are significantly more able to accept new customers.

Our claim is two–fold – that the solution with the highest EAS will accept more insertions, and that this solution represents a better choice than other natural candidates. To investi-

Table 5.3: Ratio of best to worst simulation score across the instance classes

Instance class	Simulation score ratio		
	Min	Avg	Max
RC 1XX	1.03	1.16	1.40
RC 2XX	1.01	1.18	1.51
R 1XX	1.02	1.12	1.45
R 2XX	1.00	1.44	5.80

gate the first part, we utilize the Pearson Correlation Coefficient Pearson (1901), colloquially known as the *r–value*, to measure the linear dependence between the EAS and simulation score. This is done for each instance (where each data point represents one of our 20 generated solutions), as shown in Figure 5.5. Aggregated information is presented in Table 5.4.

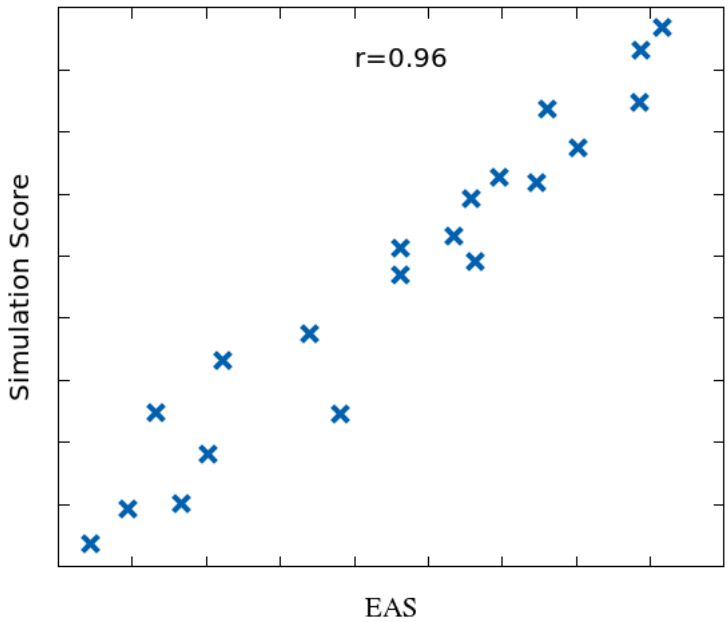


Figure 5.5: Correlation between EAS and simulation score for a single instance

Table 5.4: Correlation coefficient between EAS and simulation score

Instance class	Correlation coefficient		
	Min	Avg	Max
RC 1XX	0.38	0.73	0.96
RC 2XX	0.23	0.62	0.85
R 1XX	0.30	0.74	0.98
R 2XX	0.41	0.71	0.96

As we can see, there is a strong correlation between a solution's EAS and the observed simulation score in the majority of cases, suggesting there does exist a strong predictive relationship. These results become even more impressive by noting the EAS is calculated instantaneously while finding the simulation score takes upwards of ten minutes per instance.

5.6.3 Comparison between different strategies

Given a pool of 20 alternative solutions with similar costs, a fleet manager could have several ways of selecting which one to implement. We propose five strategies they might use, including one based on our heuristic given above. For clarity, when we generated our initial pool of solutions, we used the standard objective for these instances, i.e. that of minimum cost.

- **Strategy one: Minimum cost solution** This represents the "default" approach and is used in many real world operations. Fleet managers may be assessed (or receive performance bonuses) based on this measure, encouraging their buy–in. Additionally, it allows us to measure if additional robustness causes an increase in cost.
- **Strategy two: Private secondary objective** If presented with multiple solutions that have similar costs, some clients will select one based on their own distinct preferences. Examples includes clients who prefer routes that are visually attractive (Gretton and Kilby, 2013), that spread workload evenly amongst drivers (Banos et al., 2013), or that finish early on Fridays. As this differs between clients, selecting a solution at random is a way to approximate this.
- **Strategy three: Maximum average slack time** An alternative measure for robustness is the average slack time across all customers; greater values should allow more customers to be inserted. This strategy is intuitive and easy to justify to an industrial client. Additionally, as slack time is stored by most VRP solvers, calculating this value is computationally fast and inexpensive.
- **Strategy four: Highest simulation score** Also known as the "Oracle strategy', this relies on extensive computational power, making it impractical for use inside a metaheuristic solver. However, we can undertake this within our experimental setting, allowing us to compare our heuristic strategy against the best performance possible.
- **Strategy five: Maximum EAS** The last strategy involves selecting the solution with the greatest score from our heuristic.

Table 5.5 gives the percentage difference in insertions made compared to the solution with the greatest EAS , summarized across each class of instance. As an example, in the RC 1XX class, the solution with the greatest EAS could accept on average 6.46% more new customers than the one with the minimum cost. There are several interesting conclusions to draw from this table. First, we see the solution with the greatest EAS generally does quite well, consistently accepting more insertions than the other strategies. This indicates that the robustness measure is indeed a good predictor of the ability to accept future insertions. There was a narrower performance gap with the maximum slack strategy; this is not surprising, as its the only other strategy to directly consider slack time. Secondly, we note the minimum cost solution and the random solution have similar performance, further showing the "natural" strategy may not actually perform that well. Table 5.5 also shows that the "Min" column is always smaller in magnitude than the maximum one. In other words, using the schedule with the greatest EAS sometimes leads to small losses – but these are far outweighed by the potential for large benefits.

Table 5.5: Comparing simulation score of other strategies against greatest EAS

Instance Class	Percentage difference in simulation score					
	vs Minimum Cost			vs Random Solution		
	Min	Avg	Max	Min	Avg	Max
RC 1XX	-2.59%	6.46%	28.63%	-2.18%	6.42%	27.37%
RC 2XX	-0.36%	6.35%	36.10%	0.00%	10.20%	31.27%
R 1XX	-4.28%	4.63%	24.90%	-1.10%	4.12%	23.59%
R 2XX	-0.18%	7.02%	26.53%	-0.14%	8.95%	32.48%
Instance Class	Percentage difference in simulation score					
	vs Maximum slack			vs Oracle Solution		
	Min	Avg	Max	Min	Avg	Max
RC 1XX	-1.32%	1.21%	4.24%	-3.69%	-1.44%	0.00%
RC 2XX	-2.37%	2.02%	3.23%	-2.33%	-0.48%	0.00%
R 1XX	-1.98%	2.86%	4.98%	-7.41%	-1.52%	0.00%
R 2XX	-4.87%	3.23%	6.78%	-26.43%	-6.74%	0.00%

A possible concern arising from Table 5.5 may be that solutions with a greater EAS could be more expensive. Table 5.6 gives the percentage difference between the cost associated with each strategy compared to the solution with the highest EAS . Here, a positive percentage indicates the solution with the highest EAS was more expensive. As we can see, the solution with the highest EAS is only marginally more expensive than the minimum cost solution, and this is likely recouped by the extra customers served. It was pleasing to see the solution with the highest EAS was marginally cheaper than the random solution, showing robust solutions are not necessarily expensive. The maximum slack solution had a very similar cost to that chosen by our robustness measure, meaning there is no compensation for the smaller number of accepted insertions.

Table 5.6: Comparing cost of alternative strategies against greatest EAS

Instance Class	Percentage difference in cost					
	vs Min Cost			vs Random Solution		
	Min	Avg	Max	Min	Avg	Max
RC 1XX	0.02%	0.59%	1.23%	-1.93%	-0.12%	0.72%
RC 2XX	0.21%	1.19%	2.73%	-0.75%	-0.03%	0.26%
R 1XX	0.08%	0.31%	1.08%	-0.22%	0.05%	0.45%
R 2XX	0.00%	0.67%	2.14%	-3.31%	-0.29%	0.43%
Instance Class	vs Maximum slack			vs Oracle Solution		
	Min	Avg	Max	Min	Avg	Max
RC 1XX	-0.33%	0.45%	1.44%	-0.13%	0.36%	1.23%
RC 2XX	-0.56%	0.34%	1.87%	-0.48%	0.52%	1.32%
R 1XX	-0.87%	-0.29%	0.86%	-0.01%	0.09%	0.58%
R 2XX	-1.24%	-0.32%	1.30%	-1.37%	0.12%	1.80%

Finally, we would like to highlight an interesting feature of solutions with the greatest slack time. By the very nature of the objective, they tend to contain large amounts of wait time, meaning routes take longer to complete (even before new customers are added). In real world settings, drivers often receive a time–dependent wage and require more breaks on longer shifts. For this reason, fleet managers prefer temporally compact solutions that

only grow when necessary. Table 5.7 gives the percentage difference between the wait time in the solution with the highest EAS and the solution with the greatest slack time. We see the maximum slack solution involves significantly more wait time, as this is the only robustness mechanism measured. Conversely, our scoring procedure considers the distribution of spare capacity, if an insertion would be profitable and incorporates the non–linear relationship between time and insertion area.

Table 5.7: Comparing solutions with maximum EAS and maximum slack time

Instance Class	Percentage difference in wait time
RC 1XX	27.88%
RC 2XX	27.50%
R 1XX	27.05%
R 2XX	20.32%

5.7 Conclusion

The efficient routing of delivery vehicles aids commercial firms seeking a competitive advantage. We have investigated a setting where a firm must make permanent routing decisions without full knowledge of the customer set and presented it as a formal optimization problem. To solve this, we developed a novel geometric heuristic which estimates a solution's capability to accept future insertions through a simple numerical score. Computational experiments have given empirical evidence of a strong correlation between our EAS and simulated performance. They also illustrate that our heuristic can select the most robust solution from a pool with greater accuracy than alternative (and existing) methods.

Chapter 6

Conclusion

6.1 Research Summary

The Vehicle Routing Problem (VRP) and its associated variants are incredibly important and have a large number of practical applications, including scheduling logistics vehicles, creating travel itineraries and rostering delivery drivers. In this thesis, we focused on three important problems from two broad fields; the operation of mass transit systems and the robust scheduling of delivery vehicles. Each chapter investigated a new setting, formulated an optimization problem, provided various solution methods and presented empirical evidence highlighting gains in efficiency and profit.

6.2 EDRC Conclusions and Future Work

In Chapter 3, we investigated a shuttle service where commuters are carried from their private residence to a transit hub, where they catch a traditional public transport service to their true destination. We proposed an extended setting, where the service provider chooses the specific hub passengers are taken to (provided all relevant timing constraints are satisfied). We implemented both an exact and a heuristic solution method, with mechanisms designed for the dual–dependency time windows specific to this problem. Through computational experiments, we demonstrated our variant can achieve significant cost savings over traditional forms without unduly comprising customer service levels, potentially allowing its application in a much wider range of settings. There are a number of avenues for future work we would like to consider, including:

- **Link between number of vehicles and service quality:** There is a trade–off between the number of vehicles used and the level of service provided to participants. Evaluating and analyzing this trade–off would benefit real world practitioners wishing to estimate the cost–effectiveness of employing an additional vehicle.
- **General transit networks:** Our heuristic has exploited the special structure arising from considering a single train line. In future work, we want to explore more complex transit networks involving multiple routes with different stopping patterns e.g., an express service with a limited number of stops. To handle such situations, the

heuristic has to be revisited and revised e.g., it will no longer be possible to assume a service vehicle always leaves a station empty, and the pseudo-station concept has to be expanded to allow passengers to transfer at stations onto a different route.

- **Heterogeneous vehicle fleets:** We have assumed that all our service vehicles are identical. However, in the real world providers typically have a mixed fleet (as vehicles may be redeployed for other uses when required). Key characteristics like capacity and operational costs may vary between vehicles, which should be considered during the solution process.

- **Dynamicism:** We have so far assumed neither customers or transit services deviate from the agreed timetable and that travel times are deterministic. In reality, this is not the case. Real world practitioners need policies to handle late passengers or missed connections. Of course, such policies should consider the amount of slack in the vehicle's immediate schedule and the ability of other vehicles to assist. A simulation study would be needed to estimate the effect of such policies on passenger's behavior and the overall performance of the system. Naturally, the surrounding policy decisions will affect the perceived quality of service, which in turn affects the likelihood of attracting and retaining customers.

- **Unknown customer set:** While requiring passengers to pre–book certainly makes scheduling easier, this is not always possible for some passengers. We would like to investigate a variant with a mixed customer set, where some are known in advance, but others appear dynamically as the day progresses. It may be necessary to offer a reduced level of service to the latter group, to encourage pre–booking and to ensure earlier customers are not affected.

6.2.1 DRSDD Future work

In Chapter 4, we explored Dynamic Ride Sharing systems and discussed how the uncertainty around getting a match can discourage potential users, preventing the system from achieving the participation levels required for long term viability. In an attempt to relieve these concerns, some providers allow riders to specify that they only want a ride if they can be guaranteed a return trip. Previously, the optimization problem associated with this variant was thought to require a general integer program, but we have shown that (under minimal additional assumptions) it can be reformulated as a polynomially solvable *transshipment* problem. We then investigated a setting where the operator could offer this certainty by employing a group of *dedicated drivers* to service riders who would otherwise be unmatched. We discussed the various objectives and goals the service provider might have when scheduling these drivers and the various funding sources and business models available. We formulated a flexible optimization problem a provider could use and presented three alternative solution methods tailored for this specific problem. We also presented powerful preprocessing techniques that reduce the solution domain and the required solve time (with a very small risk of removing the optimal solution). Finally, we presented experiments showing the characteristics of networks where dedicated drivers are the most beneficial. There are some areas which we believe warrant further investigation, such as:

- **Rewards for flexible ad hoc drivers:** We can imagine schemes where ad hoc drivers are (financially) compensated for extra flexibility, i.e., by serving riders outside of their specified time windows, for accepting matches that are not cost feasible, or for repeatedly announcing trips despite not being matched. As ad hoc drivers were going to drive that route anyway, they may require less financial compensation (than dedicated drivers), although there are greater restrictions on their use.

- **Prioritizing certain riders:** We have assumed that the service provider treats all riders equally, but this does not need to be the case. Some participants could receive tighter time windows, better matches or even be guaranteed service. This could arise from the operator prioritizing first time participants, perhaps to leave a positive impression and improve retention rates. Alternatively, they could offer a premium program requiring a paid subscription that is used to offset the cost of these incentive schemes.

- **Modeling retention rates:** One of the potential benefits of dedicated drivers is the increase in retention rates. We could perform simulations over several days where the likelihood of participants joining depends on the system's historical success rate in getting a match (representing feedback from friends and family) and their continued involvement relies on their personally observed success. Importantly, we could measure the change in participation levels caused by the presence of dedicated drivers.

- **Links to public transport:** It would be interesting to imagine a scheme where participants share a trip to a transit station and take public transport to their final destination (leaving the vehicle parked at the station). Obviously, this is similar to the Demand Responsive Connector. As destinations and latest arrival times (representing stations and service departure times) are chosen from a common pool, it should be easy to find matches even at lower participation rates. If we considered the extended setting (where participants choose a final destination and the operator selects which local station they depart from), then even greater savings could be made.

- **Multiple riders:** We may wish to allow drivers to serve multiple riders (provided time windows are still respected), especially in settings where travel is concentrated along corridors (as minimal detouring would be required). It would be especially helpful if the presence of dedicated drivers lead to a surplus of riders in the system. However, there are some associated problems. First, it will (likely) increase the amount of time riders are in the car for, degrading the perceived service quality (even though time windows are still respected). Secondly, we would need to extend fare calculations to divide costs amongst multiple participants. Finally, it is unclear what to do when a single ad hoc driver can optimally (from a cost perspective) serve two riders, even though a second driver could serve one of the riders (encouraging wider participation and letting more participants share the savings).

- **Transfers between drivers:** It is easy to imagine situations where allowing riders to transfer between vehicles improves the objective value. However, it would require decisions around where riders may transfer and if any compensation should be offered for this deterioration in service quality.

6.2.2 Robustness to Insertions Future Work

In Chapter 5, we investigated a setting where a delivery firm has a dynamic customer set and must make permanent routing decisions before all customers are known. We proposed a novel, geometric heuristic to evaluate a solution's ability to accept future customers and this procedure is both computationally inexpensive and applicable in a wide range of settings. Through computational experiments, we illustrated the predictive accuracy of our method and showed it outperformed current benchmark strategies. Importantly, our heuristic does not increase the operating cost, showing that schedule robustness need not be expensive. Again, there are some topics and extensions which require further consideration:

- **Link between customer time windows and robustness score:** We have so far ignored the link between the location of new customers, and the likelihood of being able to find an insertion that satisfies their time windows. For customers near the edge of an ellipse, there is only a narrow interval in which they can feasibly be served (the reverse argument can be made for a customer near the center of an ellipse). This means area near the center of the ellipse is more "valuable" than that near the edge (although this is somewhat mitigated by the requirement that insertions increase net profit, which naturally favors area near the center). A naive solution is to repeat our method for each possible time window, but this would be incredibly expensive computationally. An alternative is a method that involves dividing the ellipse into bands, with each band being associated with a time interval; for feasibility, this interval must overlap with the new customer's time window. We believe there is some analytical pattern here, which could be exploited for computational efficiency.
- **Cluster of customers:** It is unclear how we should handle groups of customers clustered in the same physical region. There is likely to be significant overlap between the ellipses for subsequent pairs of customers, so we will count the same geographical area multiple times. A geometric solution is to try and find a combination of known shapes (e.g., circles and ellipses) which approximates the region covered; this can be viewed as a combination of the Polygon Covering Problem (Rourke et al., 1983; Johnson, 1982) and the Disk Covering Problem (Kershner, 1939). Alternatively, we could combine (some of) the customers and aggregate their characteristics in some manner, although the best way to do this is non–obvious.
- **General distributions of quantity demanded/service time/time window length:** We may be interested in the case where the quantity demanded by new customers follows some arbitrary distribution (perhaps even an empirical one). This requires no modification of our method (assuming discrete quantities) – we simply require a list of all possible demand quantities and their associated probability. This may be able to be extended to continuous quantities by carefully integrating over probability functions. Similar statements can be made about customer's service time or the length of their time windows.
- **General distribution of customer location:** We are likely interested in cases where the location of customers is not uniformly distributed. An obvious strategy is to assign a score representing the likelihood of customers appearing to different geographical regions and weight the EAS by the score of areas covered. The simplest

method would be to use a grid, but we could use the union of any set of shapes. However, this is likely to be computationally expensive. A simpler approach is to define a series of High Demand Points (HDP), which represent locations where we expect future customers to appear (perhaps representing shopping malls or large complexes). If these are contained within an ellipse, we simply apply an additional bonus score, the value of which should be determined empirically, but can be interpreted as the equivalent area you would need to serve to achieve the same number of new customers.

- **Independence of customer characteristics:** We have so far assumed that the characteristics of new customers are independent i.e., the length of a customer's time window is independent from their demand. This is unlikely to be true in practice, but relaxing this assumption would be very challenging. Instead, we could create a pool of representative customer profiles, where each profile has independent distributions for its characteristics, e.g. One profile may concern supermarkets who demand large amounts of goods with narrow time windows, and another concerns corner store owners with low demand dispersed throughout the region. We would simply repeat the analysis for each profile and weight the EAS by the probability of the associated customer type appearing. Again, efficiency gains can be achieved with an implementation that reuses calculated information for different customer types.

- **Alternative robustness definition:** So far, we have used robustness to refer to the level of new customers that could be successfully inserted. It would also be reasonable to consider "*operational robustness*", which is a temporal buffer to allow for variability on the day of operations, e.g., traffic delays, unprepared customers and vehicle breakdowns. In this case, the optimisation problem involves accepting as many new customers are possible, given this required buffer.

Taken together, we believe the findings given in this thesis represent an advancement in knowledge across multiple transport sectors and we thank our readers for their time and attention.

Bibliography

Agatz, N.; Erera, A.; Savelsbergh, M.; and Wang, X., 2010. Sustainable passenger transportation: Dynamic ride-sharing. Technical report, Georgia Institute of Technology. (cited on pages 42, 78, and 84)

Agatz, N.; Erera, A.; Savelsbergh, M.; and Wang, X., 2012a. Optimization for dynamic ride-sharing: A review. *European Journal of Operational Research*, 223, 2 (2012), 295 – 303. doi:http://dx.doi.org/10.1016/j.ejor.2012.05.028. http://www.sciencedirect.com/science/article/pii/S0377221712003864. (cited on pages 40 and 74)

Agatz, N.; Erera, A. L.; Savelsbergh, M.; and Wang, X., 2011. Dynamic ride-sharing: a simulation study in metro atlanta. *Procedia - Social and Behavioral Sciences*, 17, 0 (2011), 532 – 550. (cited on pages 43, 44, 75, 76, 78, 84, and 85)

Agatz, N.; Wang, X.; and Erera, A., 2012b. Finding optimal stable ride matches for dynamic ride–sharing systems. Technical report, Erasmus University, Rotterdam, The Netherlands. (cited on page 43)

Aggarwal, C. C.; Orlin, J. B.; and Tai, R. P., 1997. Optimized crossover for the independent set problem. *Operations Research*, 45, 2 (1997), 226–234. (cited on pages 32 and 33)

Aldaihani, M. and Dessouky, M. M., 2003. Hybrid scheduling methods for paratransit operations. *Computers and Industrial Engineering*, 45, 1 (2003), 75 – 96. (cited on page 35)

Alfa, A. S.; Heragu, S. S.; and Chen, M., 1991. A 3-opt based simulated annealing algorithm for vehicle routing problems. *Computers & Industrial Engineering*, 21, 1 (1991), 635 – 639. doi:http://dx.doi.org/10.1016/0360-8352(91)90165-3. http://www.sciencedirect.com/science/article/pii/0360835291901653. (cited on page 29)

Alt, F.; Shirazi, A. S.; Schmidt, A.; Kramer, U.; and Nawaz, Z., 2010. Location-based crowdsourcing: extending crowdsourcing to the real world. In *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries*, 13–22. ACM. (cited on page 43)

Alt, H.; Blum, N.; Mehlhorn, K.; and Paul, M., 1991. Computing a maximum cardinality matching in a bipartite graph in time $\mathcal{O}\left(n^{1.5}\sqrt{\frac{m}{\log n}}\right)$. *Information Processing Letters*, 37, 4 (1991), 237–240. (cited on page 43)

Anita, T. and Rucha, D., 2012. Article: Genetic algorithm-survey paper. *IJCA Proceedings on NCRTC*, 5 (2012), 25–29. (cited on page 32)

Archetti, C.; Feillet, D.; Hertz, A.; and Speranza, M. G., 2009. The capacitated team orienteering and profitable tour problems. *The Journal of the Operational Research Society*, 60, 6 (2009), 831–842. (cited on page 9)

Archetti, C.; Speranza, M.; and Vigo, D., 2013. Vehicle routing problems with profits. Technical report, Tech. Report WPDEM2013/3, University of Brescia. (cited on page 9)

Armas, J. D. and Melian-Batista, B., 2015. Variable neighborhood search for a dynamic rich vehicle routing problem with time windows. *Computers & Industrial Engineering*, 85 (2015), 120 – 131. doi:http://dx.doi.org/10.1016/j.cie.2015.03.006. http://www.sciencedirect.com/science/article/pii/S0360835215001151. (cited on page 31)

Arslan, A. M.; Agatz, N.; Kroon, L.; and Zuidwijk, R., 2015. Crowdsourced same day delivery. In *Sixth International Workshop on Freight Transportation and Logistics*, 356–359. (cited on page 43)

Attanasio, A.; Cordeau, J.-F.; Ghiani, G.; and Laporte, G., 2004. Parallel tabu search heuristics for the dynamic multi-vehicle dial-a-ride problem. *Parallel Computing*, 30, 3 (2004), 377–387. (cited on page 48)

Azencott, R., 1992. Parallel simulated annealing: An overview of basic techniques. *Simulated annealing: parallelization techniques*, (1992), 37–46. (cited on page 29)

Azi, N.; Gendreau, M.; and Potvin, J.-Y., 2010. *An adaptive large neighborhood search for a vehicle routing problem with multiple trips*. CIRRELT. (cited on page 32)

Azi, N.; Gendreau, M.; and Potvin, J.-Y., 2012. A dynamic vehicle routing problem with multiple delivery routes. *Annals of Operations Research*, 199, 1 (2012), 103–112. (cited on page 32)

Badeau, P.; Guertin, F.; Gendreau, M.; Potvin, J.-Y.; and Taillard, E., 1997. A parallel tabu search heuristic for the vehicle routing problem with time windows. *Transportation Research Part C: Emerging Technologies*, 5, 2 (1997), 109 – 122. doi:http://dx.doi.org/10.1016/S0968-090X(97)00005-3. http://www.sciencedirect.com/science/article/pii/S0968090X97000053. (cited on page 30)

Baker, B. M. and Ayechew, M., 2003. A genetic algorithm for the vehicle routing problem. *Computers & Operations Research*, 30, 5 (2003), 787 – 800. doi:http://dx.doi.org/10.1016/S0305-0548(02)00051-5. http://www.sciencedirect.com/science/article/pii/S0305054802000515. (cited on page 32)

Balas, E., 1987. The prize collecting traveling salesman problem. Technical report, DTIC Document. (cited on page 9)

Banos, R.; Ortega, J.; Gil, C.; Fernandez, A.; and de Toro, F., 2013. A simulated annealing-based parallel multi-objective approach to vehicle routing problems with time windows. *Expert Systems with Applications*, 40, 5 (2013), 1696 – 1707. doi:http://dx.doi.org/10.1016/j.eswa.2012.09.012. http://www.sciencedirect.com/science/article/pii/S0957417412010664. (cited on pages 29, 30, and 129)

Barbosa, E. B. d. M.; Senne, E. L. F.; and Silva, M. B., 2015. Improving the performance of metaheuristics: An approach combining response surface methodology and racing algorithms. *International Journal of Engineering Mathematics*, 2015 (2015). (cited on page 28)

Belhaiza, S.; Hansen, P.; and Laporte, G., 2014. A hybrid variable neighborhood tabu search heuristic for the vehicle routing problem with multiple time windows. *Computers*

and Operations Research, 52, Part B (2014), 269 – 281. doi:http://dx.doi.org/10.1016/j.cor.2013.08.010. http://www.sciencedirect.com/science/article/pii/S0305054813002165. (cited on pages 31 and 34)

Bent, R. and Van Hentenryck, P., 2004a. A two-stage hybrid local search for the vehicle routing problem with time windows. *Transportation Science*, 38, 4 (2004), 515–530. (cited on pages 32 and 34)

Bent, R. W. and Van Hentenryck, P., 2004b. Scenario-based planning for partially dynamic vehicle routing with stochastic customers. *Operations Research*, 52, 6 (2004), 977–987. (cited on page 48)

Berger, J. and Barkaoui, M., 2004. A parallel hybrid genetic algorithm for the vehicle routing problem with time windows. *Computers & Operations Research*, 31, 12 (2004), 2037 – 2053. doi:http://dx.doi.org/10.1016/S0305-0548(03)00163-1. http://www.sciencedirect.com/science/article/pii/S0305054803001631. (cited on page 33)

Bianchi, L.; Dorigo, M.; Gambardella, L. M.; and Gutjahr, W. J., 2009. A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing: an international journal*, 8, 2 (2009), 239–287. (cited on page 20)

Blanton, J. L. and Wainwright, R. L., 1993. Multiple vehicle routing with time and capacity constraints using genetic algorithms. In *Proceedings of the 5th International Conference on Genetic Algorithms*, 452–459. Morgan Kaufmann Publishers Inc. (cited on page 32)

Boussaid, I.; Lepagnot, J.; and Siarry, P., 2013. A survey on optimization metaheuristics. *Information Sciences*, 237 (2013), 82 – 117. doi:http://dx.doi.org/10.1016/j.ins.2013.02.041. http://www.sciencedirect.com/science/article/pii/S0020025513001588. (cited on page 28)

Boussier, S.; Feillet, D.; and Gendreau, M., 2007. An exact algorithm for team orienteering problems. *4or*, 5, 3 (2007), 211–230. (cited on page 93)

Brandao, J., 2011. A tabu search algorithm for the heterogeneous fixed fleet vehicle routing problem. *Computers and Operations Research*, 38, 1 (2011), 140 – 151. doi:http://dx.doi.org/10.1016/j.cor.2010.04.008. http://www.sciencedirect.com/science/article/pii/S0305054810000924. (cited on page 30)

Bräysy, O., 2003. A reactive variable neighborhood search for the vehicle-routing problem with time windows. *INFORMS Journal on Computing*, 15, 4 (2003), 347–368. (cited on page 31)

Breedam, A. V., 1995. Improvement heuristics for the vehicle routing problem based on simulated annealing. *European Journal of Operational Research*, 86, 3 (1995), 480 – 490. doi:http://dx.doi.org/10.1016/0377-2217(94)00064-J. http://www.sciencedirect.com/science/article/pii/037722179400064J. (cited on page 29)

Bruun, E. and Marx, E., 2006. Omnilink: Case study of successful flex route-capable intelligent transportation system implementation. *Transportation Research Record: Journal of the Transportation Research Board*, (2006), 91–98. (cited on page 1)

Cattaruzza, D.; Absi, N.; Feillet, D.; and Vidal, T., 2014. A memetic algorithm for the multi trip vehicle routing problem. *European Journal of Operational Research*, 236, 3 (2014),

833 – 848. doi:http://dx.doi.org/10.1016/j.ejor.2013.06.012. http://www.sciencedirect.com/science/article/pii/S0377221713005006. (cited on page 33)

Cayford, R. and Yim, Y., 2004. Personalized demand-responsive transit service. Technical report, University of California. (cited on page 37)

Chakraborty, B., 2004. Ga-based multiple route selection for car navigation. In *Applied Computing* (Eds. S. Manandhar; J. Austin; U. Desai; Y. Oyanagi; and A. Talukder), vol. 3285 of *Lecture Notes in Computer Science*, 76–83. Springer Berlin Heidelberg. ISBN 978-3-540-23659-7. doi:10.1007/978-3-540-30176-9_10. http://dx.doi.org/10.1007/978-3-540-30176-9_10. (cited on page 32)

Chandra, S. and Quadrifoglio, L., 2013a. A model for estimating the optimal cycle length of demand responsive feeder transit services. *Transportation Research Part B: Methodological*, 51, 0 (2013), 1 – 16. doi:10.1016/j.trb.2013.01.008. http://www.sciencedirect.com/science/article/pii/S0191261513000192. (cited on page 37)

Chandra, S. and Quadrifoglio, L., 2013b. A new street connectivity indicator to predict performance for feeder transit services. *Transportation Research Part C: Emerging Technologies*, 30, 0 (2013), 67 – 80. doi:http://dx.doi.org/10.1016/j.trc.2013.02.004. http://www.sciencedirect.com/science/article/pii/S0968090X1300034X. (cited on page 37)

Chang, M.-s.; Chen, S.; and Hsueh, C., 2003. Real-time vehicle routing problem with time windows and simultaneous delivery/pickup demands. *Journal of the Eastern Asia Society for Transportation Studies*, 5 (2003), 2273–2286. (cited on page 48)

Chen, Z.-L. and Xu, H., 2006. Dynamic column generation for dynamic vehicle routing with time windows. *Transportation Science*, 40, 1 (2006), 74–88. (cited on page 48)

Chiang, W.-C. and Russell, R., 1996. Simulated annealing metaheuristics for the vehicle routing problem with time windows. *Annals of Operations Research*, 63, 1 (1996), 3–27. doi:10.1007/BF02601637. http://dx.doi.org/10.1007/BF02601637. (cited on page 29)

Clarke, G. and Wright, J. W., 1964. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12, 4 (1964), 568–581. doi:10.1287/opre.12.4.568. http://dx.doi.org/10.1287/opre.12.4.568. (cited on page 23)

Cook, S. A., 1971. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, 151–158. ACM. (cited on page 5)

Cordeau, J.-F., 2006. A branch-and-cut algorithm for the dial-a-ride problem. *Oper. Res.*, 54, 3 (may 2006), 573–586. doi:10.1287/opre.1060.0283. http://dx.doi.org/10.1287/opre.1060.0283. (cited on page 40)

Cordeau, J.-F. and Laporte, G., 2007. The dial-a-ride problem: models and algorithms. *Annals of Operations Research*, 153, 1 (2007), 29–46. (cited on page 11)

Cordeau, J.-F.; Laporte, G.; Mercier, A.; et al., 2001. A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational research society*, 52, 8 (2001), 928–936. (cited on page 30)

Cordeau, J.-F. and Maischberger, M., 2012. A parallel iterated tabu search heuristic for vehicle routing problems. *Computers and Operations Research*, 39, 9 (2012), 2033 – 2050. doi:http://dx.doi.org/10.1016/j.cor.2011.09.021. http://www.sciencedirect.com/science/article/pii/S0305054811002826. (cited on page 30)

Coy, S. P.; Golden, B. L.; Runger, G. C.; and Wasil, E. A., 2001. Using experimental design to find effective parameter settings for heuristics. *Journal of Heuristics*, 7, 1 (2001), 77–97. (cited on page 28)

Crainic, T. G.; Toulouse, M.; and Gendreau, M., 1993. An appraisal of asynchronous parallelization approaches for tabu search algorithms. Technical report, CENTRE DE RECHERCHE SUR LES TRANSPORTS. (cited on page 30)

Czech, Z. J. and Czarnas, P., 2002. Parallel simulated annealing for the vehicle routing problem with time windows. In *10th Euromicro Workshop on Parallel, Distributed and Network-based Processing, Canary Islands–Spain*, 376–383. (cited on page 29)

Daniels, R. and Mulley, C., 2010. Overcoming barriers to implementing flexible transport services in nsw. Technical report, Monash University. (cited on page 35)

De Backer, B.; Furnon, V.; Shaw, P.; Kilby, P.; and Prosser, P., 2000. Solving vehicle routing problems using constraint programming and metaheuristics. *Journal of Heuristics*, 6, 4 (2000), 501–523. (cited on page 33)

Deleplanque, S. and Quilliot, A., 2013. Dial–a–ride problem with time windows, transshipments, and dynamic transfer points. In *Manufacturing Modelling, Management, and Control*, vol. 7, 1256–1261. (cited on page 53)

Diana, M.; Quadrifoglio, L.; and Pronello, C., 2009. A methodology for comparing distances traveled by performance-equivalent fixed–route and demand responsive transit services. *Transportation Planning and Technology*, 32, 4 (2009), 377–399. (cited on page 35)

Divsalar, A.; Vansteenwegen, P.; and Cattrysse, D., 2013. A variable neighborhood search method for the orienteering problem with hotel selection. *International Journal of Production Economics*, 145, 1 (2013), 150 – 160. doi:http://dx.doi.org/10.1016/j.ijpe.2013.01.010. http://www.sciencedirect.com/science/article/pii/S0925527313000285. (cited on page 31)

Dowsland, K. A. and Thompson, J. M., 2012. Simulated annealing. In *Handbook of Natural Computing*, 1623–1655. Springer. (cited on page 29)

Drexl, M., 2012. Synchronization in vehicle routing - a survey of vrps with multiple synchronization constraints. *Transportation Science*, 46, 3 (2012), 297–316. doi:10.1287/trsc.1110.0400. http://dx.doi.org/10.1287/trsc.1110.0400. (cited on page 35)

Dror, M.; Laporte, G.; and Trudeau, P., 1989. Vehicle routing with stochastic demands: Properties and solution frameworks. *Transportation science*, 23, 3 (1989), 166–176. (cited on page 47)

Dunstall, S.; Horn, M. E.; Kilby, P.; Krishnamoorthy, M.; Owens, B.; Sier, D.; and Thiebaux, S., 2003. An automated itinerary planning system for holiday travel. *Information Technology & Tourism*, 6, 3 (2003), 195–210. (cited on page 8)

Ehrgott, M. and Wiecek, M. M., 2005. Mutiobjective programming. In *Multiple criteria decision analysis: State of the art surveys*, 667–708. Springer. (cited on page 10)

El-Sherbeny, N. A., 2010. Vehicle routing with time windows: An overview of exact, heuristic and metaheuristic methods. *Journal of King Saud University - Science*, 22, 3 (2010), 123 – 131. doi:http://dx.doi.org/10.1016/j.jksus.2010.03.002. http://www.sciencedirect.com/science/article/pii/S1018364710000297. (cited on pages 8 and 12)

Escobar, J. W.; Linfati, R.; Toth, P.; and Baldoquin, M., 2014. A hybrid granular tabu search algorithm for the multi-depot vehicle routing problem. *Journal of Heuristics*, 20, 5 (2014), 483–509. doi:10.1007/s10732-014-9247-0. (cited on page 30)

Feillet, D., 2010. A tutorial on column generation and branch-and-price for vehicle routing problems. *4OR*, 8, 4 (2010), 407–424. doi:10.1007/s10288-010-0130-z. http://dx.doi.org/10.1007/s10288-010-0130-z. (cited on pages 21 and 22)

Feillet, D.; Dejax, P.; and Gendreau, M., 2005. Traveling salesman problems with profits. *Transportation Science*, 39, 2 (2005), 188–205. doi:10.1287/trsc.1030.0079. http://pubsonline.informs.org/doi/abs/10.1287/trsc.1030.0079. (cited on page 9)

Ferreira, L.; Charles, P.; and Tether, C., 2007. Evaluating flexible transport solutions. *Transportation Planning and Technology*, 30, 2-3 (2007), 249–269. (cited on page 35)

Fleischmann, B., 1990. *The vehicle routing problem with multiple use of the vehicles*. University of Hamburg. (cited on page 8)

Fleszar, K.; Osman, I. H.; and Hindi, K. S., 2009. A variable neighbourhood search algorithm for the open vehicle routing problem. *European Journal of Operational Research*, 195, 3 (2009), 803 – 809. doi:http://dx.doi.org/10.1016/j.ejor.2007.06.064. http://www.sciencedirect.com/science/article/pii/S0377221707011046. (cited on page 31)

Foulds, L. R., 1983. The heuristic problem-solving approach. *The Journal of the Operational Research Society*, 34, 10 (1983), pp. 927–934. http://www.jstor.org/stable/2580891. (cited on page 23)

Fu, L. and Xu, Y., 2001. Potential effects of automatic vehicle location and computer-aided dispatch technology on paratransit performance: a simulation study. *Transportation Research Record: Journal of the Transportation Research Board*, 1760, 1 (2001), 107–113. (cited on page 35)

Furuhata, M.; Dessouky, M.; Ordóñez, F.; Brunet, M.-E.; Wang, X.; and Koenig, S., 2013. Ridesharing: The state-of-the-art and future directions. *Transportation Research Part B: Methodological*, 57 (2013), 28 – 46. doi:http://dx.doi.org/10.1016/j.trb.2013.08.012. http://www.sciencedirect.com/science/article/pii/S0191261513001483. (cited on pages 40 and 43)

Garcia, B.-L.; Potvin, J.-Y.; and Rousseau, J.-M., 1994. A parallel implementation of the tabu search heuristic for vehicle routing problems with time window constraints. *Computers & Operations Research*, 21, 9 (1994), 1025–1033. (cited on page 30)

Gendreau, M.; Guertin, F.; Potvin, J.-Y.; and Taillard, E., 1999. Parallel tabu search for real-time vehicle routing and dispatching. *Transportation science*, 33, 4 (1999), 381–390. (cited on pages 30 and 48)

Gendreau, M.; Hertz, A.; and Laporte, G., 1994. A tabu search heuristic for the vehicle routing problem. *Management science*, 40, 10 (1994), 1276–1290. (cited on page 30)

Gendreau, M.; Iori, M.; Laporte, G.; and Martello, S., 2008. A tabu search heuristic for the vehicle routing problem with two-dimensional loading constraints. *Networks*, 51, 1 (2008), 4–18. (cited on page 30)

Glover, F., 1989. Tabu search-part i. *ORSA Journal on computing*, 1, 3 (1989), 190–206. (cited on page 29)

Glover, F., 1990. Tabu search—part ii. *ORSA Journal on computing*, 2, 1 (1990), 4–32. (cited on page 29)

Glover, F. and Laguna, M., 2013. Tabu search*. In *Handbook of Combinatorial Optimization* (Eds. P. M. Pardalos; D.-Z. Du; and R. L. Graham), 3261–3362. Springer New York. ISBN 978-1-4419-7996-4. (cited on page 30)

Godfrey, G. A. and et al., 2002. An adaptive dynamic programming algorithm for dynamic fleet management, i: Single period travel times. (cited on page 120)

Godfrey, G. A. and Powell, W. B., 2002. An adaptive dynamic programming algorithm for dynamic fleet management, i: Single period travel times. *Transportation Science*, 36, 1 (2002), 21–39. doi:10.1287/trsc.36.1.21.570. (cited on page 48)

Golden, B.; Assad, A.; Levy, L.; and Gheysens, F., 1984. The fleet size and mix vehicle routing problem. *Computers & Operations Research*, 11, 1 (1984), 49–66. (cited on page 11)

Golden, B. L. and Skiscim, C. C., 1986. Using simulated annealing to solve routing and location problems. *Naval Research Logistics Quarterly*, 33, 2 (1986), 261–279. (cited on page 29)

Goncalves, J. F., 2007. A hybrid genetic algorithm-heuristic for a two-dimensional orthogonal packing problem. *European Journal of Operational Research*, 183, 3 (2007), 1212 – 1229. doi:http://dx.doi.org/10.1016/j.ejor.2005.11.062. http://www.sciencedirect.com/science/article/pii/S0377221706003006. (cited on page 32)

Graffigne, C., 1992. Parallel annealing by periodically interacting multiple searches: An experimental study. *Simulated Annealing: parallelization techniques*, 27 (1992), 46. (cited on page 29)

Gretton, C. and Kilby, P., 2013. A study of shape penalties in vehicle routing. Technical report, NICTA. (cited on page 129)

Groer, C.; Golden, B.; and Wasil, E., 2009. The consistent vehicle routing problem. *Manufacturing & Service Operations Management*, 11, 4 (2009), 630–643. (cited on page 11)

Guihaire, V. and Hao, J.-K., 2008. Transit network design and scheduling: A global review. *Transportation Research Part A: Policy and Practice*, 42, 10 (2008), 1251 – 1273. doi:http://dx.doi.org/10.1016/j.tra.2008.03.011. http://www.sciencedirect.com/science/article/pii/S0965856408000888. (cited on page 34)

Häll, C. H.; Andersson, H.; Lundgren, J. T.; and Värbrand, P., 2009. The integrated dial–a–ride problem. *Public Transport*, 1, 1 (2009), 39–54. (cited on page 53)

Hansen, P.; Mladenović, N.; and Pérez, J. A. M., 2010. Variable neighbourhood search: methods and applications. *Annals of Operations Research*, 175, 1 (2010), 367–407. (cited on page 31)

Harmanani, H.; Azar, D.; Helal, N.; and Keirouz, W., 2011. A simulated annealing algorithm for the capacitated vehicle routing problem. In *Proceedings of the ISCA 26th International Conference on Computers and Their Applications, Louisiana, USA*. (cited on page 29)

Hashimoto, H.; Yagiura, M.; Imahori, S.; and Ibaraki, T., 2010. Recent progress of local search in handling the time window constraints of the vehicle routing problem. *4OR*, 8, 3 (2010), 221–238. doi:10.1007/s10288-010-0144-6. http://dx.doi.org/10.1007/s10288-010-0144-6. (cited on page 8)

Hillier, F. S.; Lieberman, G. J.; Hillier, F.; and Lieberman, G., 2015. *Introduction to Operations Research*. {McGraw-Hill Science/Engineering/Math}. (cited on page 12)

Ho, W.; Ho, G. T.; Ji, P.; and Lau, H. C., 2008. A hybrid genetic algorithm for the multi-depot vehicle routing problem. *Engineering Applications of Artificial Intelligence*, 21, 4 (2008), 548 – 557. doi:http://dx.doi.org/10.1016/j.engappai.2007.06.001. http://www.sciencedirect.com/science/article/pii/S0952197607000887. (cited on page 33)

Holland, J. H., 1992. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press. (cited on page 32)

Hopcroft, J. E. and Karp, R. M., 1973. An $n^{\frac{5}{2}}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on computing*, 2, 4 (1973), 225–231. (cited on page 43)

Hosni, H.; Naoum-Sawaya, J.; and Artail, H., 2014. The shared-taxi problem: Formulation and solution methods. *Transportation Research Part B: Methodological*, 70 (2014), 303 – 318. doi:http://dx.doi.org/10.1016/j.trb.2014.09.011. http://www.sciencedirect.com/science/article/pii/S0191261514001659. (cited on page 43)

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 (Nov. 2006), 421–438. doi:10.1287/trsc.1060.0166. http://dx.doi.org/10.1287/trsc.1060.0166. (cited on page 120)

Ichoua, S.; Gendreau, M.; and Potvin, J.-Y., 2000. Diversion issues in real-time vehicle dispatching. *Transportation Science*, 34, 4 (2000), 426–438. (cited on page 48)

Ichoua, S.; Gendreau, M.; and Potvin, J.-Y., 2003. Vehicle dispatching with time–dependent travel times. *European journal of operational research*, 144, 2 (2003), 379–396. (cited on page 48)

Ichoua, S.; Gendreau, M.; and Potvin, J.-Y., 2006. Exploiting knowledge about future demands for real-time vehicle dispatching. *Transportation Science*, 40, 2 (2006), 211–225. doi:10.1287/trsc.1050.0114. http://pubsonline.informs.org/doi/abs/10.1287/trsc.1050.0114. (cited on page 120)

Johnson, D. S., 1982. The np-completeness column: An ongoing guide. *Journal of Algorithms*, 3, 2 (1982), 182–195. (cited on page 136)

Jozefowiez, N.; Semet, F.; and Talbi, E.-G., 2008. Multi-objective vehicle routing problems. *European Journal of Operational Research*, 189, 2 (2008), 293 – 309. doi:http://dx.doi.org/10.1016/j.ejor.2007.05.055. http://www.sciencedirect.com/science/article/pii/S0377221707005498. (cited on pages 9 and 85)

Karp, R. M., 1972. *Reducibility among combinatorial problems*. Springer. (cited on page 5)

Kershner, R., 1939. The number of circles covering a set. *American Journal of mathematics*, (1939), 665–671. (cited on page 136)

Kilby, P.; Prosser, P.; and Shaw, P., 2000. A comparison of traditional and constraint-based heuristic methods on vehicle routing problems with side constraints. *Constraints*, 5, 4 (2000), 389–414. doi:10.1023/A:1009808327381. (cited on pages 23 and 33)

Kilby, P. and Shaw, P., 2006. Vehicle routing. In *Handbook of constraint programming* (Eds. F. Rossi; P. Van Beek; and T. Walsh), chap. 23. Elsevier. (cited on page 23)

Kilby, P. and Urli, T., 2015. Fleet design optimisation from historical data using constraint programming and large neighbourhood search. *Constraints*, (2015), 1–20. doi:10.1007/s10601-015-9203-0. http://dx.doi.org/10.1007/s10601-015-9203-0. (cited on pages 33 and 34)

Kilby, P. and Verden, A., 2011. Flexible routing combing constraint programming, large neighbourhood search, and feature-based insertion. In *Proceedings 2nd Workshop on Artificial Intelligence and Logistics (AILOG 2011)*, 43–49. (cited on page 125)

Kim, B.-I.; Li, H.; and Johnson, A. L., 2013. An augmented large neighborhood search method for solving the team orienteering problem. *Expert Systems with Applications*, 40, 8 (2013), 3065–3072. (cited on page 32)

Kirkpatrick, S.; Gelatt, C. D.; Vecchi, M. P.; et al., 1983. Optimization by simulated annealing. *science*, 220, 4598 (1983), 671–680. (cited on page 29)

Klein Haneveld, W. K. and van der Vlerk, M. H., 1999. Stochastic integer programming: General models and algorithms. *Annals of Operations Research*, 85 (1999), 39–57. (cited on page 20)

Koffman, D., 2004. *Operational experiences with flexible transit services*, vol. 53. Transportation Research Board. (cited on page 35)

Kytöjoki, J.; Nuortio, T.; Bräysy, O.; and Gendreau, M., 2007. An efficient variable neighborhood search heuristic for very large scale vehicle routing problems. *Computers & Operations Research*, 34, 9 (2007), 2743 – 2757. doi:http://dx.doi.org/10.1016/j.cor.2005.10.010. http://www.sciencedirect.com/science/article/pii/S0305054805003394. (cited on page 31)

Labadie, N.; Mansini, R.; Melechovský, J.; and Calvo, R. W., 2012. The team orienteering problem with time windows: An lp-based granular variable neighborhood search. *European Journal of Operational Research*, 220, 1 (2012), 15 – 27. doi:http://dx.doi.org/10.1016/j.ejor.2012.01.030. http://www.sciencedirect.com/science/article/pii/S0377221712000653. (cited on page 31)

Laporte, G., 2009. Fifty years of vehicle routing. *Transportation Science*, 43, 4 (11 2009), 408–416. http://search.proquest.com/docview/203543324?accountid=8330. (cited on page 6)

Laporte, G.; Gendreau, M.; Potvin, J.-Y.; and Semet, F., 2000. Classical and modern heuristics for the vehicle routing problem. *International Transactions in Operational Research*, 7, 4-5 (2000), 285–300. doi:10.1111/j.1475-3995.2000.tb00200.x. http://dx.doi.org/10.1111/j.1475-3995.2000.tb00200.x. (cited on page 25)

Laporte, G.; Louveaux, F.; and Mercure, H., 1992. The vehicle routing problem with stochastic travel times. *Transportation science*, 26, 3 (1992), 161–170. (cited on page 47)

Laws, R.; Enoch, M. P.; Ison, S. G.; and Potter, S., 2009. Demand responsive transport: a review of schemes in england and wales. *Journal of Public Transportation*, (2009). (cited on page 35)

Levine, J.; Hong, Q.; Hug, G. E.; and Rodriguez, D., 2000. Impacts of an advanced public transportation system demonstration project. *Transportation Research Record: Journal of the Transportation Research Board*, 1735, 1 (2000), 169–177. (cited on page 35)

Li, F.; Golden, B.; and Wasil, E., 2005. Very large-scale vehicle routing: new test problems, algorithms, and results. *Computers & Operations Research*, 32, 5 (2005), 1165–1179. (cited on page 34)

Li, J.-Q.; Mirchandani, P. B.; and Borenstein, D., 2009. A lagrangian heuristic for the real-time vehicle rescheduling problem. *Transportation Research Part E: Logistics and Transportation Review*, 45, 3 (2009), 419 – 433. doi:http://dx.doi.org/10.1016/j.tre.2008.09.002. http://www.sciencedirect.com/science/article/pii/S1366554508001166. (cited on page 47)

Li, R., 2012. *Vehicle routing models in public safety and health care*. Ph.D. thesis, The University of Alabama TUSCALOOSA. (cited on page 9)

Li, X. and Quadrifoglio, L., 2009. Optimal zone design for feeder transit services. *Transportation Research Record: Journal of the Transportation Research Board*, 2111, 1 (2009), 100–108. (cited on page 37)

Li, X. and Quadrifoglio, L., 2010. Feeder transit services: Choosing between fixed and demand responsive policy. *Transportation Research Part C: Emerging Technologies*, 18, 5 (2010), 770 – 780. doi:10.1016/j.trc.2009.05.015. http://www.sciencedirect.com/science/article/pii/S0968090X09000801. Applications of Advanced Technologies in Transportation: Selected papers from the 10th AATT Conference. (cited on page 37)

Liebchen, C., 2007. Periodic timetable optimization in public transport. In *Operations Research Proceedings 2006* (Eds. K.-H. Waldmann and U. Stocker), vol. 2006 of *Operations Research Proceedings*, 29–36. Springer Berlin Heidelberg. ISBN 978-3-540-69994-1. doi:10.1007/978-3-540-69995-8_5. http://dx.doi.org/10.1007/978-3-540-69995-8_5. (cited on page 34)

Lin, S.; Ying, K.; Lee, Z.; and Hsi, F., 2006. Applying simulated annealing approach for capacitated vehicle routing problems. In *Systems, Man and Cybernetics, 2006. SMC '06. IEEE International Conference on*, vol. 1, 639–644. doi:10.1109/ICSMC.2006.384457. (cited on page 29)

Liu, S.; Huang, W.; and Ma, H., 2009. An effective genetic algorithm for the fleet size and mix vehicle routing problems. *Transportation Research Part E: Logistics and Transportation Review*, 45, 3 (2009), 434 – 445. doi:http://dx.doi.org/10.1016/j.tre.2008.10.003. http://www.sciencedirect.com/science/article/pii/S1366554508001324. (cited on page 33)

M. Bellmore, G. L. N., 1968. The traveling salesman problem: A survey. *Operations Research*, 16, 3 (1968), 538–558. http://www.jstor.org/stable/168581. (cited on page 24)

Martello, S. and Toth, P., 1990. *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley and Sons, Inc., New York, NY, USA. ISBN 0-471-92420-2. (cited on page 16)

Martí, R.; Moreno-Vega, J. M.; and Duarte, A., 2010. Advanced multi–start methods. In *Handbook of metaheuristics*, 265–281. Springer. (cited on page 57)

Masson, R.; Lehuédé, F.; and Péton, O., 2014. The dial–a–ride problem with transfers. *Computers & Operations Research*, 41 (2014), 12–23. (cited on page 53)

McGuckin, N. A. and Srinivasan, N., 2003. Journey to work in the united states and its major metropolitan areas–1960-2000. Technical report, Federal Highway Administration. (cited on pages 50 and 65)

Metropolis, N.; Rosenbluth, A. W.; Rosenbluth, M. N.; Teller, A. H.; and Teller, E., 1953. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21, 6 (1953), 1087–1092. (cited on page 29)

Mladenović, N. and Hansen, P., 1997. Variable neighborhood search. *Computers & Operations Research*, 24, 11 (1997), 1097–1100. (cited on page 30)

Moccia, L.; Cordeau, J.-F.; and Laporte, G., 2012. An incremental tabu search heuristic for the generalized vehicle routing problem with time windows. *Journal of the Operational Research Society*, 63, 2 (2012), 232–244. (cited on page 30)

Montemanni, R.; Gambardella, L. M.; Rizzoli, A. E.; and Donati, A. V., 2005. Ant colony system for a dynamic vehicle routing problem. *Journal of Combinatorial Optimization*, 10, 4 (2005), 327–343. (cited on page 48)

Mu, Q.; Fu, Z.; Lysgaard, J.; and Eglese, R., 2011. Disruption management of the vehicle routing problem with vehicle breakdown. *Journal of the Operational Research Society*, 62, 4 (2011), 742–749. (cited on page 47)

Mulley, C. and Nelson, J. D., 2009. Flexible transport services: A new market opportunity for public transport. *Research in Transportation Economics*, 25, 1 (2009), 39–45. (cited on page 35)

Nazif, H. and Lee, L. S., 2012. Optimised crossover genetic algorithm for capacitated vehicle routing problem. *Applied Mathematical Modelling*, 36, 5 (2012), 2110 – 2117. doi:http://dx.doi.org/10.1016/j.apm.2011.08.010. http://www.sciencedirect.com/science/article/pii/S0307904X11005105. (cited on page 33)

Nelson, J. D.; Wright, S.; Masson, B.; Ambrosino, G.; and Naniopoulos, A., 2010. Recent developments in flexible transport services. *Research in Transportation Economics*, 29, 1 (2010), 243 – 248. (cited on page 35)

Ordóñez, F., 2010. Robust vehicle routing. *TUTORIALS in Oprations Research*, (2010), 153–178. (cited on page 47)

Paquette, J.; Cordeau, J.-F.; Laporte, G.; and Pascoal, M. M., 2013. Combining multi-criteria analysis and tabu search for dial-a-ride problems. *Transportation Research Part B: Methodological*, 52 (2013), 1 – 16. doi:http://dx.doi.org/10.1016/j.trb.2013.02.007. http://www.sciencedirect.com/science/article/pii/S0191261513000337. (cited on page 30)

Parragh, S. N., 2011. Introducing heterogeneous users and vehicles into models and algorithms for the dial-a-ride problem. *Transportation Research Part C: Emerging Technologies*, 19, 5 (2011), 912 – 930. doi:http://dx.doi.org/10.1016/j.trc.2010.06.002. http://www.sciencedirect.com/science/article/pii/S0968090X10001075. (cited on page 40)

Parragh, S. N.; Doerner, K. F.; and Hartl, R. F., 2008. A survey on pickup and delivery problems. *Journal für Betriebswirtschaft*, 58, 1 (2008), 21–51. doi:10.1007/s11301-008-0033-7. http://dx.doi.org/10.1007/s11301-008-0033-7. (cited on page 11)

Pearson, K., 1901. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2, 11 (1901), 559–572. (cited on page 128)

Pedersen, J.; Kocsis, D.; Tripathi, A.; Tarrell, A.; Weerakoon, A.; Tahmasbi, N.; Xiong, J.; Deng, W.; Oh, O.; and De Vreede, G.-J., 2013. Conceptual foundations of crowdsourcing: A review of is research. In *System Sciences (HICSS), 2013 46th Hawaii International Conference on*, 579–588. doi:10.1109/HICSS.2013.143. (cited on page 43)

Pesant, G. and Gendreau, M., 1999. A constraint programming framework for local search methods. *Journal of Heuristics*, 5, 3 (1999), 255–279. (cited on page 33)

Pillac, V.; Guéret, C.; and Medaglia, A. L., 2012. An event-driven optimization framework for dynamic vehicle routing. *Decision Support Systems*, 54, 1 (2012), 414 – 423. doi:http://dx.doi.org/10.1016/j.dss.2012.06.007. http://www.sciencedirect.com/science/article/pii/S016792361200190X. (cited on page 47)

Pisinger, D. and Ropke, S., 2007. A general heuristic for vehicle routing problems. *Computers & operations research*, 34, 8 (2007), 2403–2435. (cited on pages 24, 32, and 126)

Pisinger, D. and Ropke, S., 2010. Large neighborhood search. In *Handbook of metaheuristics*, 399–419. Springer. (cited on page 32)

Potts, J. F.; Marshall, M. A.; Crockett, E. C.; and Washington, J., 2010. *A Guide for Planning and Operating Flexible Public Transportation Services*. Project B-35. Transportation Research Board of the National Academies. (cited on page 35)

Potvin, J.-Y. and Bengio, S., 1996. The vehicle routing problem with time windows part ii: genetic search. *INFORMS journal on Computing*, 8, 2 (1996), 165–172. (cited on page 32)

Potvin, J.-Y.; Duhamel, C.; and Guertin, F., 1996. A genetic algorithm for vehicle routing with backhauling. *Applied Intelligence*, 6, 4 (1996), 345–355. (cited on page 32)

Potvin, J.-Y. and Rousseau, J.-M., 1993. A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *European Journal of Operational Research*, 66, 3 (1993), 331–340. (cited on page 24)

Powell, W. B., 2007. *Approximate Dynamic Programming: Solving the curses of dimensionality*, vol. 703. John Wiley & Sons. (cited on page 48)

Powell, W. B. and Topaloglu, H., 2003. Stochastic programming in transportation and logistics. *Handbooks in operations research and management science*, 10 (2003), 555–635. (cited on page 20)

Powell, W. B. and Topaloglu, H., 2005. Fleet management. In *Applications of Stochastic Programming, Math Programming Society - SIAM Series on Optimization*, 185–216. (cited on page 120)

Prescott-Gagnon, E.; Desaulniers, G.; and Rousseau, L.-M., 2009. A branch-and-price-based large neighborhood search algorithm for the vehicle routing problem with time windows. *Networks*, 54, 4 (2009), 190–204. (cited on page 32)

Prins, C., 2004. A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research*, 31, 12 (2004), 1985 – 2002. doi:http://dx.doi.org/10.1016/S0305-0548(03)00158-8. http://www.sciencedirect.com/science/article/pii/S0305054803001588. (cited on pages 32 and 33)

Prins, C., 2009. Two memetic algorithms for heterogeneous fleet vehicle routing problems. *Engineering Applications of Artificial Intelligence*, 22, 6 (2009), 916 – 928. doi:http://dx.doi.org/10.1016/j.engappai.2008.10.006. http://www.sciencedirect.com/science/article/pii/S0952197608001693. (cited on page 33)

Quadrifoglio, L.; Dessouky, M. M.; and Ordonez, F., 2008. A simulation study of demand responsive transit system design. *Transportation Research Part A: Policy and Practice*, 42, 4 (2008), 718 – 737. doi:10.1016/j.tra.2008.01.018. http://www.sciencedirect.com/science/article/pii/S0965856408000256. (cited on page 53)

Renaud, J.; Laporte, G.; and Boctor, F. F., 1996. A tabu search heuristic for the multi-depot vehicle routing problem. *Computers & Operations Research*, 23, 3 (1996), 229–235. (cited on page 30)

Ritzinger, U.; Puchinger, J.; and Hartl, R. F., 2015. A survey on dynamic and stochastic vehicle routing problems. *International Journal of Production Research*, (2015), 1–17. (cited on page 47)

Røpke, S.; Cordeau, J.-F.; and Laporte, G., 2007. Models and a branch-and-cut algorithm for pickup and delivery problems with time windows. *Networks*, 49, 4 (2007), 258–272. (cited on page 40)

Ropke, S. and Pisinger, D., 2006a. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation science*, 40, 4 (2006), 455–472. (cited on pages 32 and 120)

Ropke, S. and Pisinger, D., 2006b. A unified heuristic for a large class of vehicle routing problems with backhauls. *European Journal of Operational Research*, 171, 3 (2006), 750 – 775. doi:http://dx.doi.org/10.1016/j.ejor.2004.09.004. http://www.sciencedirect.com/science/article/pii/S0377221704005831. (cited on page 32)

Rossi, F.; Beek, P. v.; and Walsh, T., 2006. *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*. Elsevier Science Inc., New York, NY, USA. ISBN 0444527265. (cited on page 23)

Rourke, J. O.; Supowit, K.; et al., 1983. Some np-hard polygon decomposition problems. *Information Theory, IEEE Transactions on*, 29, 2 (1983), 181–190. (cited on page 136)

Rousseau, L.-M.; Gendreau, M.; and Pesant, G., 2002. Using constraint-based operators to solve the vehicle routing problem with time windows. *Journal of heuristics*, 8, 1 (2002), 43–58. (cited on pages 33 and 34)

Salazar-Aguilar, M. A.; Langevin, A.; and Laporte, G., 2014. The multi-district team orienteering problem. *Computers & Operations Research*, 41 (2014), 76 – 82. doi:http://dx.doi.org/10.1016/j.cor.2013.07.026. http://www.sciencedirect.com/science/article/pii/S0305054813002062. (cited on page 32)

Savelsbergh, M. W., 1992. The vehicle routing problem with time windows: Minimizing route duration. *ORSA journal on computing*, 4, 2 (1992), 146–154. (cited on page 25)

Schwartz, J.; Steger, A.; and Weissl, A., 2005. Fast algorithms for weighted bipartite matching. In *Experimental and efficient algorithms*, 476–487. Springer. (cited on page 43)

Semet, F.; Toth, P.; and Vigo, D., 2014. Classical exact algorithms for the capacitated vehicle routing problem. *Vehicle Routing: Problems, Methods, and Applications*, 18 (2014), 37. (cited on page 6)

Sevilla, F. C. and de Blas, C. S., 2003. Vehicle routing problem with time windows and intermediate facilities. *SEIO 003 Edicions de la Universitat de Lleida*, (2003), 3088–3096. (cited on page 8)

Sevkli, Z. and Sevilgen, F. E., 2006. Variable neighborhood search for the orienteering problem. In *Computer and Information Sciences–ISCIS 2006*, 134–143. Springer. (cited on page 31)

Shapiro, A. and Philpott, A., 2007. A tutorial on stochastic programming. *Manuscript*, (2007). (cited on page 20)

Shaw, P., 1998. Using constraint programming and local search methods to solve vehicle routing problems. In *Principles and Practice of Constraint Programming — CP98* (Eds. M. Maher and J.-F. Puget), vol. 1520 of *Lecture Notes in Computer Science*, 417–431. Springer Berlin Heidelberg. (cited on page 31)

Shen, C.-W. and Quadrifoglio, L., 2012. Evaluation of zoning design with transfers for paratransit services. *Transportation Research Record: Journal of the Transportation Research Board*, 2277, 1 (2012), 82–89. (cited on page 53)

Simão, H. P.; Day, J.; George, A. P.; Gifford, T.; Nienow, J.; and Powell, W. B., 2009. An approximate dynamic programming algorithm for large-scale fleet management: A case application. *Transportation Science*, 43, 2 (2009), 178–197. doi:10.1287/trsc.1080.0238. (cited on page 48)

Solomon, M. M., 1987. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35, 2 (1987), pp. 254–265. http://www.jstor.org/stable/170697. (cited on page 124)

Spliet, R. and Gabor, A. F., 2014. The time window assignment vehicle routing problem. *Transportation Science*, (2014). (cited on page 11)

Subramanian, A., 2012. *Heuristic, exact and hybrid approaches for vehicle routing problems*. Ph.D. thesis, Ph. D. thesis, Universidade Federal Fluminense, Niterói, Brazil. (cited on page 5)

Sun, C.; Zhou, W.; and Wang, Y., 2008. Scheduling combination and headway optimization of bus rapid transit. *Journal of Transportation Systems Engineering and Information Technology*, 8, 5 (2008), 61 – 67. doi:http://dx.doi.org/10.1016/S1570-6672(08)60039-2. http://www.sciencedirect.com/science/article/pii/S1570667208600392. (cited on page 34)

Taillard, É.; Badeau, P.; Gendreau, M.; Guertin, F.; and Potvin, J.-Y., 1997. A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation science*, 31, 2 (1997), 170–186. (cited on pages 30 and 48)

Taillard, E. D.; Laporte, G.; and Gendreau, M., 1995. Vehicle routing with multiple use of vehicles. *THE JOURNAL OF THE OPERATIONAL RESEARCH SOCIETY*, 47, 8 (1995), 1065–1070. (cited on page 8)

Tang, H. and Miller-Hooks, E., 2005. A tabu search heuristic for the team orienteering problem. *Computers & Operations Research*, 32, 6 (2005), 1379–1407. (cited on page 9)

Tarantilis, C. and Kiranoudis, C., 2002. Distribution of fresh meat. *Journal of Food Engineering*, 51, 1 (2002), 85–91. (cited on page 8)

Tavakkoli-Moghaddam, R.; Gazanfari, M.; Alinaghian, M.; Salamatbakhsh, A.; and Norouzi, N., 2011. A new mathematical model for a competitive vehicle routing problem with time windows solved by simulated annealing. *Journal of Manufacturing Systems*, 30, 2 (2011), 83 – 92. doi:http://dx.doi.org/10.1016/j.jmsy.2011.04.005. http://www.sciencedirect.com/science/article/pii/S0278612511000380. (cited on page 29)

Teodorović, D. and Pavković, G., 1992. A simulated annealing technique approach to the vehicle routing problem in the case of stochastic demand. *Transportation Planning and Technology*, 16, 4 (1992), 261–273. (cited on page 29)

Thangiah, S.; Vinayagamoorthy, R.; and Gubbi, A., 1993. Vehicle routing with time deadlines using genetic and local algorithm. In *Proc. of The Fifth Int. Conf. on Genetic Algorithms*, 506–513. (cited on page 32)

Thangiah, S. R., 1993. *Vehicle routing with time windows using genetic algorithms*. Citeseer. (cited on page 32)

Thangiah, S. R., 1995. An adaptive clustering method using a geometric shape for vehicle routing problems with time windows. In *Proceedings of the 6th International Conference on Genetic Algorithms*, 536–545. Morgan Kaufmann Publishers Inc. (cited on page 32)

Thangiah, S. R. and Gubbi, A. V., 1993. Effect of genetic sectoring on vehicle routing problems with time windows. In *Developing and Managing Intelligent System Projects, 1993., IEEE International Conference on*, 146–153. IEEE. (cited on page 32)

Thangiah, S. R.; Nygard, K. E.; and Juell, P. L., 1991. Gideon: A genetic algorithm system for vehicle routing with time windows. In *Artificial Intelligence Applications, 1991. Proceedings., Seventh IEEE Conference on*, vol. 1, 322–328. IEEE. (cited on page 32)

Toth, P. and Vigo, D., 2003. The granular tabu search and its application to the vehicle-routing problem. *INFORMS J. on Computing*, 15, 4 (Dec. 2003), 333–346. doi:10.1287/ijoc.15.4.333.24890. http://dx.doi.org/10.1287/ijoc.15.4.333.24890. (cited on pages 25 and 30)

Tricoire, F.; Romauch, M.; Doerner, K. F.; and Hartl, R. F., 2010. Heuristics for the multi-period orienteering problem with multiple time windows. *Computers & Operations Research*, 37, 2 (2010), 351 – 367. doi:http://dx.doi.org/10.1016/j.cor.2009.05.012. http://www.sciencedirect.com/science/article/pii/S030505480900152X. (cited on page 31)

Uchoa, E.; Pecin, D.; Pessoa, A.; Poggi, M.; Subramanian, A.; and Vidal, T., 2014. New benchmark instances for the capacitated vehicle routing problem. (2014). (cited on page 34)

Ursani, Z.; Essam, D.; Cornforth, D.; and Stocker, R., 2011. Localized genetic algorithm for vehicle routing problem with time windows. *Applied Soft Computing*, 11, 8 (2011), 5375 – 5390. doi:http://dx.doi.org/10.1016/j.asoc.2011.05.021. http://www.sciencedirect.com/science/article/pii/S1568494611001803. (cited on page 33)

Van Oort, N., 2014. Big data opportunities in public transport: Enhancing public transport by itcs. In *IT-TRANS, 18-20 February 2014, Karlsruhe, Germany*. (cited on page 34)

Vanderbei, R., 2014. *Linear Programming: Foundations and Extensions*, vol. 114 of *International Series in Operations Research and Management Science*. Springer US. ISBN 978-0-387-74387-5. (cited on page 12)

Velaga, N. R.; Nelson, J. D.; Wright, S. D.; and Farrington, J. H., 2012. The potential role of flexible transport services in enhancing rural public transport provision. *Journal of Public Transportation*, 15, 1 (2012), 111–131. (cited on page 35)

Vidal, T.; Crainic, T. G.; Gendreau, M.; and Prins, C., 2013a. Heuristics for multi-attribute vehicle routing problems: A survey and synthesis. *European Journal of Operational Research*, 231, 1 (2013), 1 – 21. doi:http://dx.doi.org/10.1016/j.ejor.2013.02.053. http://www.sciencedirect.com/science/article/pii/S0377221713002026. (cited on page 23)

Vidal, T.; Crainic, T. G.; Gendreau, M.; and Prins, C., 2013b. A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers & Operations Research*, 40, 1 (2013), 475 – 489. doi:http://dx.doi.org/10.1016/j.cor.2012.07.018. http://www.sciencedirect.com/science/article/pii/S0305054812001645. (cited on page 33)

Vidal, T.; Crainic, T. G.; Gendreau, M.; and Prins, C., 2014. A unified solution framework for multi-attribute vehicle routing problems. *European Journal of Operational Research*, 234, 3 (2014), 658 – 673. doi:http://dx.doi.org/10.1016/j.ejor.2013.09.045. http://www.sciencedirect.com/science/article/pii/S037722171300800X. (cited on page 33)

Waters, C. D. J., 1989. Vehicle-scheduling problems with uncertainty and omitted customers. *The Journal of the Operational Research Society*, 40, 12 (1989), 1099–1108. http://www.jstor.org/stable/2582919. (cited on page 47)

Wolsey, L. A. and Nemhauser, G. L., 2014. *Integer and combinatorial optimization*. John Wiley & Sons. (cited on pages 12 and 40)

Wong, R. C. W.; Yuen, T. W. Y.; Fung, K. W.; and Leung, J. M. Y., 2008. Optimizing timetable synchronization for rail mass transit. *Transportation Science*, 42, 1 (2008), 57–69. doi:10.1287/trsc.1070.0200. http://dx.doi.org/10.1287/trsc.1070.0200. (cited on page 34)

Wren, A. and Holliday, A., 1972. Computer scheduling of vehicles from one or more depots to a number of delivery points. *Operational Research Quarterly (1970-1977)*, 23, 3 (1972), pp. 333–344. http://www.jstor.org/stable/3007888. (cited on page 24)

Wright, S., 2013. Designing flexible transport services: guidelines for choosing the vehicle type. *Transportation Planning and Technology*, 36, 1 (2013), 76–92. (cited on page 35)

Yan, Y.; Meng, Q.; Wang, S.; and Guo, X., 2012. Robust optimization model of schedule design for a fixed bus route. *Transportation Research Part C: Emerging Technologies*, 25 (2012), 113 – 121. doi:http://dx.doi.org/10.1016/j.trc.2012.05.006. http://www.sciencedirect.com/science/article/pii/S0968090X12000812. (cited on page 34)

Yang, J.; Jaillet, P.; and Mahmassani, H., 2004. Real-time multivehicle truckload pickup and delivery problems. *Transportation Science*, 38, 2 (2004), 135–148. (cited on page 48)