The Pickup and Multiple Delivery Problem

Philip James Mourdjis

Doctor of Engineering

University of York

Computer Science

September 2016

Abstract

This thesis presents my work on the pickup and multiple delivery problem, a real-world vehicle routing and scheduling problem with soft time windows, working time and last-in-first-out constraints, developed in collaboration with Transfaction Ltd., who conduct logistics analysis for several large retailers in the UK. A summary of relevant background literature is presented highlighting where my research fits into and contributes to the broader academic landscape. I present a detailed model of the problem and thoroughly analyse a case-study data set, obtaining distributions used for further research. A new variable neighbourhood descent with memory hyper-heuristic is presented and shown to be an effective technique for solving instances of the real-world problem. I analyse strategies for cooperation and competition amongst haulage companies and quantify their effectiveness. The value of time and timely information for planning pickup and delivery requests is investigated. The insights gained are of real industrial relevance, highlighting how a variety of business decisions can produce significant cost savings.

Contents

Abstract 2

List of Tables 11

List of Figures 12

Acknowledgements 17

Declaration 19

1. Introduction 23

1.1 Thesis Outline 25

2. Literature Review: Related Problems 27

2.1 Routing Problems 28

2.1.1 Travelling Salesperson Problem (TSP) 28

2.1.2 Vehicle Routing Problem (VRP) 29

2.1.3 Pickup and Delivery Problem (PDP) 29

2.2 Real-world Problems 32

2.2.1 Multiple Vehicles and Depots 33

2.2.2 Heterogeneous Fleets 33

2.2.3 Last-in, First-out (LIFO) Loading 34

2.2.4 Green Logistics 34

2.2.5 Time Windows Constraints 34

2.2.6 Dynamic Routing Problems 35

2.2.7 Foresight Policy . . . 36
2.2.8 Stochastic Travel Times . . . 37
2.2.9 Competition . . . 38
2.2.10 Business Design Issues . . . 39
2.3 Fitness Functions . . . 40
2.4 Benchmark Instances . . . 40
2.5 Discussion and Summary . . . 41

3. Literature Review: Solution Approaches . . . 43
3.1 Exact Methods . . . 43
3.2 Non-Exact Methods . . . 44
3.2.1 Construction Operators . . . 45
3.2.1.1 Random Insertion . . . 45
3.2.1.2 Greedy Insertion . . . 45
3.2.1.3 Clarke and Wright Savings (CWS) . . . 46
3.2.1.4 Generalized Insertion (GENI) . . . 47
3.2.1.5 Least Regret Insertion . . . 47
3.2.2 Local Search Operators (LSOs) . . . 47
3.2.2.1 Relocate . . . 49
3.2.2.2 λ-interchange . . . 49
3.2.2.3 Exchange . . . 50
3.2.2.4 Chain-exchange . . . 50
3.2.2.5 CROSS-exchange . . . 50
3.2.2.6 iCROSS-exchange . . . 50
3.2.2.7 Insert Related Parallel . . . 51
3.2.2.8 Cycle-exchange . . . 51
3.2.2.9 Ejection Chains . . . 51
3.2.2.10 Edge Assembly Crossover . . . 52
3.2.2.11 Discussion: Modification, LSOs . . . 53
3.2.3 Simple Heuristics . . . 53
3.2.3.1 Sequential Insertion . . . 53

3.2.3.2 Monte Carlo Techniques 54
3.2.3.3 Additional Methods 54
3.2.4 Meta-heuristics (MH) 55
3.2.4.1 Simulated Annealing (SA) 56
3.2.4.2 Tabu search for PDPL 56
3.2.4.3 Ant Colony Optimisation (ACO) 56
3.2.5 Hyper-heuristics (HH) 57
3.2.5.1 Random Descent (RD) 58
3.2.5.2 Variable Neighbourhood Search (VNS) 58
VNS for VRP with time windows 59
VNS for MDVRP with time windows 61
VNS for DARP 61
VNS for TSPPDL 61
VNS for PDPL 62
3.2.5.3 Greedy Randomised Adaptive Search Procedure 62
3.2.5.4 Large Neighbourhood Search (LNS) 62
3.2.5.5 Adaptive Large Neighbourhood Search (ALNS) 63
3.2.5.6 Parallel ALNS 63
3.2.5.7 Q-Learning Selection (QL) 64
3.2.5.8 Binary Exponential Back-off (BEBO) 64
3.2.6 Hybrid Meta-heuristics (HM) 65
3.2.6.1 Hybrid Variable Neighbourhood Tabu Search 65
3.2.6.2 Two Phase Meta-heuristics 66
3.2.6.3 Hybrid Genetic Algorithm 66
3.2.6.4 Memetic algorithms 67
3.2.7 Dynamic Solutions 68
3.2.7.1 Reactive Vehicle Routing Framework 68
3.2.7.2 Multiple Plan Approach 68
3.2.7.3 Multiple Scenario Approach 68
3.2.7.4 Online Stochastic Programming 69
3.3 Discussion and Summary 69

4. Model: The Pickup and Multiple Delivery Problem (PMDP) 71
4.1 Introduction 72
4.2 Consignments 72
4.2.1 Time Windows 74
4.3 Vehicles 75
4.3.1 Capacity Constraints 76
4.4 Objective - Cost Function 77
4.4.1 Distance Cost 78
4.4.2 Time Cost 79
4.4.3 Delay Cost 79
4.5 Constraints 79
4.6 Discussion and Summary 81

5. Solution Methods: Heuristics for PMDP 83
5.1 Basics of Route Modification 84
5.2 Local Search Operators (LSOs) 84
5.2.1 Assumptions 85
5.2.2 Single Route Operators 86
5.2.3 Dual Route Operators 87
5.2.4 Shake Operators 88
5.3 Variable Neighbourhood Descent with Memory (VNDM) 88
5.3.1 Memory 90
5.3.2 Use of LSOs 91
5.4 Comparing Heuristics for the PDP 92
5.4.1 Parameter Tuning 93
5.4.2 Comparison on Benchmark Instances 94
5.5 Discussion and Summary 97

6. Solution Methods: Real-world PMDP 99
6.1 UK Retailers Case Study 99
6.1.1 Location, Linehauls and Backhauls 100
6.1.2 Load 101

6.1.3 Time . . . 101
6.1.3.1 Consignment Received Time . . . 102
6.1.3.2 Pickup Time . . . 104
6.1.3.3 Service Time . . . 106
6.1.3.4 Journey Time . . . 108
6.1.3.5 Delivery Time . . . 109
6.1.3.6 Arrival Time Window . . . 109
6.1.3.7 Delays . . . 110
6.1.4 Consignment Generation Process . . . 111
6.2 Department Store Case Study . . . 112
6.3 A Dynamic Problem . . . 112
6.3.1 Discrete Event Simulation (DES) . . . 112
6.4 Comparing Heuristics for the PMDP . . . 114
6.5 Discussion and Summary . . . 117

7. Cooperation and Competition . . . 119
7.1 Introduction . . . 119
7.2 Cooperation . . . 120
7.2.1 Simple Cooperative Strategies . . . 121
7.2.2 More Group Configurations . . . 123
7.2.3 Carrier Group Size . . . 125
7.3 Competition . . . 126
7.3.1 Equally Sized Companies. . . . 127
7.3.2 Differently Sized Companies. . . . 127
7.3.3 Large vs Small Companies. . . . 128
7.4 Discussion and Summary . . . 130

8. The Value of Time and Timely Information . . . 133
8.1 Introduction . . . 133
8.1.1 Parameters for Analysis . . . 134
8.2 Scenario 1 - Department Store Case Study . . . 136
8.3 Scenario 2 - Department Store, Heterogeneous Consignments . . . 139

8.4 Scenario 3 - UK Retailers Case Study . . . 143
8.5 Scenario 4 - Retailers, Heterogeneous Consignments . . . 145
8.6 Impact of Time Windows . . . 148
8.6.1 Degree of Dynamism . . . 149
8.7 Discussion and Summary . . . 151

9. Conclusions . . . 155
9.1 Key Findings . . . 155
9.2 Contributions . . . 158
9.3 Future Work . . . 158
9.3.1 Economic Impact . . . 158
9.3.2 Breakdowns and Unpredictability . . . 159
9.3.3 Green Logistics . . . 159
9.3.4 Hyper-heuristic Research . . . 159

Appendix . . . 161
A.1 Extended Glossary . . . 161
A.1.1 Simple LSOs . . . 161
A.1.1.1 k-opt . . . 161
A.1.1.2 Or-opt . . . 161
A.1.1.3 I-opt . . . 162
A.1.1.4 2-opt* . . . 162
A.1.2 Tabu Search (TS) . . . 162
A.1.2.1 Taburoute . . . 163
A.1.2.2 Tabu search with adaptive memory . . . 164
A.1.2.3 Tabu search for the VRP with soft time windows . . . 164
A.1.2.4 Tabu search for the dynamic VRP . . . 165
A.1.3 Genetic Algorithms (GA) . . . 166
A.2 Generated Times . . . 167
A.3 Full Comparison Tables . . . 169

Bibliography . . . 169

List of Tables

4.1 Adapted constraints for the PMDP. 80

4.2 New constraints for the PMDP. 81

5.1 HH Performance on 100 clustered customer benchmarks. 95

5.2 HH Performance on 400 clustered customer benchmarks. 96

6.1 Modelled probabilities for received day. 103

7.1 Percentage of the sample carrier's consignments re-allocated. 123

8.1 Manual versus optimised schedules. 136

8.2 Edod-tw for homogeneous scenarios. 150

8.3 Edod-tw for heterogeneous scenarios. 150

A.1 HH Performance on 100 random customer benchmarks. 169

A.2 HH Performance on 100 random and clustered customer benchmarks. . 170

A.3 HH Performance on 200 clustered customer benchmarks. 170

A.4 HH Performance on 200 random customer benchmarks. 171

A.5 HH Performance on 200 random and clustered customer benchmarks. . 171

A.6 HH Performance on 400 random customer benchmarks. 172

A.7 HH Performance on 400 random and clustered customer benchmarks. . 172

List of Figures

2.1 An example TSP. 28
2.2 Key to operator figures. 30
2.3 An example PDP. 31
2.4 Request time window 34

3.1 GENI operator. 48
3.2 Relocate operator 49
3.3 Exchange operator. 50
3.4 CROSS exchange operator. 51
3.5 iCROSS exchange operator. 52

4.1 An example PMDP. 73
4.2 Arrival time window for each request. 74
4.3 Time windows for a consignment with two deliveries. 74
4.4 Combining two consignments. 77
4.5 Fuel consumption at load. 78
4.6 Time costs around an arrival time window. 79

5.1 An example of consignment insertion. 84
5.2 Single route operators. 86
5.3 Dual route relocate operators. 87
5.4 Dual route swap operators. 88
5.5 Effect of memory 91
5.6 Average rank of heuristics on different sized problems. 94

5.7 Number of routes versus cost for benchmark instances. 97

6.1 Real-world locations. 101
6.2 Real-world consignment received times and approximating distribution. 102
6.3 Real-world planning windows and aproximating function. 104
6.4 Real-world distribution of pickup times. 105
6.5 Real-world pickup and delivery service times. 106
6.6 Real-world loading vs unloading service times. 107
6.7 Real-world time vs distance and approximating function. 108
6.8 Estimated delivery time. 109
6.9 Planning and arrival windows for a consignment. 109
6.10 Real-world pickup and delivery delay. 110
6.11 An example daily schedule. 113
6.12 Average performance over CPU time. 115
6.13 Distribution of final cost across 100 runs. 115
6.14 Number of routes versus cost for real data set. 116

7.1 Average costs with four different models of cooperation. 121
7.2 Percentage of assigned consignments serviced. 122
7.3 Cost per request for different carrier group configurations. 124
7.4 Scheduled consignments for different carrier group configurations. 125
7.5 Effect of carrier group size. 126
7.6 Effect of cooperation on equally sized companies. 128
7.7 Effect of cooperation between small companies. 129
7.8 Effects of cooperation across different sized companies. 129
7.9 Effect of cooperation between large and small companies. 130

8.1 Scenario 1, average delay per request. 137
8.2 Scenario 1, average cost of servicing one request. 137
8.3 Scenario 1, total distance of solutions. 138
8.4 Scenario 1, cross sections of cost per request. 139
8.5 Scenario 2, utilisation across time windows. 140
8.6 Scenario 2, average delay per request. 140

8.7 Scenario 2, average cost of servicing one request. 141
8.8 Scenario 2, total distance of solutions. 141
8.9 Scenario 2, cross sections of cost per request. 142
8.10 Scenario 3, average cost of servicing one request. 143
8.11 Scenario 3, total distance of solutions. 144
8.12 Scenario 3, utilisation across time windows. 144
8.13 Scenario 3, average delay per request. 145
8.14 Scenario 4, average cost of servicing one request. 146
8.15 Scenario 4, average delay per request. 146
8.16 Scenario 4, total distance of solutions. 147
8.17 Scenario 4, utilisation across time windows. 147
8.18 Impact of increasing arrival window. 148
8.19 Impact of increasing planning window. 149
8.20 Edod-tw versus average request cost in scenario 1. 151

A.1 2-opt operator. 161
A.2 Or-opt operator. 162
A.3 2-opt* operator. 163
A.4 GA crossover operator. 167
A.5 Consignment received times vs generated 168

Acknowledgements

This thesis was funded by Transfaction Ltd. as part of an Engineering and Physical Sciences Research Council (EPSRC) grant. The insights gained would not have been possible without data and assistance provided by Martin Robinson, my EngD industrial supervisor.

I would like to thank Yujie Chen for pushing me to work harder than I otherwise would have done and for countless discussions on the merits of various techniques used throughout my research. I would also like to thank Dr. Daniel Whitehouse for letting me know about this opportunity and for his advice and warnings regarding the undertaking of a PhD.

I would also like to thank my supervisors Fiona Polack and Peter Cowling for their invaluable guidance and feedback, York Centre for Complex Systems Analysis (YCCSA) and the AI group at the University of York for engaging weekly seminars and cake.

Declaration

I declare that this thesis is a presentation of original work and I am the sole author. This work has not previously been presented for an award at this, or any other, University. All sources are acknowledged as References.

The research presented in this thesis includes work which has been previously published or submitted for publication. In these papers, I am the first author, and I have presented the research at the relevant conferences / workshops.

- **Meta-heuristics for the Pick-up and Delivery Problem**
  Philip Mourdjis, Peter Cowling and Martin Robinson (2014) - The 14th European Conference on Evolutionary Computation in Combinatorial Optimisation (EVOCOP) 2014.
  - Background work establishing the need for better approaches to address Transfaction Ltd.'s scheduling problems.

- **The Effect of Cooperation in Pickup and Mutliple Delivery Problems**
  Philip Mourdjis, Fiona Polack, Peter Cowling, Yujie Chen and Marin Robinson (2016b) - 5th International Conference on Operations Research and Enterprise Systems (ICORES) 2016.
  - A case study analysing a specific aspect of Transfaction Ltd.'s scheduling problem, the results of this are included in Chapter 7.

- **Variable Neighbourhood Descent with Memory: A Hybrid Meta-heuristic for Supermarket Resupply**

Philip Mourdjis, Yujie Chen, Fiona Polack, Peter Cowling and Martin Robinson (2016a) - 10th International Workshop on Hybrid Meta-heuristics (HM) 2016.

  - This paper introduces and shows the effectiveness of the VNDM approach used throughout the thesis, the content of this paper is presented with further discussion in Chapter 5.

- **Competition and Cooperation in Pickup and Multiple Delivery Problems**
  Philip Mourdjis, Fiona Polack, Peter Cowling, Yujie Chen and Marin Robinson Published by Springer in CCIS 695, Chapter 9, as an invited author.

  - This work extends Mourdjis et al. (2016b) with a more realistic model of cooperation, and is also presented in Chapter 7.

Submitted Papers

The following papers have been submitted for publication.

- **Time Windows and Timely Information in Routing and Scheduling**
  Philip Mourdjis, Peter Cowling, Marin Robinson, Fiona Polack and Yujie Chen Submitted to the Journal of the Operations Research Society (JORS).

  - The results of this paper are presented in Chapter 8.

Collaborative Work

For the collaborative works with Yujie Chen as first author and others, my contributions have been in the discussion and design of the hyper-heuristics used.

- **Dynamic optimisation of preventative and corrective maintenance schedules for a large scale urban drainage system**
  Yujie Chen, Peter Cowling, Fiona Polack, Stephen Remde and Philip Mourdjis (2017) - European Journal of Operational Research.

- **Risk driven analysis of maintenance for a large-scale drainage system**
  Yujie Chen, Fiona Polack, Peter Cowling, Philip Mourdjis and Stephen Remde

(2016b) - 5th International Conference on Operations Research and Enterprise Systems (ICORES) 2016 - Best Practical Paper.

- **Evaluating hyper-heuristics and local search operators for period routing problem**
  Yujie Chen, Philip Mourdjis, Fiona Polack, Peter Cowling (2016a) - The 16th European Conference on Evolutionary Computation in Combinatorial Optimisation (EVOCOP) 2016.

- **Risk driven analysis of maintenance for a large-scale drainage system**
  Yujie Chen, Fiona Polack, Peter Cowling, Philip Mourdjis and Stephen Remde Published by Springer in CCIS 695, Chapter 9, as an invited author.

Collaborative Submissions

- **A Multi-Arm-Bandit Neighbourhood Search for Routing and Scheduling Problems**
  Yujie Chen, Philip Mourdjis, Fiona Polack and Peter Cowling - Submitted as an invited author for extended EVOCOP papers.

1

Introduction

Logistics is a trillion-dollar industry[1], spanning businesses in both the public and private sectors. It is a large and complex global system incorporating millions of companies worldwide. Whether by aeroplane, train, ship or road vehicle, effective routing, scheduling and planning strategies are vital for an efficient logistical operation. On such a large scale, small improvements in efficiency can lead to large benefits through cost savings or faster service. A global shift towards on-demand, just-in-time delivery, along with increasing pressures on companies to reduce their energy consumption, means that it is of paramount importance for supply chains to be optimised for maximum utility, with business decisions which support this. My thesis presents techniques to help achieve this and analyses a number of real-world case studies, examining how these techniques and strategies can be used to reduce transportation costs in practice.

My Research has been guided by Transfaction Ltd.[2], a logistics analysis company, working with several large retailers in the UK to identify inefficiencies in their current logistics network. The more accurately a network can be modelled, the better their forecasts and analysis. Accurate forecasting, analysis and scheduling are valuable to Transfaction Ltd. and their clients allowing smarter business decisions to be made; this desire for competitive advantage drives every aspect of my research. Transfaction Ltd.'s problem has not been directly addressed before but shares many similarities with existing research on the pickup and delivery problem (PDP).

[1] https://www.selectusa.gov/logistics-and-transportation-industry-united-states
[2] http://www.transfaction.com/

Traditionally heuristic and meta-heuristic methods have been used to solve similar problems, while state-of-the-art methods employ hyper-heuristics or hybrid-metaheuristics. Heuristic methods generate approximate solutions and are useful for real-world problems as exact values for many parameters are not known. Travel and loading times are often estimates, fuel cost and usage also. For the type of large scale problems that are prevalent in the real world, heuristic methods also find acceptable solutions much faster than exact methods. Hyper-heuristics operate at a higher level than heuristic or meta-heuristic alternatives managing a set of low level heuristics (LLHs) to speed up the search process and find better solutions through mechanisms to escape locally optimal solutions. The state-of-the-art for PDPs with last-in-first-out (LIFO) loading constraints typically employs a variable neighbourhood search (VNS) hyper-heuristic and is concerned with finding solutions to synthetic problems. Little work has been published investigating the commercial implications of business decisions or strategies for cost reduction on real-world data. I investigate the savings that can be brought about by various delivery management strategies.

A significant contribution of my research is the introduction of the novel practical aim of finding ways to add additional consignments into pre-existing routes with a LIFO loading constraint. For this I present the *pickup and multiple delivery problem* (PMDP). The specific logistics problem my research focusses on is a medium distance truck routing problem in the UK. Working with Transfaction Ltd. I study the distribution network of medium to large firms (shippers) who rely on third party haulage companies (carriers) to deliver goods to customer partners. In PMDP, a pickup and its deliveries is referred to as a consignment. Vehicles leave their home depots empty and must fully service one or more consignments on route before returning to base. Each pickup has one or more associated deliveries with a given sequence. A sequence of deliveries may be interrupted at any point in order to fully service one or more nested consignments provided time windows and loading constraints are not violated. Shippers and customers are spread throughout the UK, so multiple carriers are needed. As a real-world problem, there are constraints that must be satisfied, such as vehicle capacity, soft time windows at pickup and delivery locations as well as driver working time rules per day and week. A key constraint, arising from the configuration of load-

ing bays and rear access nature of vehicles, is that each vehicle must be loaded and unloaded in a LIFO order, imposing constraints on the sequencing of pickup-delivery pairs. Our model of the PMDP is described in Chapter 4.

As an EngD thesis my research is strongly focussed on the business practicalities of the PMDP. Although shippers plan their consignments well in advance, often in regular cycles, the consignment information is typically shared with the haulage companies at the last minute. The resultant need to plan close to required delivery leads to inefficiencies in routing and scheduling. The currently used schedules are generated manually and offer significant scope for improvement using a computational approach. Another area that has received little research is investigating the cost savings attainable when many delivery companies work together, I present findings for a number of cooperation strategies, and quantify the savings possible. Though focussed on medium distance road vehicle haulage, with modifications my research would be applicable to many areas of transportation routing, such as ship, air or rail freight planning.

1.1 Thesis Outline

Chapter 2 introduces the travelling salesperson, vehicle routing and pickup and delivery problems (TSP, VRP, PDP) as related works that share many similarities with the PMDP, common extensions are presented that are of interest to Transfaction Ltd. Benchmark instances, later used to compare approaches, are introduced here. A selection of literature involving real-world case studies is also presented.

Chapter 3 concludes the literature review presenting a selection of solution approaches covering exact, heuristic, meta-heuristic and hyper-heuristic methods. Constructive heuristics along with local search improvement operators are also introduced.

Chapter 4 describes the model developed for the PMDP and highlights how this differs from the PDP. I introduce a number of new constraints for the PMDP along with a real-world cost based objective function.

Chapter 5 presents the new Variable Neighbourhood Descent with Memory (VNDM) hyper-heuristic and local search operators developed to solve instances of the PMDP. I present comparisons to other approaches introduced in the literature review and to the state-of-the-art on benchmark instances of the PDP. The results show the benefits of

using VNDM over other approaches. The following chapters investigate further aspects of real-world case studies in more detail.

Chapter 6 presents details of Transfaction Ltd.'s case study data sets and how we use this information to generate additional problem instances for our experiments. A number of hyper-heuristics are compared on the real-world data sets.

Chapter 7 analyses a system with less than full load consignments which may be combined to save on delivery costs, providing LIFO constraints are satisfied. VNDM is used to demonstrate the savings possible when hauliers cooperate, both naïvely and in a competitive environment.

Chapter 8 establishes the financial value of early information and the width of arrival time windows; showing that cheaper schedules are possible when there is more notice given and when consignment arrival time does not matter.

Finally Chapter 9 discusses all the academic and industrial contributions of my thesis and presents areas for future research.

2

Literature Review: Related Problems

Logistics problems have been studied for thousands of years and date back to the Late Bronze Age (Tepić et al., 2011), they are amongst the earliest problems investigated with the advent of modern computing. This review provides comments on work that is of particular interest to the business operations of the industrial sponsor of this research, Transfaction Ltd.

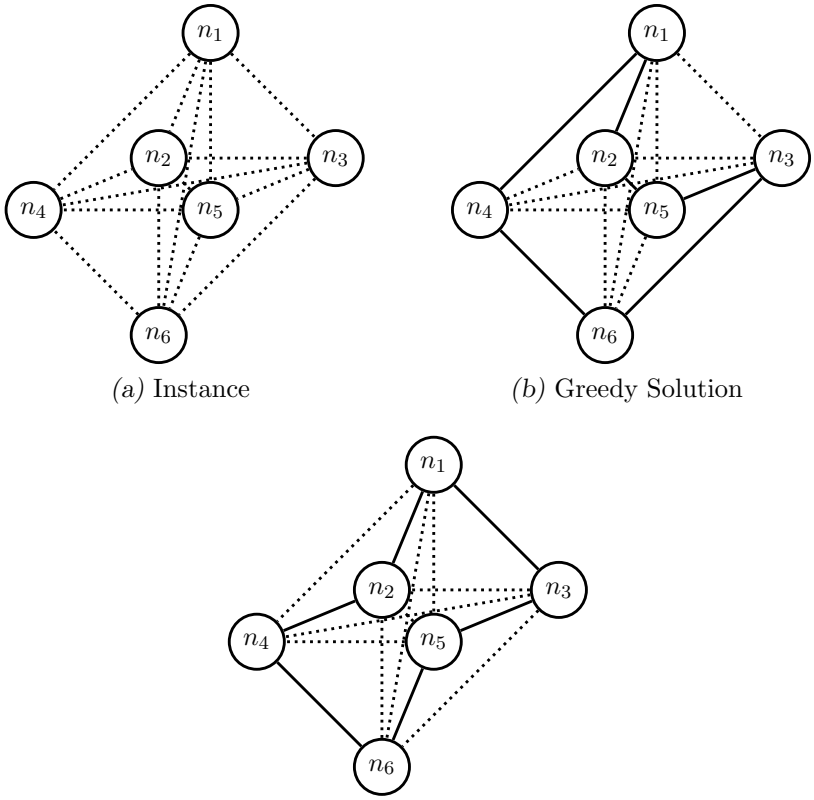
This chapter begins (Section 2.1.1) by introducing the Travelling Salesperson Problem (TSP). Section 2.1.2 describes the capacity limited, multiple route, extension of the TSP known as the Capacitated Vehicle Routing Problem (VRP). Section 2.1.3 presents a major variant of this problem known as the Pickup and Delivery Problem (PDP). Section 2.2 introduces some of the many extensions applicable to routing problems, necessary for modelling various real-world problems. These include constraints on the time of service, vehicle loading and driver working hours as well as dynamic scenarios where decisions have to be made in real time.

Section 2.4 presents the benchmark data sets used in Chapter 5 to compare the performance of our hyper-heuristic methods.

2.1 Routing Problems

This section introduces a variety of routing problems studied over the past few decades. Over this period many intricacies of real-world routing problems have been studied; the aspects covered in this section are those most related to my research.

2.1.1 Travelling Salesperson Problem (TSP)



(a) Instance *(b)* Greedy Solution

(c) Optimal Solution

Fig. 2.1: An example TSP and two potential solutions.

The Travelling Salesperson Problem (TSP), first modelled on computers by Dantzig and Ramser (1958), Clarke and Wright (1964) and Lin (1965), involves finding the shortest route a salesperson can take through a number of cities starting and ending at the same point, visiting each city once. More generally the TSP can be applied to any problem where the aim is to find the lowest cost Hamiltonian cycle of a weighted graph. The weight of each edge could be time, distance, cost etc. or a combination of factors. A Hamiltonian cycle consists of a connected subset of the edges from the initial graph forming a chain such that for each node exactly 2 associated edges are chosen. Karp

(1972) shows that finding the minimum cost Hamiltonian cycle is NP-Hard. Exact and approximate solution generation techniques are discussed in Chapter 3. Figure 2.1 is an example of a symmetric TSP with 6 nodes where the cost between nodes is Euclidean distance. Two potential solutions are presented, in this trivially small example, it is clear to see that a greedy heuristic (see Section 3.2.1) generates a non-optimal solution.

2.1.2 Vehicle Routing Problem (VRP)

The VRP, described by Dantzig and Ramser (1958) as the Truck Dispatching Problem, and later by Laporte and Osman (1995) and Fisher (1995), is an important extension to the TSP. In this model, vehicles begin and end their tours at a depot location and have a fixed maximum tour length or equivalent constraint (driving time). Vehicles must return to their depot before the constraint is violated. Since VRPs are analogous to the metric TSP (Hosny, 2010), they are all NP-hard problems as defined by Lenstra and Kan (1981).

In the capacitated VRP (CVRP), each node represents a customer request, which may have an associated demand; examples of constraints include maximum loading weight or volume. In CVRP's, once a capacity limit has been reached the vehicle must return to the depot before visiting any further customers. Alternatively, multiple vehicles may be employed to fulfil multiple, pre-scheduled, routes simultaneously. It is always the case that the total demand of the customers exceeds the capacity of a single truck, otherwise the problem becomes that of the TSP (Clarke and Wright, 1964). The objective in a VRP is usually to minimise distance though minimising the number of vehicles used or time taken given a fixed number of vehicles may also be considered.

2.1.3 Pickup and Delivery Problem (PDP)

The pickup and delivery problem (PDP) is the closest problem to the pickup and multiple delivery problem (PMDP), introduced in Chapter 4. Early work on the subject includes the single vehicle dial-a-ride problem (DARP) of Desrosiers et al. (1986) and PDP with time windows of Sexton and Choi (1986). DARPs involve a set of individuals with current and desired locations. A vehicle with a fixed capacity n is used to pickup and drop-off individuals and may make several collections before a delivery. This is

often referred to as part or partial loading, meaning that multiple consignments may be loaded onto any given vehicle concurrently. DARPs tend to be of small size and can often be solved exactly in reasonable time (Beck et al., 2003). The PDP is also referred to as either the travelling salesman problem with pickup and delivery (TSPPD) (Carrabs et al., 2007; Li et al., 2011) when dealing with a single vehicle or the vehicle routing problem with pickup and delivery (VRPPD) (Berbeglia et al., 2010). A PDP comprises fulfilling a number of *consignments*. Each consignment has a pickup location, where a vehicle is loaded, and a delivery location, where a vehicle is unloaded. This differs from the VRP where a vehicle is loaded at a single warehouse and all other locations represent unloading locations. In local freight operations, less-than-truckload (LTL) consignments may be combined to reduce delivery costs.

For clarity, we represent a vehicle's route as containing either consignment pairs or individual pickups and deliveries. The notation summarised in Figure 2.2 is used for all our illustrations of routes and operators on routes.

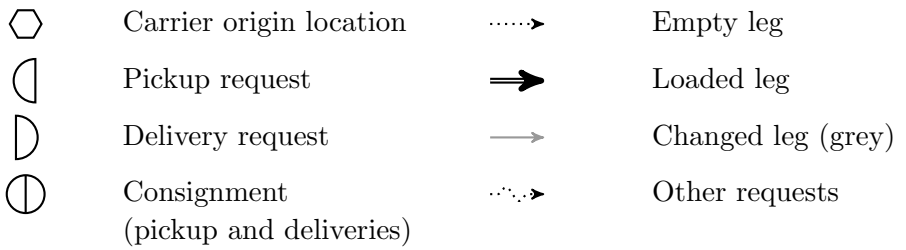
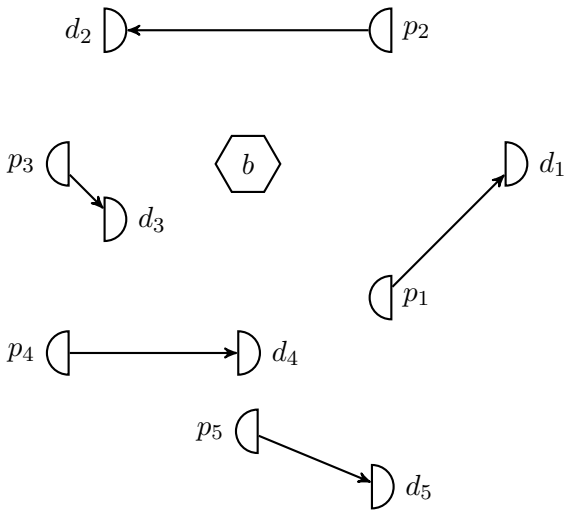


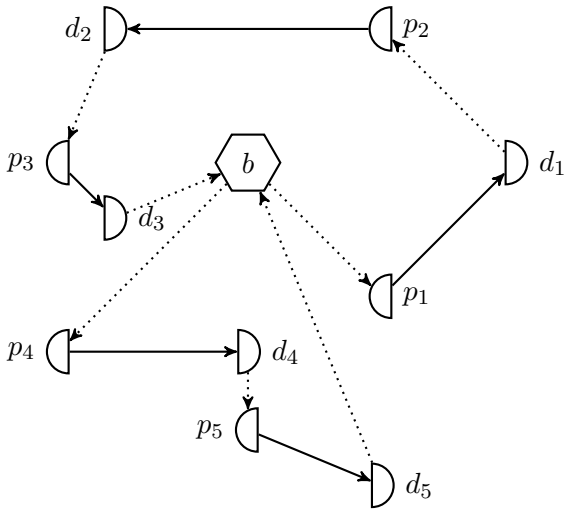
Fig. 2.2: Key to operator figures.

Abstractly, the problem can be viewed as in Figure 2.3. A delivery vehicle must follow a route that starts and ends at its base location b. Each consignment i is represented by a pair of locations, the pickup point p_i and the delivery point d_i. To fulfil a consignment, a truck must visit the pickup point before the corresponding delivery point. In Li and Lim (2003) the objectives are first to minimise the number of vehicles and second to minimise distance. Depending on the application, minimising the cost or delay of servicing all consignments are other common objectives.

Desaulniers et al. (2002) present a widely accepted mathematical formulation for the generic PDP, which they refer to as the *vehicle routing problem with pickup and delivery and time windows*. However, time windows are not specific to the PDP and may be



(a) Instance



(b) Solution

Fig. 2.3: Example pickup and delivery problem with two vehicles.

an extension to any routing problem. These and other extensions are discussed in the following section. The Desaulniers et al. (2002) model has inspired the constraints for my PMDP model (Chapter 4, page 79).

Research on PDPs usually concentrates on static models of small scale problems such as servicing taxi requests, ride sharing schemes or DARPs (Toth and Vigo, 1997; Beck et al., 2003). Cordeau et al. (2007) refer to the PDP as Transportation on Demand studying DARPs for the elderly and the disabled, urban courier services and emergency vehicle services amongst others. Delay minimisation is especially common in DARPs (Parragh et al., 2009). Berbeglia et al. (2007) summarise a number of routing problems classifying the PDP as a one-to-one problem and the VRP as one-to-many-to-one. In comparison, our PMDP (Chapter 4) is a one-to-many problem which Berbeglia et al. (2007) identify as requiring further research.

2.2 Real-world Problems

Real world problems are characterised by having many more constraints or of being much larger in size than traditional benchmark problems. Early examples include Bodin et al. (1983) who deal with crew allocation on top of a vehicle routing and scheduling problem. More recently Erera et al. (2008) investigate driver management schemes, presenting a effective greedy heuristic to match drivers to loads, given a large number of real-world constraints.

Horn (2002) models a real-world, large scale taxi routing problem within a 24h period, in the Gold Coast area of Australia. Horn takes into account predicted future patterns of demand and contingencies including breakdowns, trip cancellations and stochastic travel times. However as a taxi scheduling problem within a relatively small area the demands are different to those of our PMDP in a number of important ways. Firstly waiting time for taxi requests is a major concern, so there are likely to be more assets in use than a similar truck routing problem. Secondly, there is no ride sharing so saving through cooperation would not be as beneficial.

Although Horn presents a real world problem that has many similarities with our own, the constraints that Horn uses do not map well to our problem and the approach developed is compared only to naïve approaches.

2.2.1 Multiple Vehicles and Depots

VRPs are multiple route by default. As stated in Section 2.1.2, many real-world problems utilise multiple vehicles to service these routes. In the benchmark problems for the PDP of Li and Lim (2003), minimising the number of vehicles used is the primary objective.

Cordeau et al. (2007) consider single vehicle DARPs and Berbeglia et al. (2007) present the multiple vehicle extension to the DARP. Since the assignment of vehicles to customers must be specified along with the route for each vehicle, the solution set becomes much larger, making optimal solutions harder to find.

Other real-world scenarios involve multiple depot locations. Vehicles in the system may be located at different base depots. Any solution must specify the depot from which a customer's consignment is to be serviced, as well as its location in a route. Additional parameters such as the number of vehicles available at each depot as well as depot specific capacity constraints may need to be considered. Many large logistics operations in the UK have multiple depots and the PMDP model presented in Chapter 4 supports this.

2.2.2 Heterogeneous Fleets

Different vehicles may have different capacities or may be equipped to deal with specific cargo (e.g. refrigerated trucks). In some cases there can be no overlap, such as vehicles equipped for liquid transportation (Desrosiers et al., 1995), and in these cases it is best to treat each set of vehicles as a different problem. A heterogeneous fleet exists where vehicles are interchangeable, e.g. simply of different capacity, a choice must be made regarding the merits of different available vehicles. Paraskevopoulos et al. (2008); Koç et al. (2014) and Savelsbergh and Sol (1998) model heterogeneous fleets in VRPs while Desaulniers et al. (2002) and Xu et al. (2003) look at the same extension to the PDP. Notable surveys that include references to heterogeneous fleets are by Desrosiers et al. (1995); Cordeau et al. (2002); Laporte (2009) and Pillac et al. (2013).

2.2.3 Last-in, First-out (LIFO) Loading

In some real world scenarios it is impractical to unload anything but the most recently loaded items in a vehicle, for instance in rear access HGVs. In these cases a last-in, first-out (LIFO) ordering constraint is placed on deliveries. Of the routing problems presented, LIFO loading only applies to the PDP. LIFO ordering has only recently received attention with the works of Carrabs et al. (2007), Li et al. (2011), Cheang et al. (2012), Cherkesly et al. (2015), Crainic et al. (2015) and Benavent et al. (2015). A variety of techniques have been used in each case, these are explored in more detail in Chapter 3. Due to the vehicles used, Transfaction Ltd.'s problem and the PMDP model have LIFO constraints.

2.2.4 Green Logistics

In green planning, Demir et al. (2013) survey in detail how vehicle routing can be designed to minimise CO_2 emissions. From vehicle load, speed, congestions on route, road gradient and efficient routing, they estimate potential savings of 10%. Sbihi and Eglese (2010) also look at issues of green logistics but from the viewpoint of routing vehicles used in green initiatives such as recycling; this research is at a tangent to our own, having problem specific constraints to consider. Green logistics represent a potential area for future research on the PMDP.

2.2.5 Time Windows Constraints

VRPs with time windows are traditionally referred to as VRPTWs. In PDPs time windows may exist at both pickup and delivery locations. Time windows are such a common feature in PDPs that most authors assume a PDPTW when discussing the PDP. I will do the same for brevity and consistency.

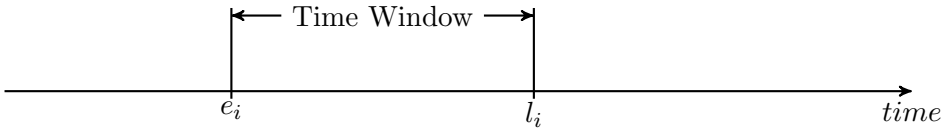


Fig. 2.4: Request time window - Earliest and latest service start times, e_i and l_i respectively are shown for a request i. Arrival at any time between these points is equally good.

Figure 2.4 shows the time window for a request i (Toth and Vigo, 2002). For the

general VRP, we treat the time window constraint as a period of time defined by an earliest and latest time during which a customer should be serviced (For time windows in Transfaction Ltd.'s problem see Section 4.2.1, page 74). Arrival before a time window is typically allowed, with the truck having to wait. Late arrival may be either a hard or soft constraint, respectively not allowed or allowed with a fixed or delay dependent variable penalty (Desrosiers et al., 1995; Taillard et al., 1997; Krumke et al., 2002).

Desrosiers et al. (1995) present a thorough summary of the history of temporal information in transportation logistics problems and should be consulted for further information.

2.2.6 Dynamic Routing Problems

All the problems introduced thus far are static, with perfect information; in which all requests are known in advance of any vehicles commencing its journey (Berbeglia et al., 2007). In real-world systems, this is rarely the case. Schedules may have to be altered "on-the-fly" to accommodate new customer's consignments arriving at any time, in any order. These are known as real time (Zhu and Ong, 2000; Krumke et al., 2002; Nahum, 2013) or dynamic problems (Larsen, 2001; Bent and Van Hentenryck, 2004; Cowling et al., 2004; Gendreau et al., 2006; Berbeglia et al., 2010; Ouelhadj and Petrovic, 2009; Gschwind and Irnich, 2012; Pillac et al., 2012; Albareda-Sambola et al., 2014). To solve a dynamic problem, a solution must change over the course of a real or simulated time period. The current location of each vehicle must be kept up to date in this scheme and valid insertion points are only those beyond a vehicle's current location. The dynamic model for the PMDP is presented in Section 6.3, page 112.

A dynamic problem can be split into a number of static routes, known before the simulation of time begins, and dynamic consignments which arrive sporadically whilst the currently planned consignments are serviced. The resulting schedule must evolve to cope with the changing demands placed upon it. Bianchi (2000) compares a number of strategies for the dynamic VRPTW and PDP.

Bouros et al. (2011) states that a conventional way to solve these dynamic problems is by using a two phase local search algorithm, based on the quality of the solution from the point of view of both the shippers (customers) and the carriers (delivery companies).

The research presents a novel solution using a graph based formulation of the problem where each request is treated independently. Bouros et al. (2011) show that Bellman-Ford (Bellman, 1958; Ford Jr., 1956) or Dijkstra-like (Dijkstra, 1959) algorithms cannot be applied to this problem as sub-path optimality (critical to the correctness of those algorithms) does not hold when there are two independent cost functions. A branch and bound approximation algorithm is presented that is shown to find solutions significantly faster than a conventional two-phase search.

Hentenryck et al. (2009) present a strategy whereby likely consignments are anticipated with a probability based on how often these consignments have occurred in the past. Schedules can then be designed to easily accommodate anticipated consignments.

The dynamic vehicle routing problem has attracted a considerable amount of attention in recent years. The survey by Pillac et al. (2013) presents a general description, along with the notion of degree of dynamism where different systems may be classified as more or less dynamic based on two factors, the frequency of changes and the urgency of requests. (Larsen, 2001) introduces a measure of effective degree of dynamism for problems with variable time windows, we investigate the effect of this in Section 8.6.1, page 149. The research presented in this thesis focusses on long distance PMDP, as prevalent in primary goods distribution, e.g. from a farm or supplier, to a number of supermarkets. Though not in the area of vehicle routing, Ouelhadj and Petrovic (2009) survey scheduling in manufacturing systems which share a number of similarities with the VRP, namely complex constraints and a variety of unexpected disruptions. The Ouelhadj and Petrovic (2009) survey has a strong emphasis on scheduling and prediction of real time events, whilst the Pillac et al. (2013) survey is primarily a thorough review of dynamic VRPs, with real-time events and stochasticity representing small asides.

2.2.7 Foresight Policy

Problems with clearly repeating similar cycles of activity, such as the internet grocery shopping problem, are described by Yang et al. (2014) as "e-fulfilment problems". Customer deliveries can be priced based on either their impact into the current schedule "*hindsight*" or based on their projected impact in the eventual schedule "*foresight*".

The projected cost of servicing a consignment based on a foresight model (Yang et al., 2014) is:

$$C_j^s = w_j C_H^s + (1 - w_j) C_F^s, \tag{2.1}$$

where C_j^s is the expected cost of insertion, C_H^s is the cost of insertion into the actual schedule and C_F^s is the cost of insertion into a previous week's schedule. Since foresight methods are more accurate when fewer consignments are known and hindsight methods are more accurate towards the end of a planning period, a weighting w_j is applied to the hindsight cost and $1 - w_j$ is applied to the foresight cost. w_j is defined as j/J where j is the number of the current consignment and J is the total number of consignments expected in the planning period.

Repeating cycles of delivery activity occur in many areas of vehicle routing, not just e-fulfilment. A foresight model using previous scheduling information to inform future delivery costs could be applied to Transfaction Ltd.'s problem.

2.2.8 Stochastic Travel Times

In many logistics problems, travel time is not a fixed quantity and may depend on weather, traffic or other considerations. Problems which attempt to deal with these problems often use "stochastic" travel time, where journey times are altered by a random (stochastic) quantity.

Figliozzi (2010) looks into the impact of congestion on routes where journey times are modelled as a normal distribution. Congestion changes both the mean and standard deviation of the journey time distribution, as well as adding further costs.

Haghani and Jung (2005) discuss time-dependent travel times and presents a genetic algorithm that can solve trivially small problems with results within 8% of the exact solution. They also show that, with uncertain travel time, the dynamic model vastly outperforms the static model. Haghani and Jung (2005) consider both stepwise and continuous functions for travel time in relation to time of day.

Lorini et al. (2011) model a situation where customer's consignments can be altered "*online*" (while on route to a destination). Their solutions are generated using a greedy insertion heuristic followed by a descent search using CROSS moves (Taillard et al., 1997) before a final local descent search using a relocate operator on each route (Taillard

et al. 1997, see Section 3.2.2). Their model includes varying weights for edges at different times of day, using a stepwise function.

Li et al. (2009) present a Lagrangian relaxation for the real-time VRP with time windows then use a "*dynamic programming heuristic*" to find feasible solutions in problems with vehicle breakdowns. Li et al. (2009) test their approach on modified Solomon (1987) benchmark instances of small size. Their insertion heuristic has a number of clever features, such as reducing the size of the modifiable solution by removing sections of routes that are near optimal in terms of distances and time window satisfaction. They then use a greedy heuristic on locations that produce feasible schedules. They conclude that although this approach produces schedules of slightly higher operating cost the cancellation and total costs are reduced.

2.2.9 Competition

Figliozzi et al. (2003) suggest that the widespread adoption of the internet has reduced the costs of working with different companies, opening up the possibility of using online auction houses to determine logistics suppliers for individual shipments rather than relying on a single company for all shipments. We investigate the potential savings when delivery companies work together in Chapter 7.

Figliozzi et al. (2003) developed a dynamic model of a logistics trading market supporting multiple shippers and carriers. Each of these agents has a set of beliefs about the system that gradually changes over time. Shippers initiate auctions by presenting consignments to be fulfilled. Figliozzi et al. (2003) also investigate the effect of changing the number of carriers (keeping the number of vehicles constant) and the effect of modifying the arrival rate of shipments on the quality of the solution obtained. Subsequently, Figliozzi et al. (2007) investigate the effect of hidden rewards for consignments which only become known after the auction phase. More recently, Robu et al. (2011) developed a similar, multi-agent auction platform as part of a Dutch collaboration to improve transportation logistics.

Zhu (2004) analyses electronic markets that can be used by haulage companies to bid on delivery contracts. The analysis takes a game theoretic point of view and suggests that transparency to see other companies' bids would weaken the appeal of joining such

a system, because companies may intentionally bid lower than their competitors even if it is not directly in their best interests to starve the competition. Their analysis does not extend to the quality of schedules generated as a result of the auction with respect to number of vehicles used, distance travelled or financial implications.

Moon et al. (2012) investigate a problem where additional vehicles may be outsourced with a different price structure. In their work on heterogeneous vehicles, Moon et al. (2012) also consider driver overtime as an additional feature affecting the cost of delivery schedules.

2.2.10 Business Design Issues

Most research on VRP and PDP is concerned with finding better solutions to a set of problem specific, synthetic or benchmark problems. Although incredibly useful, research has not considered the business costs and savings that pertain to the scheduling strategy analysed. Problems that consider customer requests to have time windows either treat them as hard constraints (Mester and Bräysy, 2005) or with a penalty function (Taillard et al., 1997) but do not investigate how efficiently the requests could have been served if the time windows were looser or removed entirely.

McLeod et al. (2012) show that a dynamic service person routing and scheduling approach can work almost as well as a static solution and could enable companies to offer same day rather than next day service. Their approach can generate schedules fast enough that vehicles are not left without instruction and for a similar service cost.

Mitrović-Minić et al. (2004) propose a double-horizon approach for dynamic problems where requests that are not imminent are scheduled separately, using a different fitness function. They note that although promising, "percentage improvements go down as instances become larger". Since real-world instances of the PMDP are very large, we felt that the additional effort required for double-horizon scheduling was not worthwhile in the development of our VNDM approach.

Yang et al. (2014) find that scheduling in anticipation of repeated events in a VRPTW can produce savings of up to 6%. Similarly, Thomas and White III (2004) find anticipation can greatly help in LTL PDP especially when customer requests arrive late in a vehicle's route. I expect anticipation would be beneficial to the PMDP

developed in this thesis but remains an area for future research.

In a handicapped person's transportation problem, Toth and Vigo (1997) show that a heuristic approach is capable of solving real world instances better than existing manual schedulers. Solutions assessed by their fitness function (fixed and routing costs plus user inconvenience penalties) showed a 36% improvement whilst adhering to problem specific constraints. The heuristic solution was able to more efficiently utilise assets, "doubling the average number of trips served per route". Similarly, Erera et al. (2008) show that a heuristic approach to driver management can reduce the number of drivers required by up to 10% in comparison to manual driver management strategies. Dorer and Calisti (2005) uses an Agent-based approach to solve a real-world dynamic PDP with partial loading. They find that an agent-based approach can produce overall cost savings of around 12% when compared to manually planned dispatching. Given these precedents in other areas of routing and scheduling we felt confident that we could outperform manual scheduling on the PMDP.

2.3 Fitness Functions

All routing problems have an associated fitness function used to evaluate how good any given solution is, relative to an other. Theoretical studies starting with the TSP focus on distance minimisation. However, the fitness function varies greatly on an individual problem basis. VRP and PDP problems often have a multi-objective fitness, first to minimise the number of vehicles used and second to minimise total distance travelled. Problems drawn from real world situations often use a fitness function more aligned with business needs, for example taxi and DARPs may use a fitness function that seeks to minimise customer waiting time (Horn, 2002). For haulage problems it is often most appropriate to minimise the cost of delivering all consignments or to maximise profit. If it is not possible to service all consignments, part of the problem is to decide which consignments to service.

2.4 Benchmark Instances

Christofides (1979), Solomon (1987) and Golden and Assad (1988) have all produced benchmark instances for the VRP. The OR library links to the VRP web where these

benchmark problems are hosted (Diaz, 2006). Li et al. (2005) introduced a set of benchmarks for large-scale VRPs, more appropriate for real-world scenarios and modern solution approaches. Laporte (2009) includes tables by Ropke (2005) identifying the 10 best heuristics for the Christofides (1979) and Golden and Assad (1988) test sets. The smaller test sets of Christofides (1979) ($51 \leq n \leq 199$) are solved by memetic algorithms (Nagata and Bräysy, 2009), tabu search with adaptive memory (Rochat and Taillard, 1995) and local search limitation strategies (Nagata and Bräysy, 2008). Genetic and guided evolution strategies presented by Mester and Bräysy (2005, 2007) attain results close to these (0.03% above the best solution). For the more complex scenarios ($200 \leq n \leq 480$) of Golden and Assad (1988), local search limitation strategies (Nagata and Bräysy, 2008) produce results within 0.01% of the optimal solution. Other techniques applied to the larger test sets are less competitive than on the small test sets but all still produce results within 1% worse than optimal. A derivative of tabu search by Tarantilis (2005) achieves only 0.76% worse than optimal. These meta-heuristic approaches are introduced in more detail in Section 3.2.4. Li et al. (2011) present benchmarks for the single vehicle TSPPD but do not compare their results against others. Lim et al. (2016) present a new set of benchmarks for the PDP with manpower scheduling for a non-emergency ambulance service, comparisons are made against a manual schedule.

The most widely studied benchmark instances of the PDP (from Li and Lim (2003)) are used in this thesis to compare our methods with the state-of-the-art. These benchmarks are more tightly constrained than either of our retailer case studies, introduced in Chapter 6, and use a different objective function, but are otherwise similar enough for comparisons. The benchmarks have been investigated by Bent and Van Hentenryck (2003); Blocho (2015); Hasle et al. (2007); Hosny (2010); Koning (2011); Li and Lim (2003); Ropke and Pisinger (2005), and corporate researchers: Quintiq (2015); Tetra-Soft (2003). The best published results are kept up to date online by Sintef (2008).

2.5 Discussion and Summary

A common theme is that real-world routing problems are rarely the same, though similar techniques may be used to solve them (see Chapter 3). Berbeglia et al. (2007)

identify one-to-many vehicle routing problems such as the PMDP as requiring further research and no work on similar problems can be found in literature.

Much research has gone into solving the PDP, specifically the benchmarks of Li and Lim (2003), which may be seen as a relaxed variant of the PMDP where each pickup has only a single delivery and there is no LIFO constraint. The state-of-the-art for PDP is ALNS of Ropke and Pisinger (2005) which set many of the best known solutions to these problems. Fitness functions for benchmarks traditionally focus on minimising number of routes then distance whilst real-world problems are often more concerned with cost or timeliness. Though a practical solver for the PMDP is the goal of my thesis, I compare against these PDP benchmarks to ensure a good standard of solution quality in a less constrained problem.

Another more closely related problem is the PDPL, however, solvers for this problem use block or tree structures which are not easily expandable to a multiple delivery problem and there are fewer existing approaches to compare against, thus the PDPL solvers have not been used as a basis for solutions for the PMDP. This chapter presented an overview of vehicle routing and scheduling problems and models that are of particular relevance to this thesis. The differences between PDP, VRP and TSP are explained and extensions are introduced that more accurately model real-world problems. All of these routing problems are NP-hard as they are at most reducible to TSP. The following chapter looks at solution methods that have been applied to these problems.

3

Literature Review: Solution Approaches

Chapter 2 introduced the TSP, VRP and PDP along with some of many extensions. This section presents a brief history of solution approaches to VRPs. Section 3.1 covers exact solution approaches, whilst Section 3.2 introduces non-exact methods. Sections 3.2.1 and 3.2.2 present common constructive and modification methods used in non-exact approaches. In Section 3.2.3 an overview of a number of important heuristics for the routing problems is given. Section 3.2.4 looks into more intelligent meta-heuristics that have been successfully applied to these problems. Section 3.2.6 presents hybrid methods that combine multiple meta-heuristic strategies into problem specific solutions. Section 3.2.5 presents a number of hyper-heuristics, the focus of this thesis, which attempt to build hybrid meta-heuristics automatically and therefore work across different problem domains.

3.1 Exact Methods

Exact methods solve complex problems mathematically, producing provably optimal solutions. For the VRP, a good summary of exact methods is presented by Laporte (1992), notably including the works of Malandraki and Daskin (1992) and Desrochers et al. (1992). For VRP and PDP, exact methods are most useful for small size problems as the computation time required to solve large instances grows exponentially, often

becoming infeasible, due to the NP-hard nature of these problems.

Exact solutions to static PDPs favour branch-and-cut-and-price algorithms using column generation techniques. Ropke and Cordeau (2009) provide a good description of using branch-and-cut for the pickup and delivery problem with time windows. A tree of potential solutions is generated, of which not all are feasible solutions, while searching the tree, branches can be cut either by proving infeasible or by guaranteeing a worse solution than the best already found. Dumas et al. (1991) use this approach to solve a multi-depot PDP for problems with up to 55 requests. No indication is given of whether their approach scales to larger problem sizes. Baldacci et al. (2010) has produced an exact solution framework for a broad class of VRPs and more recently Gschwind and Irnich (2012) presented a branch-and-cut-and-price exact solution approach to the dynamic time window DARP of up to 96 requests.

Xu et al. (2003) solve a PDP based on real-world logistics with multiple carriers, vehicle types and LIFO constraints of up to 500 requests using a column generation formulation containing an exponential number of columns. However, in order to produce solutions in acceptable time, they introduce heuristics into the column generation subproblem, and cannot guarantee optimal solutions. In general, exact methods do not scale well, so non-exact approaches that can quickly find near-optimal solutions, have become popular for large-scale, real-world problems. These are discussed in the following section.

3.2 Non-Exact Methods

Most real-world problems are too large or complex to be solved quickly using exact methods. In these cases non-exact methods are preferred, enabling approximate solutions to be generated quickly. Some non-exact methods take exact methods and make simplifying assumptions or approximations, for example, the large neighbourhood search (LNS see Section 3.2.5.4) of Shaw (1998) uses truncated branch-and-bound to avoid searching entire solution trees.

Heuristic methods are widely used alternative approaches that build up and modify a solution using operators. These can be broadly divided into construction and modification operators. A heuristic defines a set of rules specifying the choice and usage

of operators. A simple *greedy* heuristic for the TSP would specify an initial solution, a pair of nodes in the graph, and one operator. The operator would choose a node whose edge to the current chain is shortest and add this node/edge to the solution. This operator would be used repeatedly until all nodes were in the solution and finally join the first node to the last to complete the TSP cycle.

To improve upon this result, modification operators can be utilised, often for a user-specified amount of time. This process is referred to as heuristic search. Running a heuristic search for a longer period of time should result in a better solution though in practice they offer diminishing returns as the approximate solution approaches the optimal one. Heuristic search may also suffer from becoming stuck in poor quality local optima and being unable to find solutions close to the global optimum. In practice, for big problems, we probably approach the optimum *very* rarely so a soon-enough-good-enough solution is sufficient.

3.2.1 Construction Operators

Construction operators add customers to an existing (initially empty) partial solution. This section surveys a number of construction operators, common amongst heuristic methods.

3.2.1.1 Random Insertion

Random insertion (Mester and Bräysy, 2005) is one of the simplest construction heuristics and simply adds a customer in a random, feasible, position in the existing partial solution. It is very fast as no features are used to decide where to place the customer and no consideration is given to the quality of the resulting partial solution.

3.2.1.2 Greedy Insertion

Greedy insertion simply finds the best position for a new customer given the customers that have already been added to the solution. It does not consider the future implications of any decision so is fast but solutions generated in this way are often far from optimal. Yang et al. (2014) present a method for greedily inserting a customer's consignment into an existing schedule as part of a choice based demand management

problem in e-fulfilment (e.g. online shopping) which shares many similar constraints with our problem, introduced in Chapter 4.

3.2.1.3 Clarke and Wright Savings (CWS)

The savings algorithm by Clarke and Wright (1964) (CWS) is one of the first computational solutions to the VRP introduced in Section 2.1.2, building on the work of Dantzig and Ramser (1958). It is a relatively simple procedure as shown in Algorithm 3.2.1.

Algorithm 3.2.1 Clarke and Wright Savings Algorithm (Clarke and Wright, 1964).

Precondition: assign a unique route for each customer
  repeat
    for all unique customer pairs
      calculate 4 savings possible by combining these routes
    update the route giving the largest saving
  until no savings possible

Taking two routes as defined in Equation 3.1, new routes are made by removing one link from each existing route, connecting together c_i and c_j, and re-connecting the severed routes to base. The resultant routes are shown in Equations 3.2 though 3.5. Note that routes in Equations 3.2 and 3.5 reverse the direction of some parts of the existing routes.

$$(b, \ldots, c_{i-1}, c_i, c_{i+1}, \ldots, b)\,(b, \ldots, c_{j-1}, c_j, c_{j+1}, \ldots, b) \tag{3.1}$$

$$(b, \ldots, c_{i-1}, c_i, c_j, c_{j-1}, b)\,(b, c_{i+1}, \ldots, b)\,(b, \ldots, c_{j+1}, b) \tag{3.2}$$

$$(b, \ldots, c_{i-1}, c_i, c_j, c_{j+1}, b)\,(b, c_{i+1}, \ldots, b)\,(b, c_{j-1}, \ldots, b) \tag{3.3}$$

$$(b, \ldots, c_{i-1}, b)\,(b, \ldots, c_{j-1}, c_j, c_i, c_{i+1}, \ldots, b)\,(b, c_{j+1}, \ldots, b) \tag{3.4}$$

$$(b, \ldots, c_{i-1}, b)\,(b, \ldots, c_{j-1}, b)\,(b, \ldots, c_{j+1}, c_j, c_i, c_{i+1}, \ldots, b) \tag{3.5}$$

The savings are calculated as the sum of the edges that were removed minus the sum of edges added for each new route. CWS is used by Takes and Kosters (2010) and

Benavent et al. (2015) to solve subproblems of PDPs.

3.2.1.4 Generalized Insertion (GENI)

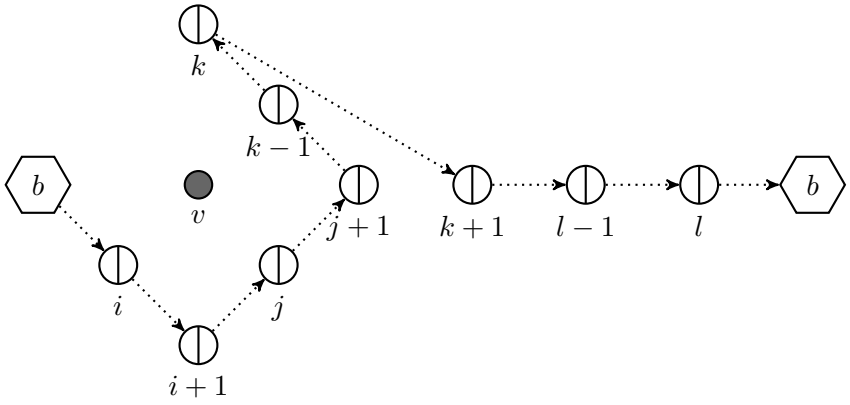
Gendreau et al. (1992) introduces Generalized Insertion (GENI), an insertion method which considers a customer's p closest neighbours and considers all routes which locate the new customer such that it connects two of its p closest customers without regard for their position in the target route. For each pair of nearest neighbours (i, j), and for each third distinct node k in the route, there are two potential new routes, shown as Type 1 and Type 2 in Figure 3.1. These two routes represent different ways to reconnect the severed initial route. The lowest insertion cost route is chosen for insertion. The original connections of the target route are maintained as far as possible (but may be reversed) given the new customer's position in the route.

3.2.1.5 Least Regret Insertion

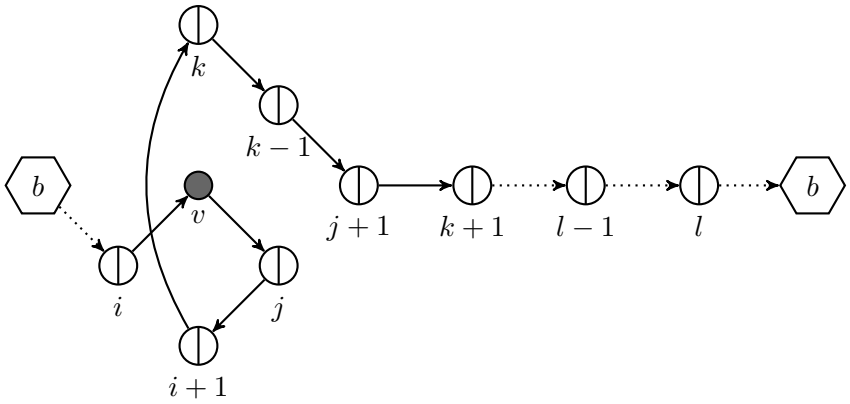
Ropke and Pisinger (2005) present a least regret construction operator that considers the k-best locations for inserting all unscheduled customers into the current solution. The difference between the best and worst insertion location for each customer is its regret and the objective is to minimise the total accumulated regret across all customers. The customer with the largest potential regret is inserted first so that its regret is not manifested. The process repeats until all customers have been added to the solution.

3.2.2 Modification, Local Search Operators

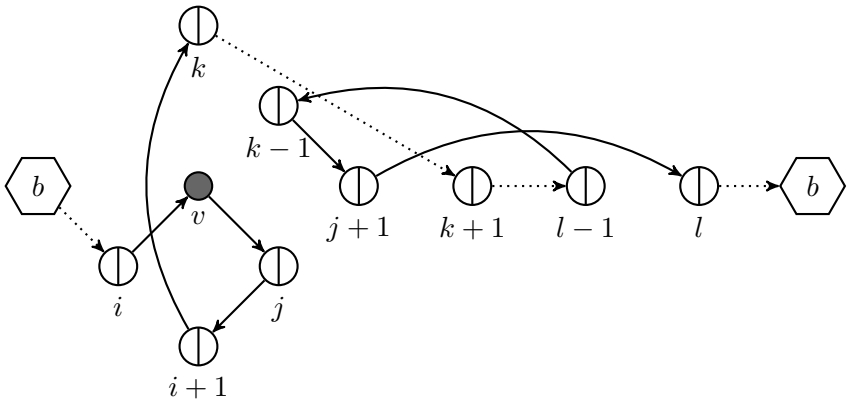
Once a solution has been generated using construction operators, modification operators can be used to improve the solution in terms of the fitness function (Section 2.3). In the literature, these are often referred to as Local Search Operators (LSOs) or neighbourhoods and serve as the LLHs used by the hyper-heuristics introduced in Section 3.2.5. Using an LSO on a solution leads to the creation of new solutions, similar to that given, where usually only a small number of customers will have changed position. Bräysy and Gendreau (2005a) presents a comprehensive analysis of LSOs that are summarised below along with others from Taillard et al. (1997); Bräysy (2003). Figures are presented where appropriate using the notation introduced in Figure 2.2. In all cases,



(a) Before



(b) Type 1



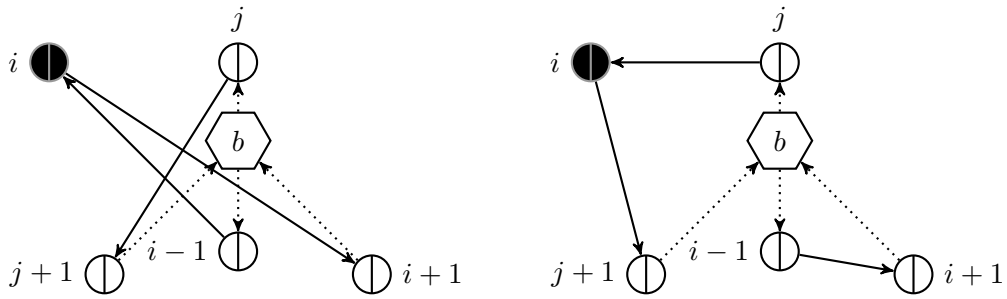
(c) Type 2

Fig. 3.1: GENI operator.

dotted edges are those which the LSO leaves unchanged, while solid edges are changed from one solution to the next. K-opt and its variants are presented in the Appendix, Section A.1.1.

3.2.2.1 Relocate

A simple strategy for local search across a solution with multiple routes is the Relocate operator, proposed by Savelsbergh (1992) (Figure 3.2), which takes a customer from one route and adds it to another route.



(a) Before *(b)* After

Fig. 3.2: Relocate operator

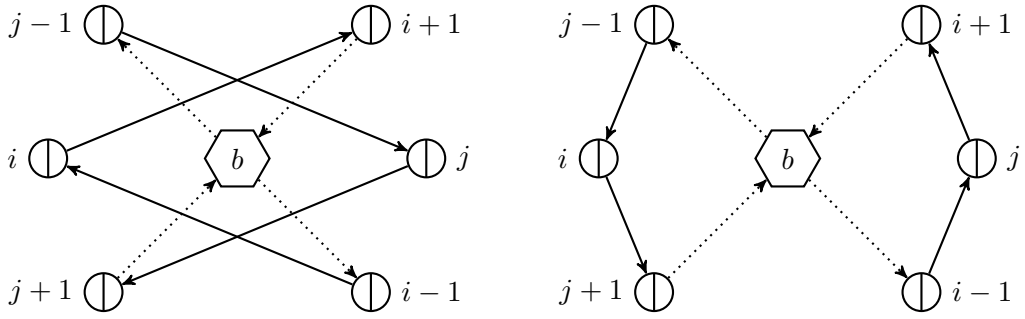
3.2.2.2 λ-interchange

The neighbourhood of a λ-interchange can be thought of as a being generated by a number of relocation operations. For a pair of routes a number less than λ is chosen; this many customers are chosen at random from route 1 and relocated into route 2. Concurrently a different number ($< \lambda$) of customers is chosen and relocated from route 2 to route 1, such that no customer is moved from route 1 to route 2 and back again or vice versa.

Osman (1993) uses λ-interchange to generate routes, and proposes a way to speed up the best admissible policy. The changes are stored in a 2 dimensional array of customers against routes: when a customer is moved from one route to another only 2 entries must be updated, the original route and the target route corresponding to that customer. All other results are the same so are not modified. Osman (1993) shows that this approach produces a 50% reduction in computing time.

3.2.2.3 Exchange

The exchange operator swaps a pair of customers in 2 routes. 4 edges are changed, as shown in Figure 3.3, but the resulting two routes have only one node different to their respective initial configurations.



(a) Before *(b)* After

Fig. 3.3: Exchange operator.

3.2.2.4 Chain-exchange

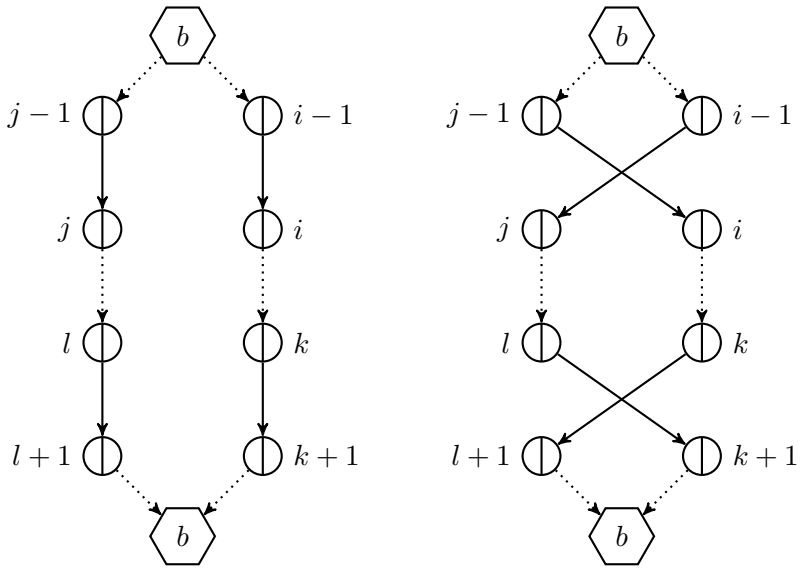
Fahrion and Wrede (1990) presents Chain-exchange where two arbitrary length chains of customers are selected from two routes and are swapped between routes, the insertion point is taken as the location which would minimise cost in both cases.

3.2.2.5 CROSS-exchange

CROSS-exchange, presented by Taillard et al. (1997), is an extension of the exchange operator where chains of customers are exchanged between routes rather than individuals. It is also very similar to Chain-exchange, except that no calculations regarding lowest insertion cost are made, the chains are simply swapped directly. The chains may be of varying length but the process continues as shown in Figure 3.3 where i and j now represent arbitrarily long chains of customers. Figure 3.4 shows a case where a chain of customers from $i...k$ is exchanged with the chain of customers $j...l$ from another route.

3.2.2.6 iCROSS-exchange

Bräysy (2003) presents an alternative local search method, similar to CROSS-exchange where the chains that are swapped from one route to another have their sequencing



(a) Before *(b)* After

Fig. 3.4: CROSS exchange operator.

reversed. As shown in Figure 3.5 the customer chains $i...k$ and $j...l$ are reversed to $k...i$ and $l...j$ respectively while simultaneously being interchanged between routes.

3.2.2.7 Insert Related Parallel

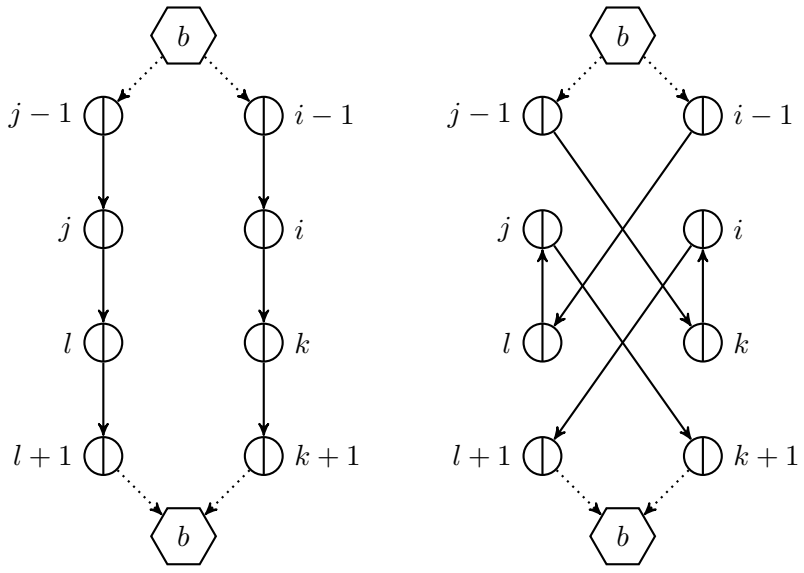
Bräysy (2003) presents Insert Related Parallel (IRP) in which a set of nodes is chosen from two routes such that the difference (distance, cost, etc.) between nodes is less than a given threshold. These nodes are then removed from their respective routes and reinserted into the gaps in both routes using a parallel lowest cost insertion heuristic which at each stage considers all nodes in all feasible locations. I-opt (Appendix, Section A.1.1) is used in the process to further optimise the newly created chains.

3.2.2.8 Cycle-exchange

Thompson and Psaraftis (1993) introduce cycle transfers for vehicle routing problems which involve moving a small number of customers between routes, potentially affecting a large number of routes in a solution.

3.2.2.9 Ejection Chains

Glover (1996) proposes and Bräysy (2002) and Sontrop et al. (2005) make use of the idea of ejection chains. Given an initial solution, a chain of consignments is moved



(a) Before *(b)* After

Fig. 3.5: iCROSS exchange operator.

from one route in a solution to another. If the target route now violates capacity or other constraints a chain of consignments is moved from this route to another, this process repeats, taking each route in sequence as the target route, until no constraints are violated or until the initial route becomes the target route. The process is similar to that used by cycle exchange but with connected chains of consignments and more emphasis on constraints.

3.2.2.10 Edge Assembly Crossover

Nagata (2007) extends his earlier work on Edge Assembly Crossover (EAX) for the TSP (Nagata and Kobayashi, 1997) to the CVRP. Rather than working on one or more routes, EAX works across 2 entire solutions. A crossover strategy is used whereby two new solutions are generated from a pair of existing solutions. 4 intermediate solutions are generated in this process whose edges can be partitioned into 4 sets: edges from parent A, edges from parent B, edges from both parents, new edges. Child solutions are built by alternating edges from A and B. Where this is not possible or where more than one choice is available, the shortest edge is chosen.

3.2.2.11 Discussion: Modification, LSOs

Though many of these operators come from VRP solvers they are commonly used in PDPs as the structure of the solutions are broadly similar. The reversal of large sections of routes as seen in Or-opt, I-opt and ICROSS is typically not useful for problems which have tight time window constraints as the resultant route will typically violate these if they were satisfied originally. The LSOs used in my research are introduced in Chapter 5, Section 5.2

3.2.3 Simple Heuristics

On their own, operators may not be particularly effective. Using purely constructive heuristics, such as repeatedly using the CWS operator (Section 3.2.1), may produce a solution to a problem quickly, but it is often far from optimal. LSOs require an initial solution built by construction operators, and if any one LSO was used on its own the solution would quickly become stuck in a locally optimum solution. One way of combining the usefulness of LSOs is to use heuristics. Heuristic methods define rules to combine the use of several different constructive operators and LSOs, offering an efficient way of finding good, approximate, solutions to a wide variety of problems. Summaries of heuristic solutions for the Capacitated VRP are given by Laporte (1992); Cordeau et al. (2002); Laporte (2009). This section provides a brief introduction to two important heuristic methods before summarising work in this area.

3.2.3.1 Sequential Insertion

In the following c_{ij} refers to the cost of travelling from node i to node j (node 0 is the depot). Two scores are associated with every insertion (Equations 3.6 and 3.7). Parameters λ and μ control the insertion preferences. $\lambda = 1$, $\mu = 0$ results in minimum extra distance. $\lambda = \mu = 0$ results in smallest distance between two neighbouring nodes. $\lambda > 1$, $\mu = \infty$ results in the node furthest from the depot being inserted first.

$$\alpha(i,k,j) = c_{ik} + c_{kj} - \lambda c_{ij}. \tag{3.6}$$

$$\beta(i,k,j) = \mu c_{0k} - \alpha(i,k,j). \tag{3.7}$$

A simple 3-step sequential insertion approach, reproduced in Gendreau et al. (2002), is given below:

1. Initialize a route with a tour including one unrouted customer, k.
2. For each customer calculate the insertion cost at all feasible locations (between existing routed customers).
    - If no insertions are feasible, go to step 1.
    - Else choose the insertion which maximises Equation 3.7 (If there is a tie, choose the node which minimises Equation 3.6) and update the route.
3. Optimize the modified route using 3-opt, then go to step 2.

Christofides (1979) adds a second, parallel construction phase to the sequential insertion approach, in which, any unrouted nodes are inserted into a minimal additional cost route, and 3-opt is used to minimise cost.

3.2.3.2 Monte Carlo Techniques

Takes and Kosters (2010) present a study into the under-researched area of applying Monte-Carlo techniques to VRPs. Taking the ordered saving list from the first stage of CWS, Takes and Kosters (2010) apply a Monte-Carlo selection operation to decide if each of the savings should be made. They note that their implementation (BinaryMCS-CWS) achieves results within 3% of best known solutions on test sets from Christofides (1979) and Augerat et al. (1998). It also outperforms ALGACEA-2 (Faulin and Juan, 2007) which follows a similar approach but uses purely random sampling, ignoring the original ordering derived from the CWS algorithm.

3.2.3.3 Additional Methods

Cordeau et al. (2002) provides an excellent summary of common heuristic approaches from the past for vehicle routing problems including:

- Elementary Clustering
- Sweep Algorithm

- Fisher and Jaikumar Algorithm
- Limited Discretionary Search (Caseau, 1999)

While more suitable for real world problem sizes than exact methods, Cordeau et al. (2002) conclude that since c.2000 meta-heuristics have proved to be more efficient solvers, both in speed of execution and in the quality of the solutions produced so details of these methods are not presented here.

3.2.4 Meta-heuristics (MH)

Talbi (2009) states that compared to heuristics, Meta-Heuristics "represent more general approximate algorithms applicable to a large variety of optimization problems. They can be tailored to solve any optimization problem. Metaheuristics solve instances of problems that are believed to be hard in general, by exploring the usually large solution search space of these instances. These algorithms achieve this by reducing the effective size of the space and by exploring that space efficiently." Meta-heuristic approaches may use some of the heuristics introduced in Section 3.2.3 and provide alternative ways of exploring the solution space, avoiding local minima in the search of a global optimum. A summary of meta-heuristic solutions for the capacitated VRP is given by Gendreau et al. (2002). For problems with time windows, the summary by Bräysy and Gendreau (2005b) is also a good resource. Real-world scenarios are investigated by Cordeau et al. (2004b); Ropke (2005).

The meta-heuristics in this section are all solution improvement methods which assume an initial solution to the problem is given. The initial solution may be created by using one or more of the constructive heuristics introduced in Section 3.2.1. Meta-heuristics typically run a loop until either no further improvements are possible (hard to judge) or for a fixed amount of time or fixed number of iterations. In the following, an iteration lasts until the solution is changed; this may involve many thousands of attempted modifications that do not result in the solution being altered. Many meta-heuristics implement a re-initialisation of the solution if they detect that they are stuck in a local optimum. For vehicle routing problems, this involves discarding large portions of the existing solution and regenerating a solution using different construction

heuristics, or using the same construction heuristics with customers inserted in an alternative order.

3.2.4.1 Simulated Annealing (SA)

Lim and Zhu (2006) Present a simulated annealing (SA) approach to the multi-depot VRP with a fixed distribution of vehicles. A randomized best insertion (RBI) algorithm is used to generate initial solutions followed by a standard simulated annealing procedure. During a SA search, initially, changes that produce worse solutions are accepted, over time, the chance of a change being accepted if it produces a worse solution decreases until only improving moves are accepted. Finally, every route is optimised using a dynamic programming TSP solver. The resultant program is both fast and effective at minimizing the number of vehicles. During each iteration of simulated annealing one of four procedures (intra and inter route n-node chain relocation or swapping) is used.

3.2.4.2 Tabu search for PDPL

Benavent et al. (2015) present a multi-start tabu search (TS) (Glover (1990), see Appendix, Section A.1.2) approach for the multiple vehicle pickup and delivery problem with LIFO constraints and maximum time (PDPLT). The maximum time component of this is a duration constraint on routes that is common to all PDP's so I refer to this as the PDPL. CWS (Clarke and Wright, 1964), and two random schedule heuristics are used to build seed routes, and TS is employed to repeatedly remove and re-insert consignments. A number of traditional strategies are employed to prevent cycling and promote diversification of the solution. Results are presented on instances from Li and Lim (2003)

3.2.4.3 Ant Colony Optimisation (ACO)

Ant colony optimisations are inspired by how ants navigate in the real world. Good solutions leave a strong trail of pheromone that other ants can follow. Ant colony optimisations start with a solution, potentially randomly generated, and at each iteration generate a new solution. Better solutions add pheromone to the edges they visited. Subsequent new solutions are generated stochastically with each edge having a prob-

ability of being included, pheromone along an edge increases the probability of using that edge again. This technique ensures that good solutions are exploited to find better neighbouring solutions. If an edge with pheromone is used, the pheromone present is consumed leading to exploration of less well known edges in future searches.

Gambardella et al. (1999) propose a multiple ant colony system for the VRP with time windows (MACS-VRPTW). Multiple independent ant colonies are used, one for each objective, globally optimal solutions are shared between populations. The example given looks to be extensible to the multi-depot problem, as their method duplicates the depot with each visit to it. The objectives are to find the minimum number of vehicles required and to take the minimum amount of time. Two ant colonies are used, the first to reduce the number of vehicles and the second to reduce the time taken by the current best number of vehicles. Both colonies are reinitialised if either finds a better solution for their problem, with the best solution propagated across. The authors found their MACS-VRPTW was on par with or better than: adaptive memory programming, large neighbourhood search, guided local search and alternate k-exchange reduction. The approach was not tested against TS.

3.2.5 Hyper-heuristics (HH)

Hyper-heuristics (Cowling et al., 2001) operate at a higher level of abstraction than meta-heuristics approaches, rather than operating directly on the solution to a problem a hyper-heuristic manages a set of lower level heuristics (LLHs) to perform function minimisation. A hyper-heuristic has no knowledge of the problem domain that it is tasked with solving. For routing problems, a hyper-heuristic is given an initial solution and set of problem specific LSOs. The hyper-heuristic applies LSOs to make iterative changes to the solution, in order to maximise reward / minimise cost. With no knowledge of the underlying problem, a hyper-heuristic must choose an LSO to use at each iteration, based only on the previous performance of the LSOs, defined in terms of change in fitness and time taken. Burke et al. (2013) note that hyper-heuristics are either concerned with heuristic selection or heuristic generation. The ideas of hyper-heuristics pre-date the appearance of the term hyper-heuristic in literature. As a result, some of the approaches in this section may not refer to themselves as a HH.

Research has focused on different insertion, removal and local search operators, and on the heuristics that choose between operators at any point. For example, Gendreau et al. (2006) use neighbourhood search heuristics and ejection chains to tackle same-day courier PDP. Mitrović-Minić et al. (2004) use a double horizon approach with routing and scheduling sub-problems to schedule similar problems of a larger size. Albareda-Sambola et al. (2014) use probabilistic information to inform their routing of a multi-period VRP. Existing approaches to dynamic scheduling of PDPs (summarised in Bräysy and Gendreau (2005b)) often use a two-phase hyper-heuristic (Berbeglia et al., 2010): requests are first inserted into a schedule, then optimisation is performed, either on a route that has been changed or on an entire schedule.

3.2.5.1 Random Descent (RD)

The simplest hyper-heuristic is Random Descent (RD) (Remde et al., 2011). At each iteration, a random LSO is chosen and tested repeatedly. If the LSO produces a better solution, it is accepted and the current best solution is updated; otherwise it is discarded. A parameter is used to control the number of non-improving iterations that is allowed before the solution is re-initialised. In contrast to VNS (introduced below) RD uses LSOs in a random rather than systematic fashion and there is no shake procedure to escape local minima.

3.2.5.2 Variable Neighbourhood Search (VNS)

Mladenović and Hansen (1997) present Variable Neighbourhood Search (VNS) as a multi-purpose MH and demonstrated its effectiveness on the TSP. Recent implementations of VNS for the VRP are by Hansen and Mladenović (2001); Hansen et al. (2009). VNS combines simple local search techniques intelligently in order to reach globally maximal values quickly. A neighbourhood is defined as the set of possible moves from a given solution, for example, the set of 2-opt or 3-opt moves, as described in the Appendix, Section A.1.1. Since VNS works at the level of managing LLHs, I classify it as a HH. VNS relies on the fact that a global minimum is locally minimal for all neighbourhoods. The strength of VNS is thought to be due to two observations:

1. A local minimum of one neighbourhood is not necessarily a local minimum for

another neighbourhood.

2. Local minima for differing neighbourhoods tend to be clustered close together.

A clear description of VNS is given by Cowling and Keuthen (2005) and is modified for general use in Algorithm 3.2.2.

Algorithm 3.2.2 Procedure Variable Neighbourhood Search

Precondition: Create an initial tour T
Precondition: Set $T* = T, k = k_{min}$

```
1:  function VNS(k_min, k_max)
2:      repeat
3:          Apply Local Search to T
4:          if cost(T) < cost(T*)
5:              Set T* = T, k = k_min
6:          else
7:              Set T = T*
8:              if k < k_max
9:                  Set k = k + 1
10:         Perform k^th neighbourhood modification to T
11:     until Stopping criterion is met
```

Hansen et al. (2009) introduce various VNS techniques. The two most important of these are Variable Neighbourhood Descent (VND) and the General VNS (GVNS), shown in Algorithm 3.2.3.

A key point when using any VNS method is the choice and number of LSOs to be used; these can be any of the techniques introduced in Section 3.2.2, or new LSOs.

VNS for VRP with time windows

Bräysy (2003) uses a four phase technique for solving the VRPTW comprising route construction, route elimination and two VNS stages. In route construction, cheapest insertion heuristics are used to sequentially build starting routes, with Or-opt being used periodically to optimise the generated routes. Route elimination using ejection chains is carried out in phase two, the aim being to reduce the number of routes in the initial solution as much as possible, by reinserting all of one route's customers into another route. This may involve moving customers in adjacent routes to further

Algorithm 3.2.3 Procedure General Variable Neighbourhood Search and VND

Precondition: x = The current saved best solution.
Precondition: x' = The newly generated solution.
Precondition: k = The current neighbourhood.
Precondition: $N_i(x)$ = The i^{th} neighbourhood of solution x.
Precondition: t = Time.
Precondition: Label Neighbourhoods 1 through to k.

```
1:  function VND(x, k_max)
2:      repeat
3:          k ← 1
4:          repeat
5:              x' ← N_k(x)*
6:              if x' < x
7:                  x ← x'
8:                  k ← 1
9:              else
10:                 k ← k + 1
11:         until k = k_max
12:     until no improvement

13: function GVNS(x, k'_max, k_max, t_max)
14:     repeat
15:         k ← 1
16:         repeat
17:             x' = A random solution from the k^th neighbourhood.   ▷ Shaking
18:             x' = VND(x', k'_max)                ▷ Run current solution through VND
19:             if x' < x
20:                 x ← x'
21:                 k ← 1
22:             else
23:                 k ← k + 1
24:                 t ← current time
25:         until k = k_max
26:     until t = t_max
```

additional routes. Phases three and four employ the VND aspects of VNS, and use first-improvement rather than best-improvement to select the moves to accept. The overall approach does not include a random shake component and therefore proceeds in deterministic fashion from problem to solution. Four purpose made LSOs are used in the VNDs: ICROSS, IRP, I-opt and Or-opt, introduced in Section 3.2.2. The third and fourth stages of the technique differ only in that after the initial objective of minimising travel distance is achieved, the objective function is modified to also include a preference for minimising the delay per consignment.

VNS for MDVRP with time windows

Polacek et al. (2004) present an adaptation of VNS to the Multi-Depot VRPTW that is shown to be competitive with TS approaches, their approach's strengths are its scalability to real-world sized problems, simplicity and ease of extension to support additional constraints, both on the types of journeys and on fleet composition. Their approach uses CROSS and iCROSS operators with increasing chain lengths as the LSOs.

VNS for DARP

Parragh et al. (2009) apply VNS to the DARP using a 2-phase solution approach. Jarboui et al. (2013) apply VNS to the location routing problem where the location of depots is considered a variable in addition to other constraints. Both TS and VNS approaches to this problem work well highlighting their strengths and flexibility.

VNS for TSPPDL

Carrabs et al. (2007) Use VNS to solve the single vehicle, single depot pickup and delivery problem with LIFO. They refer to this problem as the TSPPD. They propose a block structure to deal with the LIFO constraint where each consignment is a block and multiple blocks can be combined to produce composite blocks. Operators then work with knowledge of these blocks to avoid producing routes which violate the LIFO constraint. This block structure would need substantial modifications for the PMDP as due to having more than one delivery associated with each pickup there are many

ways that two consignment blocks could be combined together (discussed in detail in Section 4.2).

VNS for PDPL

Li et al. (2011) and Cheang et al. (2012) both make use of VNS to solve instances of PDP with LIFO loading (PDPL). Cheang et al. (2012) call this a multiple PDPL (MPDPL) but the use of multiple here is a misnomer, referring to having more than one request to service, the problem is in no way distinct from PDPL. Both Li et al. (2011) and Cheang et al. (2012) use a tree structure to represent a route that adheres to LIFO constraints and this works well as conceptually the children of a node fall between the pickup and delivery of the consignment represented by the node. It would not work in a situation where there are multiple deliveries per consignment and other consignments can be inserted between any pickup or delivery, such as the PMDP.

3.2.5.3 Greedy Randomised Adaptive Search Procedure

GRASP ranks customers based on either how difficult they are to service or the quality of solution possible by their inclusion. GRASP is an iterative approach in which each iteration consists of a construction and optimisation step. A set of solutions is stored and diversity in the solution set is maintained through a restricted candidate list which prevents well ranked individuals from being used too frequently. Local search using LSOs is carried out in the optimisation step. Kontoravdis and Bard (1995) present a GRASP for the VRP with time windows. Cherkesly et al. (2015) use a three phase approach that creates multiple initial routes using a GRASP to solve a problem with LIFO constraints. VND is used for local search, and new solutions are created from existing routes using a diversification strategy derived from Rochat and Taillard (1995). A crossover step is used to combine solutions together to form additional candidate solutions.

3.2.5.4 Large Neighbourhood Search (LNS)

Shaw (1998) introduces LNS as a process of continual relaxation and re-optimisation. Customer visits are removed from a solution and the resulting solution is optimised. The

removed customers are then re-inserted into the schedule using minimum cost branch-and-bound. LNS is shown to compare favourably with contemporary MH solution approaches "while being significantly simpler" and being able to effectively address the additional real-world constraints of their problem.

3.2.5.5 Adaptive Large Neighbourhood Search (ALNS)

Ropke and Pisinger (2005) and Pisinger and Ropke (2007) present Adaptive Large Neighbourhood Search (ALNS) following a similar fashion to VNS except that the neighbourhoods to be searched are not structured in a hierarchical fashion. They are instead built up from a selection of insertion and removal operators. SA is used to allow exploration at the start of the search. ALNS is shown to produce good solutions to PDPs with up to 500 requests, producing the best known solutions to many of the benchmark problems of Li and Lim (2003). Demir et al. (2012) use ALNS and a speed optimisation algorithm in a two phase approach to solving the pollution-routing problem (Section 2.2.4), they carry out extensive computational experimentation to confirm the efficiency of their algorithm on instances of up to 200 nodes. Koç et al. (2014) combine this with an evolutionary algorithm capable of solving different classes of problem without modification.

3.2.5.6 Parallel ALNS

Pillac et al. (2012) present a parallel adaptive large neighbourhood search (pALNS) to generate a set of non-dominated solutions for the single depot dynamic VRP that a decision maker may choose between. Their two objectives are distance minimisation and driver inconvenience but this could be altered to investigate delay or other factors. Their destroy and repair scheme for local moves includes the following destroy operators:

- Random - choose a random customer to remove from the solution.
- Related - choose one customer at random then the most related customer to it (weighted product of distance and time window)
- Critical - remove one customer that minimises cost of solution.

Customers are re-inserted using *regret-q heuristics* with three regret levels (1, 2 & 3) at level 1 this corresponds to a greedy best insertion heuristic. The adaptive nature of the algorithm deals with the selection of destroy and repair operators, these are chosen randomly with a probability which changes throughout execution to favour more successful operators. Pillac et al. (2012) do not consider variable travel times, removed consignments or vehicle breakdowns.

3.2.5.7 Q-Learning Selection (QL)

QL (Watkins and Dayan, 1992) is a learning hyper-heuristic that shares a number of similarities with the choice function of Cowling et al. (2001). QL attempts to learn good sequences of LSOs, these are stored in a Q-state dictionary which maps sequences of n LSOs to Q-values. At each iteration, QL identifies sequences from the dictionary that start with the most recently used $n-1$ LSOs. The next LSO to try is chosen based on a roulette selection over these entries Q-values. The Q-values in the dictionary are updated using the function:

$$Q(s,a) = Q(s,a) + \alpha \left[r + \left(\gamma \max_{a'} Q(s',a') \right) - Q(s,a) \right] \tag{3.8}$$

where s is the current $1, \ldots, n-1$ sequence of LSOs, a is the next LSO to use and s' is the resultant sequence, after this operator is used. The reward r is set to the improvement produced by the operator, divided by the time taken to find it or to half the smallest observed reward if no improvement is found. α, the learning rate, γ, the discount factor, and n, the length of LSO sequences to store, are parameters. Traditional QL allows non-improving moves. However, since our problem has a very limited set of improving moves, in Chapter 5 we have adapted it to only accept moves which result in better solutions.

3.2.5.8 Binary Exponential Back-off (BEBO)

BEBO (Remde et al., 2011) is a tabu based learning hyper-heuristic derived from methods used to avoid packet collision in communications systems. If a packet collision is detected an exponential time backoff is applied, delaying subsequent packets from a specific sender, reducing future collisions. In hyper-heuristic form, a tabu list is stored

along with a backoff value for each LSO. At each iteration, all non-tabued LSOs are tested. These are then categorised as good or bad. A "Bad" LSO may be any non-improving move, or may be defined as being one of the worst x% of LSOs. "Good" LSOs have their backoff set to backoff-min. "Bad" LSOs have their backoff set to backoff$^2 + 1$. A tabu tenure is then chosen for each LSO, randomly between backoff-min and its backoff.

3.2.6 Hybrid Meta-heuristics (HM)

Often, the use of a single meta-heuristic prevents some potentially interesting areas of the solution space from being investigated. Hybrid meta-heuristics (HMs) seek to overcome this problem by combining effective features from a variety of sources (Blum et al., 2011). They typically rely heavily on domain knowledge, and transfer of methods across problem domains is not usually possible.

Fox (1993) combines features of SA, TS and GA (Genetic Algorithms, see Appendix, Section A.1.3), retaining spatial and temporal memory from TS, but dropping the necessity for aspiration criteria by utilising a cooling schedule for acceptance taken from SA. New solutions are generated using crossover and mutation strategies from GAs. Fox (1993) proves correctness and goes on to highlight the speed and parallelizable nature of this strategy though no results for VRP problems are presented. Moon et al. (2012) compare integer linear programming (ILP), a GA and a hybrid approach based on SA to solve a real-world VRP. The GA and hybrid approaches are found to perform similarly and both outperform ILP in terms of CPU time. Cordeau and Maischberger (2012) embeds a TS meta-heuristic within iterated local search and uses a simple parallel computing framework to take advantage of modern multi-core processors.

Other HMs e.g. Caric et al. (2007); Ostertag et al. (2008); Paraskevopoulos et al. (2008); Pirkwieser and Raidl (2009); Repoussis et al. (2006), have proved successful when applied to the VRP but have seen limited applicability to the PDP.

3.2.6.1 Hybrid Variable Neighbourhood Tabu Search

Hybridisation of VNS and TS has proved beneficial for VRPs; Paraskevopoulos et al. (2008) presents a reactive variable neighbourhood tabu search (reVNTS) for the hetero-

geneous VRP and Belhaiza et al. (2013) presents a hybrid variable neighbourhood tabu search (HVNTS) for the VRP with time windows. Modification operators increasing in complexity are used as the LSOs, from simple, intra-route, 2-opt moves to 3-route extensions of the CROSS and relocate operators. A tabu list is used to prevent cycling of solutions. The method presented in Chapter 5 has similarities with these methods.

3.2.6.2 Two Phase Meta-heuristics

Many methods can be called two phase. The term may refer to types of operation used, e.g. a construction phase followed by an optimisation phase. In this description, almost all meta-heuristics for the VRP could be described as two phase. Separation of a solver into construction and optimisation phases usually enables a method to be portable to a dynamic situation. When a new customer needs to be added to the current solution, the construction phase is triggered. Otherwise, the optimisation phase is used to improve the current solution. The two phase HMs in this section refer to optimisation methods that have two phases. The first phase minimises one objective whilst the second phase minimises a second.

Gehring and Homberger (1999) present a parallel hybrid evolutionary meta-heuristic for the VRPTW. The objectives of their approach are firstly to minimise number of vehicles used and secondly to minimise travel distance. The first phase of their approach utilises a (1, λ) evolution strategy, described in Algorithm 3.2.4. The second phase is a TS utilizing the same set of move operators.

This method was found to produce results similar to TS approaches whilst being easy to parallelise coarsely over a network of PCs using a master and slave model.

3.2.6.3 Hybrid Genetic Algorithm

Baker and Ayechew (2003) present a generic GA that generates results within 2.5% of TS approaches on a number of famous VRP test cases from Solomon (1987). Baker and Ayechew (2003) propose enhancements to the GA that reduce this deficit to 0.5%. The approach is named a hybrid GA, as it uses problem specific knowledge to accelerate the convergence of the GA. A couple of techniques are introduced. Firstly, generated solutions with non-zero unfitness (see Section A.1.3) are run through a procedure to

Algorithm 3.2.4 λ Evolution

Precondition: Existing Best Solution S^*

```
1: repeat
2:     for λ iterations
3:         Select a move operator                          ▷ (2-opt, 3-opt, etc.)
4:         Generate a candidate solution using this move
5:         Optimise the result using Or-opt
6:         Evaluate the solution and add it to the neighbourhood set
7:     S ← Best solution from the neighbourhood set
8:     if cost(S) < cost(S*)
9:         S* ← S
10: until The time limit
```

swap nodes between adjacent routes, allowing increases in distance if unfitness levels reduce, repeated until unfitness is zero or no further reductions are possible in this manner. Secondly, every 10,000 iterations two types of neighbourhood search are carried out. In the first, 2-opt is carried out on the whole solution, treated as a TSP with the depot replicated between each route. In the second λ-interchange is carried out between adjacent routes. The reliance on polar angles from the base location makes this approach less suitable for multi-depot problems. Keeping track of adjacent routes is not scalable to larger problems with thousands of trucks and multiple depots.

3.2.6.4 Memetic algorithms

Memetic algorithms combine population based GA with local search techniques to reduce the likelihood of premature convergence. Nagata and Bräysy (2008, 2009) present memetic algorithms to solve the CVRP. Neighbourhoods are composed using EAX (Nagata, 2007, Section 3.2.2.10), 2-opt, relocate and swap operators. Infeasible solutions are generated in this process, which are addressed with an efficient modification algorithm. A method using only new routes with "*don't look*" bids was found to outperform other methods on 26 instances from the Christofides (1979), Golden and Assad (1988) and Taillard (1993) benchmarks, finding new best solutions to 12 of the problems.

3.2.7 Dynamic Solutions

3.2.7.1 Reactive Vehicle Routing Framework

Zhu and Ong (2000) present an approach where each vehicle is represented as an agent and as new requests come in, incremental local optimization is carried out. The system is inherently parallel and fast at reacting to incoming events. Their implementation is based on a discretized model of time which is a sensible abstraction for this problem.

3.2.7.2 Multiple Plan Approach

Bent and Van Hentenryck (2004) propose a Discrete Event Simulation (DES) of the VRP with the Multiple Plan Approach (MPA) which creates and maintains a number of feasible solutions. A least commitment strategy is used to choose the plan used at each event, denoted σ^*. The list of feasible solutions is updated on four events:

1. **Customer Request.** A newly received request results in n feasible solution plans where n is the number of feasible insertion locations for the new request in all of the previous plans.

2. **Timeout.** If σ^* calls for a truck to leave a node, all plans where the truck stays at that node become infeasible and are removed from the set of plans.

3. **Vehicle Departure.** The opposite of Timeout. If σ^* calls for a truck to stay at a node, plans which have the truck leaving the node become infeasible.

4. **Plan Generation.** Whenever a new plan is generated σ^* is regenerated.

MPA assumes that trucks wait at delivery locations until the latest time to reach the next customer for the beginning of their time window in order to give the most time possible for rerouteing.

3.2.7.3 Multiple Scenario Approach

Bent and Van Hentenryck (2004) extend the MPA introduced above to include predicted future events in the plans generated. The resulting Multiple Scenario Approach (MSA) is otherwise identical to MPA and outperforms it in situations with a high degree of dynamic consignments.

3.2.7.4 Online Stochastic Programming

Hentenryck et al. (2009) use randomised predictions of future events as part of the initial planning in dynamic problems. Likely events are stochastically added to the set of known nodes with equal probability to that of the event occurring. The vehicle traverses the route as planned if the expected consignments arrive and updates its route to accommodate new consignments if appropriate.

Albareda-Sambola et al. (2014) present a dynamic multi-period VRP with probabilistic information. Their single depot problem consists of a set of known customer locations with known probability distributions for requiring service however the exact demand is not known. Service requests consist of a number of potential time windows. Once again variable travel times, removed consignments and vehicle breakdowns are not considered. They use VNS with an additional skipping procedure to break out of local optimum. A compatibility index is built between pairs of customers, representing the potential savings (similar to CWS) if these were to both occur. When one customer's request is received it is delayed until the potential savings are nullified by not servicing it.

3.3 Discussion and Summary

We presented a selection of exact and non-exact methods for solving a variety of vehicle routing and scheduling problems. The exact algorithms presented are often only capable of solving small scale problems in reasonable time, which is why non-exact methods are crucially important. A number of construction and modification LSOs are presented that can make small changes to a schedule and heuristic methods that use these operators iteratively to produce good approximate solutions in reasonable time. Though useful, heuristic methods can often become trapped in local optima, meta-heuristic solutions such as TS and VNS offer a wide range of techniques to escape these traps and recent hybrid-meta-heuristics represent the current cutting edge of research in VRP. Hyper-heuristics attempt to provide the functionality of meta-heuristics across problem domains without being constrained to a specific purpose and often include learning mechanisms to accelerate the search.

The focus of my research is the Pickup and Multiple Delivery Problem (PMDP) which shares a number of similarities with both PDP and VRP problems, it is a relatively unexplored area of research, and is derived from real-world logistical operations. Relevant ideas and concepts including LIFO loading, soft time windows and local search operators have been adapted to suit this problem (Chapter 5). Lee (2013) investigates merging consignments using a two-phase approach which first intelligently combines consignments then creates routes using a sweep heuristic. Our work differs in two key respects. Firstly, our problem is dynamic, not static, and secondly it is a pickup and multiple delivery problem, not based on a centralised depot. Packing first and routing second is also explored by Bortfeldt and Homberger (2013) who add a number of constraints related to box size, weight and stacking. Each consignment is first solved as a 3D strip packing problem; the length of these strips then represents the capacity requirement and constraints in a VRP, solved using the two-phase heuristic involving (μ, λ)-evolution and tabu search of Homberger and Gehring (2005). Our model does not go into as much detail, though is dynamic pickup and delivery, where their model is of a static distribution centre.

4

Model: The Pickup and Multiple Delivery Problem (PMDP)

The PMDP is a graph based vehicle routing and scheduling problem, consisting of servicing customer consignments within time and vehicle constraints. A consignment is characterised by a single pickup event, at a specific time and location, and a sequence of delivery events which must be serviced in order. The PMDP is closely related to the one-to-one, pickup and delivery problem (PDP), common in taxi dispatching and Dial-a-Ride problems (DARP) (Berbeglia et al., 2007), and the one-to-many-to-one, vehicle routing problem (VRP) for delivery problems with centralised goods depots (see Chapter 2). The PMDP falls somewhere between and can be classified as a one-to-many problem where one pickup is associated with many deliveries as in VRP but does not have to immediately return to the pickup location. It differs from the one-to-one definition, in that a customer request may specify more than one delivery location. However, there is no central dispatching / receiving depot, as in one-to-many-to-one problems.

Our real-world problem based on the experiences of Transfaction Ltd., a logistics analysis company working with UK retailers, is the dynamic scheduling of shared loads for truck haulage in the UK. We model this problem as a dynamic PMDP. Waisanen et al. (2007) is the only mention of a model similar to PMDP but is unpublished and concerned with communications networks. Berbeglia et al. (2010) note that one-to-

many problems such as the PMDP are of significant real-world importance requiring additional research; and that to date, no realistic model has been published.

4.1 Introduction

We have chosen to model our problem as a PMDP rather than as a PDP due to a number of additional constraints that must be satisfied to meet the requirements of the retailers Transfaction Ltd. works with, such as vehicle capacity, soft time windows and driver working time rules. The problem is defined in terms of consignments which include a single pickup location and one or more delivery locations. Consignments vary in size, and may be able to share one delivery vehicle, to save cost. A key constraint is that each vehicle must be unloaded in the reverse order to the loading order: deliveries from one vehicle are constrained to a last-in, first-out (LIFO) order. Concretely, consignment A may be interrupted by another if all of the second consignment's deliveries are serviced before continuing with consignment A's deliveries. The PMDP has been designed as an extension to the PDP rather than an extension to the PDPL since the tree structures utilised to handle LIFO constraints in PDPL are overly complicated when a consignment may be interrupted at any point. It would be possible to map PMDP as a tree structure but each leg of a consignment (from pickup to delivery or delivery to delivery) would have to be its own node in the tree, making the trees much larger with a higher branching factor than in PDPL. Additionally it would be much harder to turn the LIFO constraint off for comparisons to the state-of-the-art to be made.

4.2 Consignments

Our model for the PMDP is drawn from the formulation of the multi-commodity vehicle routing problem with pickup and delivery and time windows (MCVRPPDTW) presented by Desaulniers et al. (2002) (more simply, commonly and henceforth, referred to as the pickup and delivery problem (PDP)). The main difference between the PMDP and PDP is expressed as follows. Instead of a request i being identified by nodes i and $n + i$, a consignment c is identified by a number of requests and a received time (p_c, D_c, t_c) where p_c is the pickup-request and $D_c = d_c^1, \ldots, d_c^{nc}$ is the sequence of delivery-requests, as shown in Figure 4.1. Each consignment has a received time t_c, when it is

first seen by the system. We define C as the set of consignments.

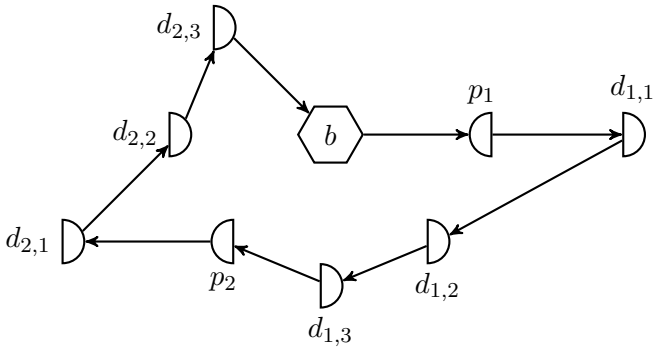


Fig. 4.1: An example PMDP with a single depot, b, and two consignments $(p_1, d_{1,1}, d_{1,2}, d_{1,3})$ and $(p_2, d_{2,1}, d_{2,2}, d_{2,3})$.

The PMDP is defined on a directed graph $DG = (N, A)$ where A is the arc set and N is the node set. Each pickup or delivery request r is identified by (n_r, l_r $[t_r^{start}, t_r^{end}]$, $tt_r^{service}$) where n_r is the location, l_r is the load, $[t_r^{start}, t_r^{end}]$ represents the start and end time of the arrival window respectively and $tt_r^{service}$ is the service time, all described in more detail below. We define R as the set of requests where $R = P \cup D \cup O$, P being the set of pickup-requests and D the set of delivery-requests. O is the set of origins which are dummy requests used to represent the multiple depots of the problem.

Location n_r is the location of request r in two dimensional Cartesian space (there may be multiple requests per location). An arc (r, u) between requests r and u has distance that can be represented by nn_{ru}. Similarly, we define tt_{ru} as the time taken to travel an arc at an average speed. Pickup and delivery locations are based on UK postcodes which are translated to standard Northings and Eastings. For simplicity we model distances as straight-lines with travel times based on analysis of a large data set. This is accurate enough for our "strategic" investigation of the problem.

We consider load $l \in L$ in terms of weight and measure it as a percentage of maximum vehicle load. This works well for fuel consumption calculations, but would need additional steps if modified for volumetric use. The load of a request r is denoted by l_r; this is positive for pickup requests and negative for delivery requests. For a consignment c the sum of the loads for requests in R_c must equal 0.

In our real-world scenarios, there are two types of consignments, 'linehauls' which represent goods delivery from a supply location to many store branches and 'backhauls'

representing the return of empty pallets, unsold goods etc. from branches to supply locations. Linehauls are generally of larger load than backhauls which are point to point and have no precedence relationships with linehauls, though they must still adhere to LIFO loading.

4.2.1 Time Windows

For a request r an arrival time window is defined between [t_r^{start} and t_r^{end}], representing when the customer would like to be serviced.

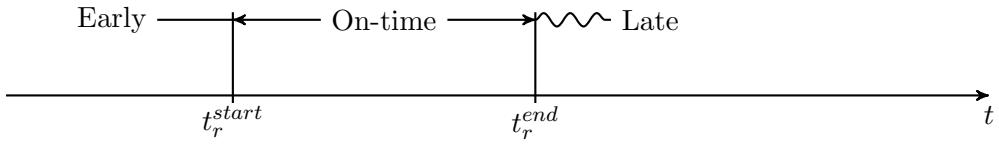


Fig. 4.2: Arrival time window for each request.

Figure 4.2 shows the three states possible when arriving at a request r. Unlike many PDPs, real-world logistics problems such as those of Transfaction Ltd. have soft time windows (see Section 2.2.5) at both pickup and delivery locations. A vehicle can therefore arrive at a location early, on-time or late. There is no penalty for arriving at a location early, though both vehicle and driver will have to wait until the specified earliest time to be serviced. If the vehicle arrives after t_r^{end} the request is said to be delayed by $t_r - t_r^{end}$ where t_r is the actual time that request r is serviced. If delayed, a penalty is applied (see Section 4.4.3). Not shown in Figure 4.2 is the service time required for loading / unloading at a customer location, $tt_r^{service}$. Waiting and delay are both calculated based on a vehicle's arrival time at a location. For example, if a vehicle arrives at the latest arrival time the delay is 0 even though the vehicle will not leave until $tt_r^{service} + t_r^{end}$ (after the latest arrival time t_r^{end}).

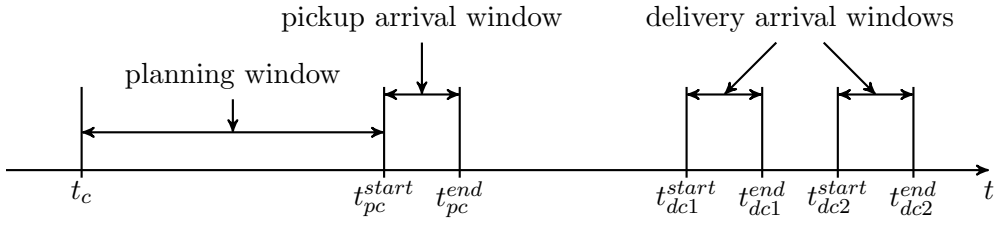


Fig. 4.3: Time windows for a consignment with two deliveries.

In Figure 4.3 a consignment consisting of pickup p_c and deliveries d_{c1} and d_{c2} is shown. There are three arrival windows, one for each request. We define a planning window as the amount of warning that the scheduler gets before a pickup is required, defined between [t_c and t_{pc}^{start}]. The effect of changing the size of these windows and therefore their importance from a cost perspective is investigated in Chapter 8.

Given a list of scheduled consignments, $1 - n$, and an insertion location i s.t. $1 \leq i \leq n$. An additional point, j (with time window (t_j^{start}, t_j^{end})), may be considered for insertion at point i by computing the earliest feasible insertion time e_j and the latest feasible insertion time l_j (Yang et al., 2014):

$$e_j = max(e_{i-1} + tt_j^{service} + t_{i-1,j}, t_j^{start}), \tag{4.1}$$

$$l_j = min(l_i - tt_j^{service} - t_{j,i}, t_j^{end}), \tag{4.2}$$

where $t_{i,j}$ is the time taken to travel from point i to point j and $tt_j^{service}$ is the time taken to load / unload the vehicle. If $e_j \leq l_j$, the insertion point is feasible. In problems with hard time window constraints such as the Li and Lim (2003) benchmarks, if the insertion point is not feasible, the insertion is not considered. However, in our real-world problem, these need to be considered. If an insertion point is not within a soft time window then the difference between the insertion point and the soft time window is denoted either as waiting time or delayed time, with associated costs as described in Section 4.4.3.

4.3 Vehicles

To service a set of consignments, a fleet of vehicles must be effectively routed and scheduled. The PMDP is a multi-depot problem with many vehicles, each of which may be assigned a number of routes. Let K be the set of all routes. For each route $k \in K$ define the directed sub-graph $DG_k = (N_k, A_k)$ where $N_k = N \cup o(k)$ is the set of requests inclusive of the route's origin location (dummy request) $o(k)$. The subset A_k of $N_k \times N_k$ comprises all feasible arcs for route k.

A vehicle has an associated capacity $l_k \in L$ of the same dimension as request loads. The load of a vehicle changes after each visit to a request r. We define ll_{rk} as the load

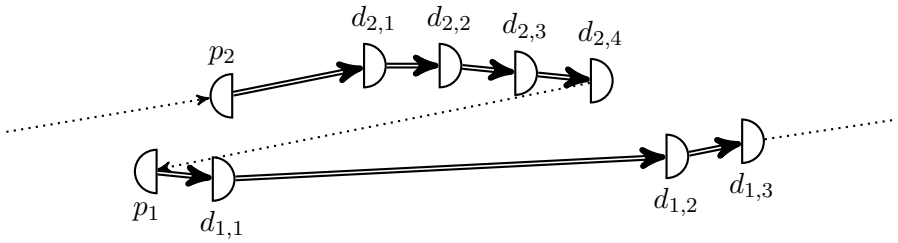
on truck-route k after servicing request r. ll_{rk} must always be less than or equal to l_k.

A vehicle servicing route k is assumed to leave unloaded from its base depot at time $t^{start}_{o(k)}$. Each feasible pickup and delivery tour for this route corresponds to a path beginning and terminating at $o(k)$ in network DG_k, servicing each request at most once. If a route contains consignment c it must serve request p_c and all requests in D_c in order, not necessarily consecutively, but adhering to LIFO queuing. A consignment may be 'nested' wholly between requests in another consignment and there is no depth limit for nested consignments (only capacity constraints). If truck-route k services request $r \in R$ it does so at time t_{rk}, at or after time t^{start}_r, when the service time $tt^{service}_r$ begins. If the vehicle arrives after t^{end}_r, the request is delayed, we define this as $tt^{delay}_{rk} = \max(0, (t_{rk} - t^{end}_r))$. Similarly waiting time is the time between the scheduled arrival time t_{rk} and the start of the requested arrival window t^{start}_r ($tt^{waiting}_{rk} = \max(0, (t^{start}_r - t_{rk}))$). This is wasted time that has an associated time cost. A vehicle has an associated time limit for its routes defined as tt_k. A truck may be assigned multiple routes commencing and terminating at its origin o_k, providing all routes satisfy working time rules (route time duration is less than tt_k) and the routes do not overlap in time (one route must terminate at $o(k)$ before the next can start.

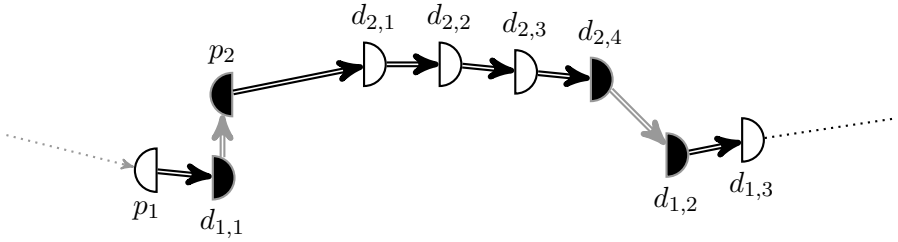
4.3.1 Capacity Constraints

In the literature there is little research into vehicle sharing in PDPs; many authors focus on taxi (Horn, 2002) or emergency services (Cordeau et al., 2007) problems where vehicle sharing is not a feature of the underlying problem. Capacity and similar constraints are more common in DARP (Toth and Vigo, 1997; Cordeau and Laporte, 2003) and VRP, including recent work by Lee (2013) and Bortfeldt and Homberger (2013). Both of these favour a two phase approach in which consignments are identified as candidates for combination before route planning commences. This works well for static VRP models but would require significant adaptation for the PDP, as combination must take into account pickup and delivery locations as well as time windows. For the dynamic case, problems arise because not all consignments are known a priori.

The vehicles we consider may only be loaded and unloaded from the rear. Since it is a legal requirement for trucks to be loaded evenly, from left to right (Bortfeldt and



(a) Before



(b) After

Fig. 4.4: Combining two consignments, here consignment $p_2 ... d_{2,4}$ is nested between two of $p1$'s deliveries $d_{1,1}$ and $d_{1,2}$ (Symbols are defined in Figure 2.2, page 30).

Homberger, 2013; Department for Transport, 2015), when combining consignments we are constrained by a last-in, first-out (LIFO) packing scheme. If this were not the case, it would require the unpacking and repacking of loads on a truck at some delivery locations which is time consuming and not generally acceptable. Therefore, the only form of combination allowed is *nesting*. A single layer nesting is shown in Figure 4.4. Here, consignment 1 is collected and its first delivery ($d_{1,1}$) is made. At this point there is sufficient capacity left in the truck to fully service (pickup and deliver) all of consignment 2, before returning to consignment 1 and delivering $d_{1,2}$ & $d_{1,3}$. Consignments may be interrupted at any point, the key constraint is that nested consignments must be fully completed before returning to the original consignment. There are no artificial constraints on the number of consignments nested or the number of layers of nested consignments. In practice both of these are limited by schedule duration and vehicle capacity constraints.

4.4 Objective - Cost Function

The formulation of our model objective requires a binary flow variable b_{ruk}, which is set equal to one if arc $(r, u) \in A_k$ is used by vehicle k and 0 otherwise. The goal is to

minimise the total cost of servicing all requests $r \in R$:

$$\min \sum_{k \in K} \sum_{(r,u) \in A_k} (nc(nn_{ru}, ll_{rk}) + tc(ruk) + dc(tt_{rk}^{delay})) * b_{ruk} \tag{4.3}$$

subject to the constraints in Tables 4.1 and 4.2, described in Section 4.5. This objective represents a linear combination of the dominant factors influencing the real cost of servicing a set of customer consignments including vehicle maintenance, fuel, driver pay and delay penalties, described below.

4.4.1 Distance Cost

The driving cost $nc(nn_{ru}, ll_{rk})$ of an arc ru is a function of distance nn_{ru} and the load ll_{rk} of the assigned truck k after servicing request r. Fuel cost per kilometre is calculated as consumption in litres per 100 km multiplied by the cost of fuel per litre divided by 100.

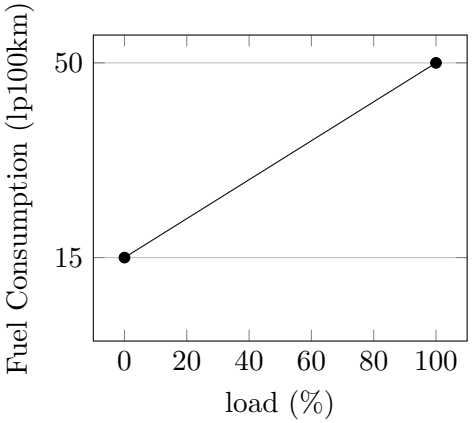


Fig. 4.5: Fuel consumption at load.

Figure 4.5 shows the fuel consumption of a vehicle based on its load for any arc. A fuel cost of £1.483 per litre was chosen when this pricing was introduced and has not been changed since to account for recent changes in fuel price; this has been done so that we can compare our previous results to current experiments. A fixed maintenance cost of 10p per km is added to the fuel cost per km based on data in Dff International Ltd. (2014).

4.4.2 Time Cost

The time cost for each leg of a route, between requests r and u can be expressed as $tc(ruk) = tc(tt_{ru} + tt_{rk}^{waiting} + tt_{r}^{service})$, note that there is no waiting or service time at the first or last request; these are both dummy requests $o(k)$ representing the depot. It is assumed that trucks travel at a constant 56 km/h, based on analysis of real-world data, outlined in Section 6.1.3.4. The cost per unit time is set in accordance with industry practice at £12 per hour (Stobart, 2013). The cost of waiting at a request which has been arrived at early is shown in Figure 4.6.

4.4.3 Delay Cost

An additional cost is associated with delayed consignments and is based on a stepwise function of the time that a consignment is delayed, this is represented by $dc(tt_{rk}^{delay})$. This function has been specified by Transfaction Ltd. as £40 per hour, after the first hour of delay and is shown in Figure 4.6.

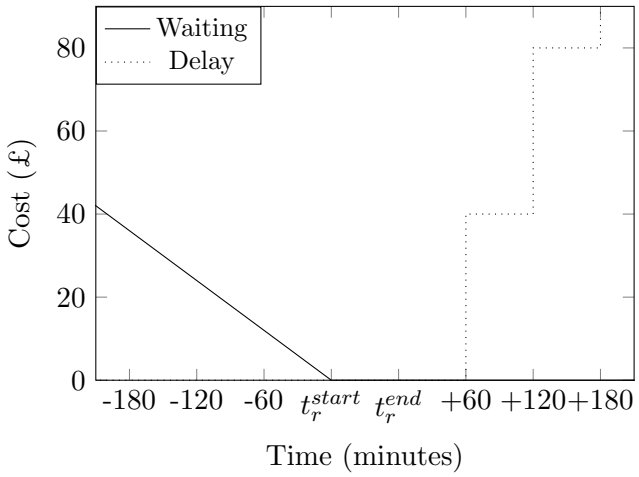


Fig. 4.6: Time costs around the arrival time window of request r, from t_r^{start} to t_r^{end}.

4.5 Constraints

The real-world PMDP has a number of constraints, some of which have been mentioned already. This section presents mathematical descriptions for all the constraints in the problem. The constraints in Table 4.1 have been adapted and expanded from the formulation for the PDP presented by Desaulniers et al. (2002); Table 4.2 presents the

additional, new constraints for the PMDP.

Tab. 4.1: Adapted constraints from Desaulniers et al. (2002), here $\equiv$ implies that this constraint is equivalent to a constraint presented by Desaulniers et al. and $*$ implies that this constraint has been modified for the PMDP.

=	Constraint	Applied to	#
$\equiv$	$\sum_{k\in K}\sum_{u\in R_k} b_{ruk} = 1$	$\forall r \in R$	(4.4)
$*$	$\sum_{u\in P_k} b_{ruk} * \lvert D_j\rvert - \sum_{w\in D_j} b_{rwk} = 0$	$\forall k \in K, r \in R_k$	(4.5)
$*$	Removed		(4.6)
$*$	Removed		(4.7)
$*$	Removed		(4.8)
$\equiv$	$b_{ruk}\left(t_{rk} + tt_r^{service} + tt_{ru} - t_{uk}\right) \leq 0$	$\forall k \in K, (r,u) \in A_k$	(4.9)
$*$	$t_r^{start} \leq t_r^{end}, t_r^{start} \leq t_{rk}$	$\forall k \in K, r \in R_k$	(4.10)
$*$	$t_{rk} + tt_r^{service} + tt_{ru} \leq r_{uk}$	$\forall k \in K, r \in P_k, u \in D_r$	(4.11)
$\equiv$	$b_{ruk}\left(ll_{rk} + l_u - ll_{uk}\right) = 0$	$\forall k \in K, (r,u) \in A_k$	(4.12)
$*$	$0 < l_r \leq ll_{rk} \leq l_k$	$\forall k \in K, r \in P_k$	(4.13)
$*$	$l_r + \sum_{u\in D_r} l_u = 0$	$\forall r \in P$	(4.14)
$\equiv$	$l_o(k) = 0$	$\forall k \in K$	(4.15)
$\equiv$	$b_{ruk} \geq 0$	$\forall k \in K, (r,u) \in A_k$	(4.16)
$\equiv$	b_{ruk} binary	$\forall k \in K, (r,u) \in A_k$	(4.17)

Constraints (4.4) and (4.5) ensure that each arc is only included once and that a pickup and all its corresponding deliveries are handled by the same truck. Here, $|D_u|$ is the number of delivery-requests for pickup-request u. Constraint (4.5) is non-standard for the PDP and is necessary as there may be multiple delivery-requests per pickup-request. It states that for each pickup request there exists a $b_{ruk} = 1$ and that this, multiplied by the number of deliveries, is the same as the number of arcs that end at each of the corresponding delivery requests. Unlike Desaulniers et al. (2002), we are not interested in multicommodity flow, so we omit Constraints (4.6) to (4.8). Constraint (4.9), imposing total schedule duration, remains unchanged. To model the real world constraints on truck working times, each vehicle may have any number of routes; however these must all start on different days (and, on each day, a fixed length schedule is allowed to start at any time)[1]. Constraints (4.10) and (4.11) have been modified to allow for soft time windows. Constraints (4.12) to (4.14) specify that a pickup node must have positive load and that deliveries must have negative load,

[1] In reality there are more complex rules defining working times for truck drivers including limits on the amount of work, outside of driving, that is allowed and the ability to work extra hours if these are made up for in the following 2 weeks (Department for Transport, 2016), we ignore these rules as they are designed to account for unforeseen traffic issues rather than as an area we should look to exploit. Another area we currently do not consider is driver pairing, where two drivers share a truck and can therefore drive for longer before having to take breaks.

and that the sum of pickup and delivery loads is zero. The initial vehicle load, non-negativity and binary requirements (Constraints (4.15) to (4.17)) are the same as those in Desaulniers et al. (2002).

Tab. 4.2: New constraints for the PMDP.

Constraint	Applied to	#
$\lvert P_c \rvert = 1$	$\forall i \in I$	(4.18)
$\lvert D_c \rvert \geq 1$	$\forall i \in I$	(4.19)
$t_{rk} < t_{uk}$	$\forall k \in K, r \in P_k, u \in D_r$	(4.20)
$t_{rk} < t_{uk} \Rightarrow t_{vk} < t_{wk}$	$\forall k \in K,\ \forall r, u \in P_k,\ \forall v \in D_u,\ \forall w \in D_r$	(4.21)
$\sum_{(r,u)\in A_k} b_{ruk} \left(tt_r^{service} + tt_{ru}\right) \leq tt_k$	$\forall k \in K$	(4.22)

Table 4.2 presents the new constraints for the PMDP: (4.18) and (4.19) specify that a request has exactly one pickup and may have arbitrarily many deliveries. (4.20) specifies the precedence between a pickup and its deliveries while (4.21) expresses the LIFO constraint. Finally, (4.22) specifies that the length (in time) of any tour is less than a value E_k which may be set according to local conditions.

Minimising k, the number of vehicles used, is not considered as part of this problem since we have a fixed vehicle fleet, though it is kept low as a side effect of the heuristics used. For each truck, requests may be nested within other requests if LIFO and capacity constraints are not violated.

4.6 Discussion and Summary

This chapter presents the mathematical model for the PMDP, built with data analysis and insights from Transfaction Ltd.. The objective of minimising cost is detailed and broken down into its constituent time, distance and penalty components. Constraints are specified to encapsulate LIFO packing, soft time windows, delay penalties and driver working hours.

Though we have used specific values relevant to Transfaction Ltd. in the description of this model. The PMDP is generic enough to be applied to other similar problems with only minor changes being needed to costs, delay penalty curves etc. Our straight line distance assumption could be made more realistic with adequate data however, since modelling real-world traffic flows is outside the scope of this research we did not invest the time required to do this, we do not feel the overall conclusions of this thesis

would change with real-world distance though the absolute value of any single solution would be different.

The PDP which we use to compare our solution methods can be seen as a special case of PMDP where all consignments have only one delivery and there is no LIFO constraint. Existing solution approaches for the PDP have been considered as candidates for basing a solution to the PMDP during the design and implementation of the VNDM HH introduced in Chapter 5. Care must be taken with any method that assumes a pair of pickup and delivery nodes, a common problem is that the IDs of the nodes can be simply i and $i + n$ where n is the number of customers, this is not the case with PMDP. A block or tree structure to represent LIFO consignments cannot be easily applied to PMDP due to the many potential positions that one consignment may interrupt another. There is no inherent reason that other models of PDP could not be extended to support the PMDP. Another potential approach could be to group all deliveries together in order to generate initial solutions using unmodified PDP solvers, the resulting solutions could be optimised with LSOs with awareness of PMDP.

In the following chapter the variable neighbourhood descent with memory (VNDM) hyper-heuristic to solve the PMDP is presented. Its performance is compared against alternative approaches and the state-of-the-art on static benchmark instances of the PDP.

5

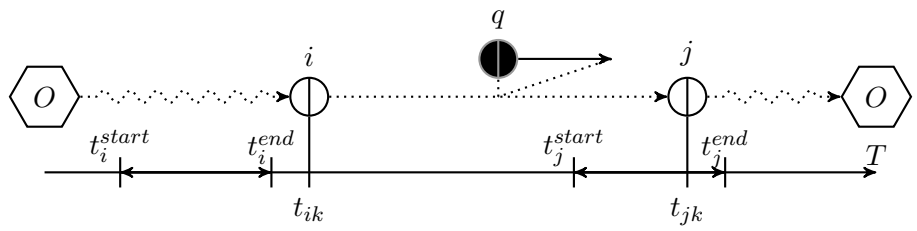
Solution Methods: Heuristics for PMDP

Commercial logistics scheduling in companies such as Transfaction Ltd.'s clients is generally done manually. Analysis of provided manually produced schedules shows that these are similar in quality to a greedy constructive heuristic. In this chapter we present a new variable neighbourhood descent with memory (VNDM) hyper-heuristic for the PMDP. VNDM is a two phase descent based local search, where new service requests are first greedily inserted into an existing schedule, before a first improvement local search is performed to improve the schedule. Due to the number of constraints in PMDP, the space of improving moves is greatly reduced in comparison to PDP, whilst the number of potential moves is still very large, resulting in a particularly difficult optimisation problem. A number of local search operators (LSOs, see Section 3.2.2) are created or adapted, that allow small changes to the existing schedule to be analysed and adopted if improvements are found. A "memory" of attempted moves is used to minimise repeated or redundant work. We show that our new VNDM algorithm is capable of producing substantial cost savings when compared to currently used scheduling methods, in acceptable time.

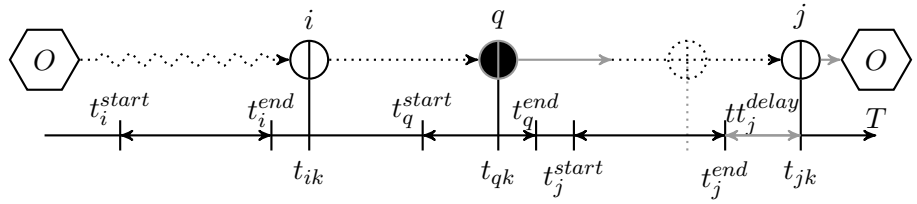
This chapter presents the LSOs designed and the VNDM optimisation method designed to produce good practical solutions for PMDP instances in order to enable the case study research presented in later chapters. I compare our VNDM against

approaches discussed in the literature review (Chapter 3) on static benchmark instances of the PDP. Comparisons are also presented to state-of-the-art PDP solvers.

5.1 Basics of Route Modification



(a) Before



(b) After

Fig. 5.1: An example of consignment insertion - Two consignments i and j are currently scheduled and only i is delayed. If another consignment (q) were inserted between the two existing consignments, i would remain at its current time while j may have to occur later, in this case becoming delayed by $t_{jk} - t_j^{end}$. For notation see figure 2.2.

Any change to a PMDP solution can be mapped to a series of insertion and removal operations. As the consignments themselves are present in both solutions (before and after any change) the only aspects that need to be considered are the legs between requests. However, when considering a modification, time and load constraints for each subsequent request must be checked. If the new load exceeds the truck capacity at any point, the new route is invalid. If any request becomes delayed, the cost of serving it may increase based on the delay cost function (Figure 5.1, see Section 4.4.3). Therefore, the cost of a route must be recalculated from the point at which it is modified, sub-total costs are saved for each request in each route to facilitate this.

5.2 Local Search Operators (LSOs)

In this section, we first state our assumptions and describe how our LSOs are used to alter routes when attempting to produce improved schedules. The LSOs are presented

under operators that work on single routes, and those that work on pairs of routes.

LSOs for the PDP have been drawn from similar problems (Bräysy, 2003; Cherkesly et al., 2015; Desaulniers et al., 2002; Gendreau et al., 1992; Savelsbergh, 1992), described in more detail in Section 3.2.2, and chosen to cover a wide range of potential variations from an existing schedule. Specifically we use: Exchange; Exchange Chain (Cross exchange Bräysy (2003)); Relocate and Relocate Chain operators that may act either within a route or between two routes. Exchange operators swap the positions of two (chains of) consignments while relocate operators move only a single (chain of) consignments. Chains have a fixed maximum length of 5 consignments to reduce computational complexity.

5.2.1 Assumptions

Our LSOs make or obey a number of important assumptions.

- Since a pickup request must occur before its delivery requests, reversing a section of a route will significantly alter the distance. Time windows are also usually tight enough that one or more requests would be rendered significantly delayed. Methods relying on partial route inversions such as GENI (Gendreau et al., 1992) and iCROSS (Bräysy, 2003) cannot work well without substantial alteration. We therefore use only the non sub-tour inverting CROSS exchange of Savelsbergh (1992) which is also used by Taillard et al. (1997), and our own variation of GENI, described below. Additional LSOs have been chosen or developed to preserve ordering as much as possible.

- A consignment may only be moved if the target position in the new schedule results in a valid schedule, i.e. one satisfying the constraints described in the problem definition (Section 4.5).

- A "valid location" is any pickup, delivery or base event that has not already happened, such that the insertion of a consignment at any point does not result in the creation of an invalid schedule. The rationale for this assumption is that, in dynamic problems, many potential operations become impossible as simulated time passes; consignments which could have been moved to another time or vehicle

are serviced according to the plan at the time.

- When a consignment is removed from a schedule, any requests nested within it are not removed, except in the case of cross exchange where more than one consignment may fall within the chain. This provides a means for the algorithm to undo nested consignments, if it can provide an improvement elsewhere.

5.2.2 Single Route Operators

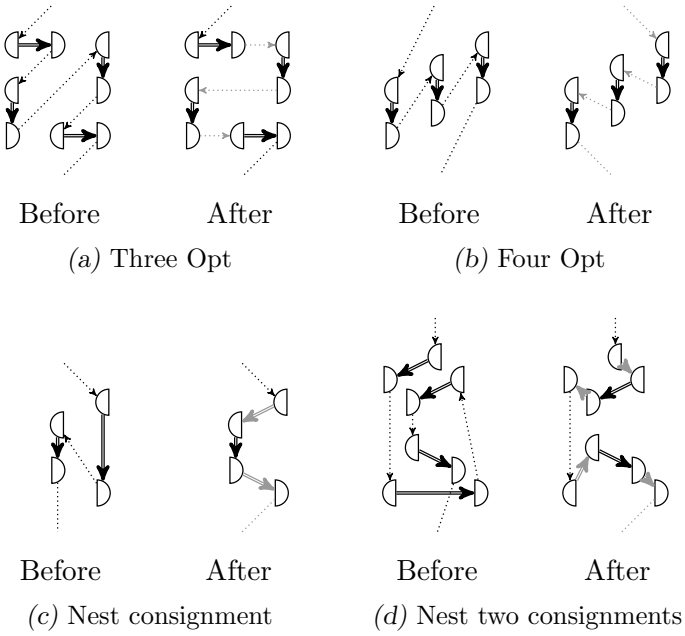


Fig. 5.2: Single route operators.

For a single given route, four potential LSOs are used, if an operator can generate more than one permutation for given inputs, the least disruptive to sub-tour ordering is used. Sub-tour inversion is not allowed.

Three Opt moves one consignment to a new position (Figure 5.2a).

Four Opt swaps two consignments positions (Figure 5.2b).

Nest Consignment moves a consignment inside another (Figure 5.2c).

Nest Two Consignments moves two consignments inside two other consignments. This is useful, as sometimes a single nesting produces no improvement and a first improvement selection strategy is employed (Figure 5.2d).

5.2.3 Dual Route Operators

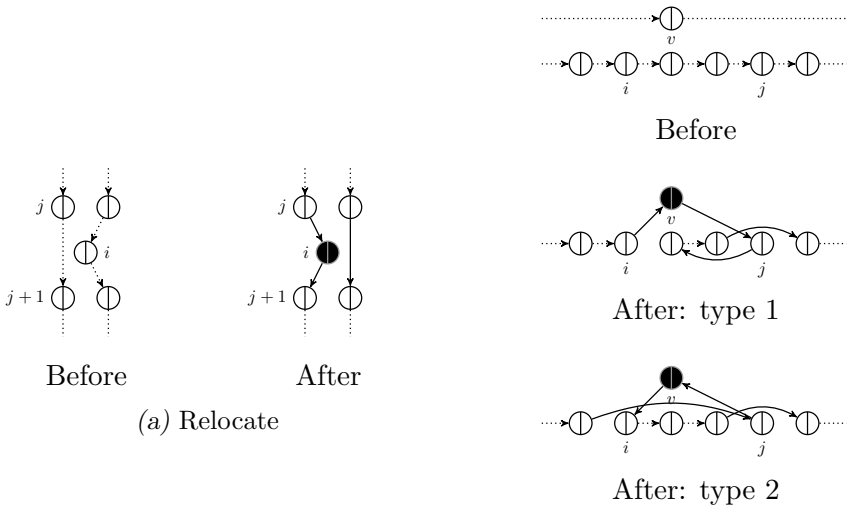


Fig. 5.3: Dual route relocate operators.

Considering multiple routes, four additional LSOs are used. With the exception of GENI by Gendreau et al. (1992) these local search operators were originally proposed by Savelsbergh (1992).

Relocate (Bräysy and Gendreau, 2005a) moves one consignment to a new valid location in a different route (this may introduce nesting) (Figure 5.3a).

Geni-PO is a variation of relocate that should generate better routes. It is modified (from GENI (Gendreau et al., 1992)) to preserve as much previous ordering as possible. Originally this would connect the new consignment to two of its p closest consignments in the new route. However we currently check all possible insertion location pairs (the removed consignment may have previously been nested) (Figure 5.3b).

Swap (Bräysy and Gendreau, 2005a) exchanges two consignments in different routes, (removing locally nested consignments) (Figure 5.4a).

Cross (Taillard et al., 1997) exchanges two chains of consignments between routes, preserving the existing ordering within each chain. Cross considers all possible chains

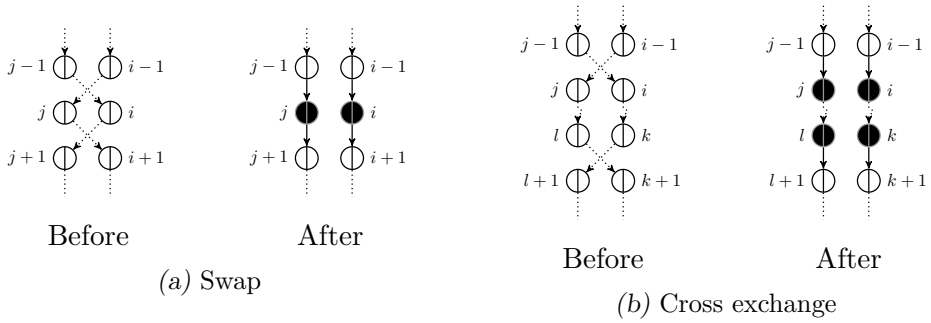


Fig. 5.4: Dual route swap operators.

from each route. The chains' maximal length is bounded only by the working hour rules imposed on each route (nested consignments are preserved) (Figure 5.4b).

5.2.4 Shake Operators

A shake operator makes a large, randomly chosen, local search move to the current solution, and is used once no LSO produces any further improvement. It could be thought of as a random re-start of the search from a different initial solution, or it could be a more extreme example of a shake operation that may be used in VNS. For the PMDP, this step consists of removing a random number of routes (drawn uniformly between 1 and the number of routes in the solution) and a random number of additional customers (drawn between 1 and the number of customers left in the solution). The combined list of all removed customers is then re-inserted into the remaining routes, creating new additional routes if required. The scale of the destruction of the original solution normally results in a substantially different solution from which to restart the search, though we do not guarantee it has not already been visited.

5.3 Variable Neighbourhood Descent with Memory

VNDM (Algorithm 5.3.1) is a variant of VNS with a strong bias towards exploitation, appropriate to large problems that must be solved quickly. The LSOs introduced in Section 5.2 are used in a first improvement descent strategy described in Algorithm 5.3.2. Alternative strategies include "best improvement" which requires full enumeration of all possible moves at every step and is very time consuming, "always accept" which

leads to very bad solutions, as most changes will either make no improvement or make the schedule worse. Another approach "SA accept", in which moves are accepted with a percentage chance that reduces over time at a specified cooling rate, was also considered. In preliminary testing SA accept yielded very poor performance as this cooling rate must be tuned to specific problem instances. This works well for static problems that can be solved repeatedly to tune this parameter but is not suitable for dynamic real-world problems where the optimal cooling temperature may be constantly changing. The LSOs are used in a hierarchical ordering presented in Section 5.3.2. Shaking (Section 5.2.4) is only performed once no LSOs are capable of producing improving moves, in contrast to traditional VNS where it is used at every neighbourhood.

Algorithm 5.3.1 VNDM

Precondition: Memory *db* storing route and LSO IDs

```
1:  function VNDM(Schedule s)
2:      s* ← s
3:      repeat
4:          for all l in LSOs                          ▷ See Section 5.2.
5:              repeat
6:                  First Improvement (s, db, l)       ▷ See Algorithm 5.3.2.
7:                  if found improvement
8:                      Update s with improvement
9:                      if s better than s*
10:                         s* ← s
11:             until no improvement for l
12:         Shake (s, db)                              ▷ See Section 5.2.4.
13:     until current time ≥ time limit
14:     return s*
```

In Algorithm 5.3.2, routeList rl contains all k-element subsets of route ids in the schedule where k is the number of routes required by LSO l. rl is ordered by the summed cost of the routes in each k-element subset, descending so that "worse" routes are considered for modification first (lower cost is better). *rl.GetNextRoutes(l,s)* returns the selected routes sr (the first k-element subset in rl). This is then removed from rl so it is only chosen once per LSO l.

The structure of routeList tl depends on the characteristics of an LSO m. If m requires only one route, tl is structured as a list of routes ordered by cost descending

Algorithm 5.3.2 First Improvement

Precondition: RouteList rl sorted by fitness descending

```
 1: function First Improvement(Schedule s, Memory db, LSO l)
 2:     repeat
 3:         selectedRoutes sr ← rl.GetNextRoutes(l,s)
 4:         if (sr & l) not in db
 5:             moves ← l.GetMoves(sr)
 6:             for all move m in moves
 7:                 fitnessDelta d ← l.Test(m)
 8:                 if d < 0                                  ▷ Improvement found.
 9:                     db.Remove(sr)              ▷ Other improvements now possible.
10:                     return sr, m & d
11:             db.Add(sr, l)                  ▷ No improvement found, add to memory.
12:         else
13:             do nothing                     ▷ Tried before and found no improvement.
14:         rl.Remove(sr)
15:     until no more routes
16:     return null                                ▷ No improvement possible using l.
```

(lower is better). If m requires more than one route, tl contains all k-element subsets of K (${}^nC_k$) where n is the number of routes in K and k is the number of routes required by move m. tl is ordered by the combined cost of each k-element set of routes, again descending. *tl.GetNextroutes(m)* returns the first route or routelist in tl. This is then removed from tl so it is only checked once per LSO m.

l.GetMoves(sr) generates all potential moves M for a given LSO l on sr. *l.Test(m)* generates the difference in fitness for a given move $m \in M$ with LSO l. If the fitness delta d is less than 0 (an improvement), the selected routes, move and fitness delta are returned. The move is then applied to the schedule, updating fitnesses for individual routes as appropriate. The memory is updated to remove the altered routes from all LSOs as this move may have enabled changes that were not previously possible.

5.3.1 Memory

VNDM stores information on which routes have been analysed by which LSOs, items are removed from this list when the related routes are changed as new improvements may now be possible. This hybridisation is inspired by the related reVNTS (Paraskevopoulos et al., 2008) and HVNTS (Belhaiza et al., 2013). In contrast, HVNTS stores recently seen solutions and distinguishes between large and small moves in its

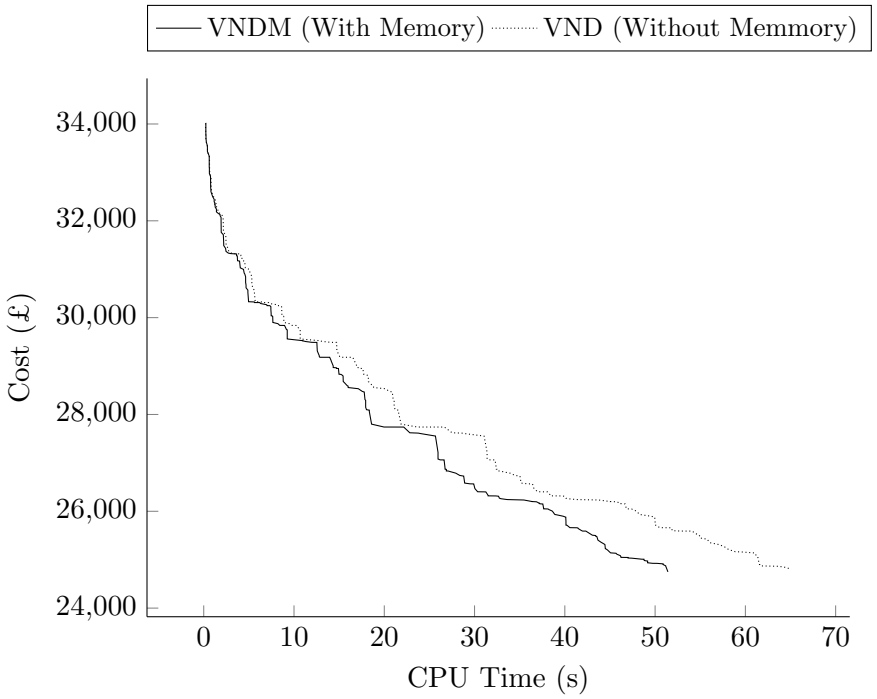


Fig. 5.5: Effect of memory on a static PMDP with 200 consignments.

neighbourhood structure. ReVNTS uses tabu search to find a local optimum within each neighbourhood of a VNS; additional features are learnt to control the use of LSOs. Both of these sources found that adding memory to traditional VNS improves performance. In VNDM, the memory structure has no impact on the logical operation of the algorithm, it merely enables substantially more iterations of the algorithm to be completed in the same amount of time.

An example comparison of with and without memory approaches is presented in Figure 5.5. Here we use a static PMDP of 200 consignments (drawn at random from our UK hauliers data set, Section 6.1) as we can quickly find near optimal solutions for this size problem. Using VND takes 65 seconds to reach a cost of around £25,000. When we use VNDM, a similar cost can be achieved in 52 seconds, a 20% reduction in CPU time.

5.3.2 Use of LSOs

VNDM's use of LSOs is deterministic; given an ordering of LSOs, VNDM repeatedly uses the first LSO until it stops finding improvements. The next LSO is then selected and used in the same manner. This process repeats until all LSOs have been used, the solution is then re-initialised using one of the shake operators described above. The

shake procedure will likely make the solution worse but offer a new position from which to search the space of possible solutions, helping escape local optima.

The LSOs described in Section 5.2 can be split into two groups based on their average run time:

1. Easy: 3-Opt, 4-Opt and Nest consignment
2. Hard: Nest two consignments, Relocate, Geni, Swap, Cross

The hard moves generate several orders of magnitude more potential moves than the easy moves. Instead of placing them all in the VNDM hierarchy of moves, at each call to VNDM one move is chosen at random from the five hard moves, such that our neighbourhood structure is:

$$\text{3-Opt} \rightarrow \text{4-Opt} \rightarrow \text{Nest consgt.} \rightarrow \text{Random Hard LSO} \tag{5.1}$$

This follows convention from VNS of moving from simplest to most complicated neighbourhood, the reasoning is that after making any change the HH jumps back to the simplest neighbourhood that could yield an improvement. Ordering LSOs by size is a convention proposed by Hansen et al. (2009). This neighbourhood structure is used because a time limit is imposed on each optimisation step; without this the algorithm may exceed its time limit if a hard LSO is chosen early in the sequence, and may only rarely attempt other hard LSOs. Since there is no intuitive reason to prefer one hard LSO to another, choosing one uniformly at random each time VNDM is called should ensure that all hard LSOs are used frequently, and provides ample diversification. Any hard LSO used individually does not produce enough diversification to produce good results.

5.4 Comparing Heuristics for the PDP

VNDM is compared to a number of techniques introduced in the literature review: Random Descent (RD, Section 3.2.5.1); Binary exponential back off (BEBO, Section 3.2.5.8); Q-learning (QL, Section 3.2.5.7) and the state-of-the-art. Since no benchmarks exist for the PMDP, to compare our methods to existing approaches, we use the Li and

Lim (2003) benchmarks for the static PDP. This problem is simpler and more abstract than the PMDP having single deliveries, no LIFO constraint and unitary time, distance and speed (A vehicle moves at 1 unit distance per 1 unit time). We first perform parameter tuning (Section 5.4.1) and secondly compare the performance of the four methods on all 100, 200 and 400 customer instances of the static Li and Lim (2003) benchmarks (Section 5.4.2). In these comparisons we consider hard time windows and no LIFO constraint such that direct comparisons may be made to best known solutions.

The solution methods mentioned above share the same LSOs, are coded in single threaded C♯ and distributed over a heterogeneous cluster of Intel Xeon based servers totalling 72 cores and 120GB of RAM. All methods are given 5, 10 or 20 minutes of CPU time based on problem size (100, 200 or 400 customers, respectively) and each is repeated 10 times. The results presented in this section thus represent over 1200 hours of CPU time.

5.4.1 Parameter Tuning

Each method is tested using three sets of parameters. For RD there is only one parameter, the number of iterations before re-initialisation. In RD 1, 2 and 3 the number of iterations is set at 250, 500 and 1000 respectively. The parameter we modify for BEBO controls the number of LSOs that are backed off at any stage of the search. The three values chosen for BEBO 1, 2 and 3 represent backing off all but the best solution, or all solutions more than 5% or 10% worse than the best solution (see Section 3.2.5.8, page 64). VNDM has two parameters, controlling the number of iterations (a) without improvement before re-initialisation and (b) before reverting to the previous best solution. These are set as (50, 125), (100, 250) and (200, 500) respectively for VNDM 1, 2 and 3. For QL, we investigate changes to the learning function (Equation 3.8, page 64) by setting α and γ to (0.25, 0.75), (0.5, 0.5) and (0.75, 0.25) respectively for QL 1, 2 and 3.

The performance of the heuristics is ranked for each instance, using their best result over 10 runs, and are given a score from one (best) to twelve (worst) where ties score equally low ranks. Figure 5.6 shows the average rank for each of the twelve methods for the 100, 200 and 400 customer instances. The 100 customer instances have a generally

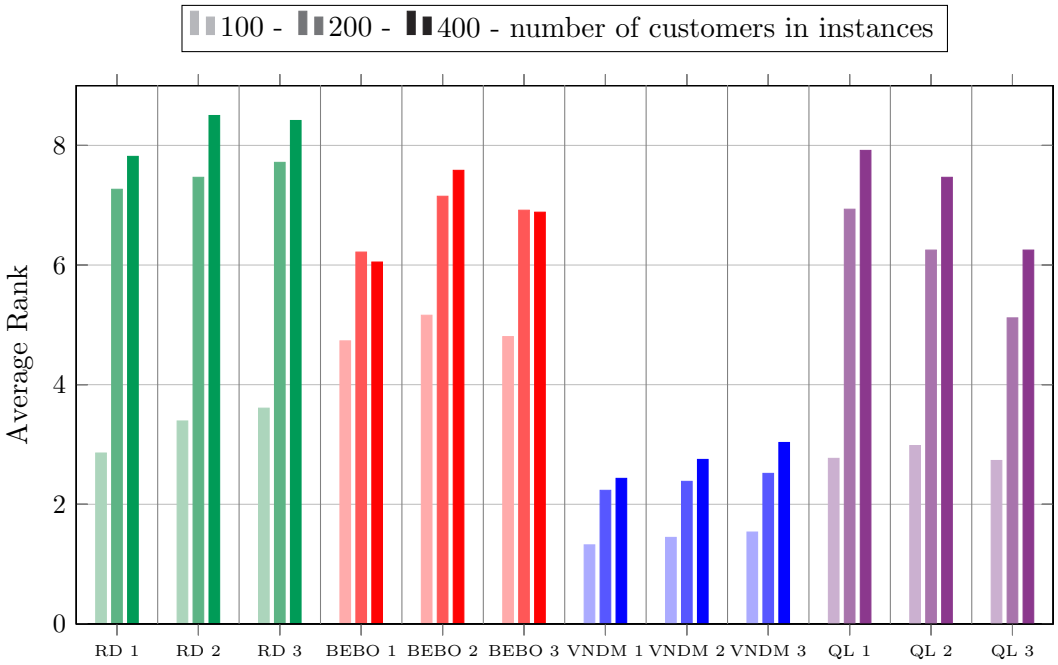


Fig. 5.6: Average rank of heuristics on different sized problems, lower is better.

lower average rank because, in many of these cases the best known solution is found by several methods which are then given equal rank one. An interesting point is that BEBO is weaker on smaller data sets but better on larger data sets compared to both QL and RD, suggesting that the overhead involved in trialling sets of LSOs is not worth the effort on small instances. VNDM consistently outperforms its competitors whilst demonstrating robustness to changes in parameter settings.

5.4.2 Comparison on Benchmark Instances

The Li and Lim (2003) benchmarks are split into three groups characterised by the spatial characteristics of the problem: Random instances (LRx-y-z) have customer locations that are spread uniformly randomly across space; clustered instances (LCx-y-z) have customer locations that are tightly grouped into a number of distinct clusters; and mixed instances (LRCx-y-z) have a mix of both random and clustered locations. For each instance, x can be either 1 (tight time windows) or 2 (lax time windows); y represents the number of customers in the instance (divided by 100) and z is the instance id.

Presented in Tables 5.1 and 5.2 are the best results of 10 repeats for each heuristic,

Tab. 5.1: Results for 100 clustered customer benchmarks. For explanation see text.

	RD		BEBO		VNDM		QL		Gap
Name	r	d	r	d	r	d	r	d	
LC1-1-1	**10**	**828.94**	**10**	**828.94**	**10**	**828.94**	**10**	**828.94**	0%
LC1-1-2	**10**	**828.94**	**10**	**828.94**	**10**	**828.94**	**10**	**828.94**	0%
LC1-1-3	9	1072.83	9	1082.18	**9**	**1038.35**	9	1048.40	0.29%
LC1-1-4	9	904.10	9	993.98	**9**	**861.95**	9	876.88	0.23%
LC1-1-5	**10**	**828.94**	**10**	**828.94**	**10**	**828.94**	**10**	**828.94**	0%
LC1-1-6	**10**	**828.94**	**10**	**828.94**	**10**	**828.94**	**10**	**828.94**	0%
LC1-1-7	**10**	**828.94**	**10**	**828.94**	**10**	**828.94**	**10**	**828.94**	0%
LC1-1-8	**10**	**826.44**	**10**	**826.44**	**10**	**826.44**	**10**	**826.44**	0%
LC1-1-9	**10**	**827.82**	10	882.86	**10**	**827.82**	**10**	**827.82**	1r
LC2-1-1	**3**	**591.56**	**3**	**591.56**	**3**	**591.56**	**3**	**591.56**	0%
LC2-1-2	**3**	**591.56**	**3**	**591.56**	**3**	**591.56**	**3**	**591.56**	0%
LC2-1-3	**3**	**591.17**	3	772.52	**3**	**591.17**	**3**	**591.17**	0%
LC2-1-4	3	676.03	3	614.65	**3**	**590.60**	3	652.95	0%
LC2-1-5	**3**	**588.88**	**3**	**588.88**	**3**	**588.88**	**3**	**588.88**	0%
LC2-1-6	**3**	**588.49**	**3**	**588.49**	**3**	**588.49**	**3**	**588.49**	0%
LC2-1-7	**3**	**588.29**	3	606.10	**3**	**588.29**	**3**	**588.29**	0%
LC2-1-8	3	591.39	3	594.69	**3**	**588.32**	**3**	**588.32**	0%

obtained using the best performing parameters identified in Section 5.4.1, for the 100 and 400 customer clustered instances respectively. For each method, the r and d columns are the number of routes and distance for the best observed run. The gap column records the difference between our best solution and the state-of-the-art for PDP solvers, as reported by Sintef (2008). The best known solutions are from a variety of sources: Bent and Van Hentenryck (2003); Blocho (2015); Hasle et al. (2007); Hosny (2010); Koning (2011); Li and Lim (2003); Ropke and Pisinger (2005); Quintiq (2015); TetraSoft (2003). Where our best solution has the same number of routes but is longer, the gap records this difference as a percentage. Where our solution has n extra routes, the gap is nr. We highlight in **bold** the best solutions found for each instance and highlight the instance name in **bold** where we match the best known solution from the literature. Overall we find the best known solution in 54 out of 153 benchmarks[1].

The four hyper-heuristics perform very similarly on the 100 customer instances, finding the best known solutions in many cases. In cases where the best known solution is not found, VNDM is the best or joint best of the methods tested and produces results within 0.3% or 1 route of the best known solution

Table 5.2 shows the results for the 400 customer clustered instances. It is clear that VNDM is the strongest method we compare, matching best known solutions in many

[1] Full results are available in the appendix, Tables A.1 to A.7.

Tab. 5.2: Results for 400 clustered customer benchmarks. For explanation see text.

	RD		BEBO		VNDM		QL		Gap
Name	r	d	r	d	r	d	r	d	
LC1-4-1	**40**	**7152.06**	40	7208.31	**40**	**7152.06**	**40**	**7152.06**	0%
LC1-4-2	40	7184.42	40	7491.64	**40**	**7170.60**	40	7235.69	2r
LC1-4-3	37	8089.33	37	8684.88	**37**	**7871.19**	37	8383.98	4r
LC1-4-4	32	8328.28	32	8544.82	**32**	**7403.17**	32	7748.67	2r
LC1-4-5	**40**	**7150.00**	**40**	**7150.00**	**40**	**7150.00**	**40**	**7150.00**	0%
LC1-4-6	**40**	**7154.02**	40	7237.16	**40**	**7154.02**	40	7170.01	0%
LC1-4-7	41	7542.55	42	8734.36	**40**	**7149.44**	41	7435.92	0%
LC1-4-8	**39**	**7111.16**	40	7706.57	39	7179.98	**39**	**7111.16**	0%
LC1-4-9	38	8197.97	38	8390.38	**37**	**7819.79**	39	8479.35	1r
LC1-4-10	38	7940.02	37	8016.53	**37**	**7670.50**	37	7990.68	2r
LC2-4-1	14	6824.82	**12**	**4116.33**	**12**	**4116.33**	13	5444.85	0%
LC2-4-2	14	9135.06	13	5108.89	**13**	**4844.74**	14	7999.00	1r
LC2-4-3	13	7145.52	13	5967.34	**12**	**5364.88**	13	6375.44	1r
LC2-4-4	13	7727.34	12	6193.68	**12**	**5766.83**	13	7311.49	35%
LC2-4-5	15	8612.69	13	5243.16	**13**	**4717.13**	14	6886.47	1r
LC2-4-6	14	7560.98	13	4936.46	**13**	**4721.75**	14	7125.17	1r
LC2-4-7	14	8312.98	14	5882.84	**13**	**4616.22**	14	7542.35	2r
LC2-4-8	14	7883.71	13	5456.19	**13**	**4523.78**	14	7582.07	1r
LC2-4-9	14	7770.37	13	6334.76	**13**	**5419.32**	14	7450.32	1r
LC2-4-10	14	7867.45	**13**	**4655.07**	13	4737.62	13	6330.36	1r

cases. All our methods have difficulties with the LC2 data set due to looser time constraints resulting in a much larger quantity of feasible solutions. VNDM still produces results that either have fewer routes or, for instances with the same number of routes, solutions that are on average around 10% shorter than the other three approaches. However, VNDM often produces more routes in comparison to best known solutions.

Closer investigation of the results for various instances shows that a wide array of solutions is created, each of which is subtly different. Since all our heuristics are based on first improvement and have ample time to converge, this can be attributed not to time constraints but to the nature of the problem itself. The solution landscape for PDPs is not smooth and contains many local optima, making it difficult for heuristics to converge on the same result. To visualise this, we plot all of our results in two dimensions, showing the number of routes against schedule distance. Figure 5.7 presents solution space maps generated in this way for four representative instances.

Figure 5.7 highlights an interesting finding in the benchmark data sets. In the clustered (LC) data sets there is a clear trend between the number of routes and the total distance of a solution. In the random data set, however, these aspects are not closely correlated, so using the total number of routes as the main objective does not

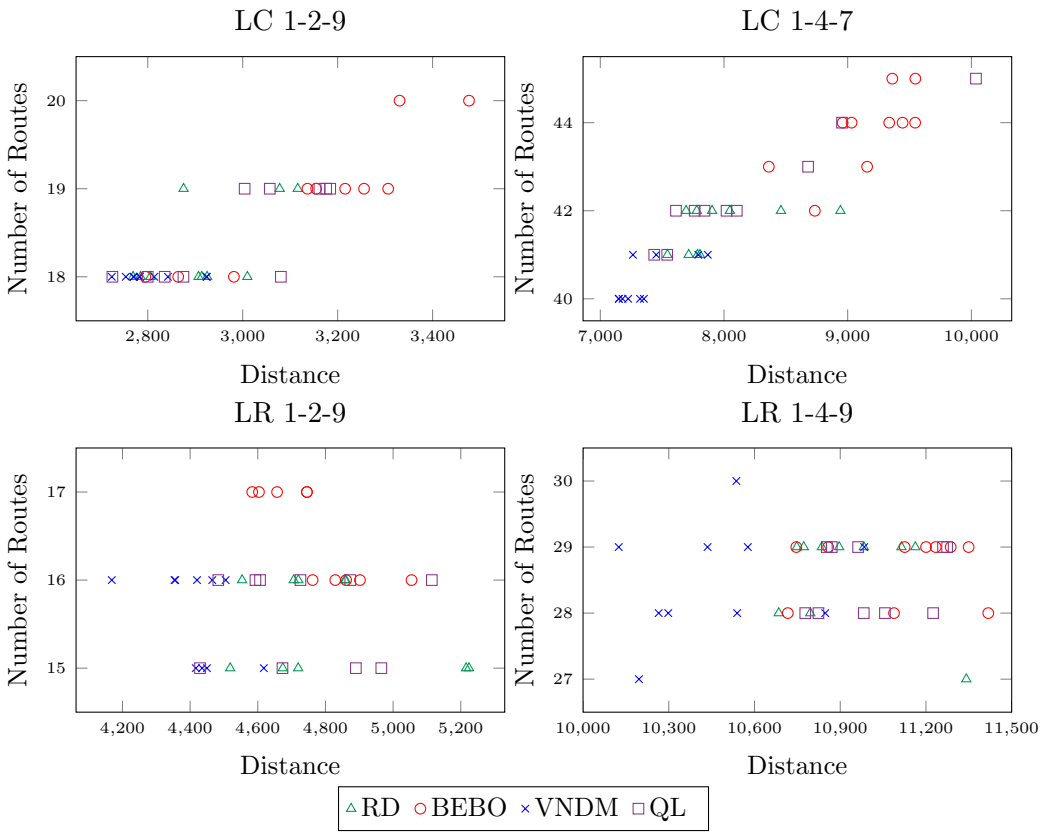


Fig. 5.7: Number of routes versus cost for clustered (LC 1-2-9 and LC 1-4-7) and random (LR 1-2-9 and LR 1-4-9) benchmark instances.

seem to be appropriate. We note also that, as expected, the distance is greater in the random scenarios, however, the number of routes is lower, probably due to the looser time window constraints and smaller service times in these problems. Koning (2011) notes that relaxing the hard time window constraints, applied in the Li and Lim (2003) benchmarks, produces notably shorter routes with only minor delays, which may be preferable in the real world.

5.5 Discussion and Summary

This chapter presents our LSOs and the VNDM hyper-heuristic used to solve instances of the PMDP, in relation to the objective and ordering constraints set out in Chapter 4. We introduce the VNDM hyper-heuristic to solve this problem and show it to be competitive with the state-of-the-art for small benchmark PDP instances (Li and Lim, 2003; Sintef, 2008). We have shown that, in limited CPU time, VNDM outperforms BEBO, QL and RD on many of the 100, 200 and 400 customer static benchmark

instances. This result is shown to be robust to changes in parameter settings and is performing well enough to be used as a practical optimiser for PMDP to be used in our case-study research.

For many of the random and some of the clustered instances in the Li and Lim (2003) benchmarks, shorter solutions are possible if more routes are used. All the methods tested are shown to struggle on random instances of the PDP, performing best on clustered instances. Though domain specific knowledge is used to identify suitable LSOs for the VNDM, the majority of the method is portable across domains and has been used successfully on the periodic vehicle routing problem (Chen et al., 2016a).

Having analysed performance relative to the state-of-the-art for PDP benchmarks, in the next chapter I analyse Transfaction Ltd.'s data, present the process of cleaning the data, generate distributions that closely match the cleaned data and use these to produce additional data sets for the PMDP. I compare the approaches introduced in this chapter again, on dynamic data with our additional constraints.

6

Solution Methods: Real-world PMDP

Transfaction Ltd. has provided a pair of representative real-world data sets from a department store and from a number of retailers, all based in the UK. Section 6.1 introduces the first data set, a large number of consignments from UK retailers, the distributions that are apparent in this and how we clean and use this data in our experiments. Section 6.2 introduces a second data set, a smaller number of consignments from a UK department store chain, of higher quality for which we have access to the real schedule, enabling direct comparisons of our methods to those used in industry. Section 6.3 presents the discrete event simulation techniques that we use to simulate the real-time execution of a scheduling strategy in silico, in order to evaluate our solution and analyse the potential improvements to current scheduling practice. Finally, Section 6.4 presents a comparison of the HH methods introduced in Chapter 5 on dynamic, real-world instances of the PMDP.

6.1 UK Retailers Case Study

The UK retailers case study is a large real-world data set representing 3 large distributors and 220 haulage company "carriers". This data set comprises 27,153 deliveries between supplier and destination locations of retailers in the UK. However, we have insufficient consignment data for a full scale simulation (see Chapters 7 and 8), so we generate additional consignments from the data we have access to. This also allows us

to generate an infinite range of realistic problem instances with different characteristics.

Each generated consignment requires a location, load and a set of time windows (consignment-received time; scheduled pickup time; service times; journey times; scheduled delivery times and an arrival time window, as laid out in Section 4.2.1). This chapter investigates the real-world distributions for each of these factors.

6.1.1 Location, Linehauls and Backhauls

Our real-world data cover much of England, Wales and southern Scotland, as shown in Figure 6.1. The data represent "linehauls", primary deliveries of goods from manufacture to multiple points of sale. "Backhauls", carrying used packaging back to the shipper or depot, are not present in the real data; however we know that they exist as return, point to point, legs, originating at linehaul delivery locations and terminating at linehaul pickup locations. There are no precedence constraints on backhauls in relation to linehauls, though LIFO loading still applies. Backhauls are therefore generated by randomly choosing an existing delivery as the point at which a backhaul starts and routing to its corresponding pickup location. For new backhauls, we assign load and time values mined from our existing data.

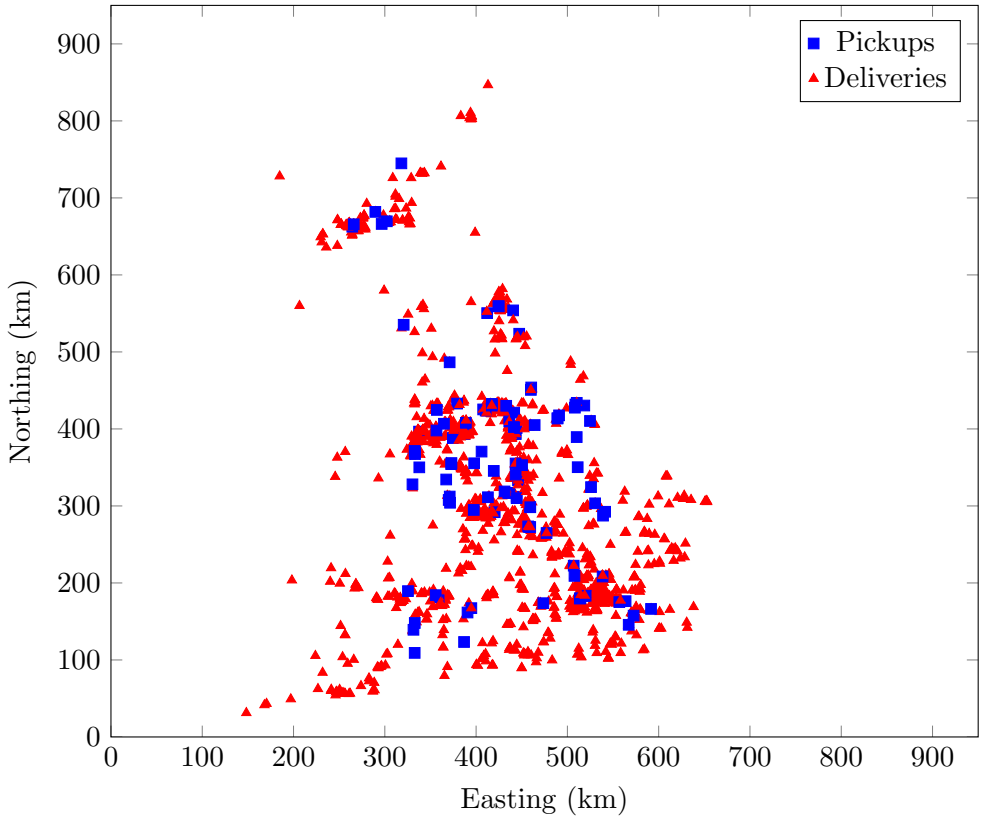


Fig. 6.1: Locations of real-world data, covering England, Wales and southern Scotland.

6.1.2 Load

There is no data regarding the loads of trucks in the Transfaction Ltd. data, but we do know whether each consignment is a linehaul or a backhaul since all backhauls have been generated. After discussion with Transfaction Ltd., linehaul load sizes (as a percentage of a full truck load) are drawn from a normal distribution with mean of 50% and standard deviation of 5%. Backhauls are set to a fixed 30%.

6.1.3 Time

Time information is often recorded by hand leading to errors in Transfaction Ltd.'s data set. Out of 27,153 consignment records, only 14,382 records (≈53%) include valid time information. Recorded times are often missing, filled with a place holder such as 00:00 or clearly impossible, such as having an arrival time before the previous departure. The following sections present the distributions observed in the 14,382 valid records and approximating functions used to generate missing information.

6.1.3.1 Consignment Received Time

The consignment-received time records the time when a consignment is created. From the available records, it is clear that the values in the real data are manually-entered, by people working a standard 09:00 to 17:00 day (see Figure 6.2). The majority of consignments are created before lunchtime with additional consignments created between this time and the end of the working day. I assume that these times are a fixed aspect of the system over which we have no control. Though each day follows approximately the same distribution, the number of consignments changes substantially through the course of a week, as shown in the Appendix, Figure A.5.

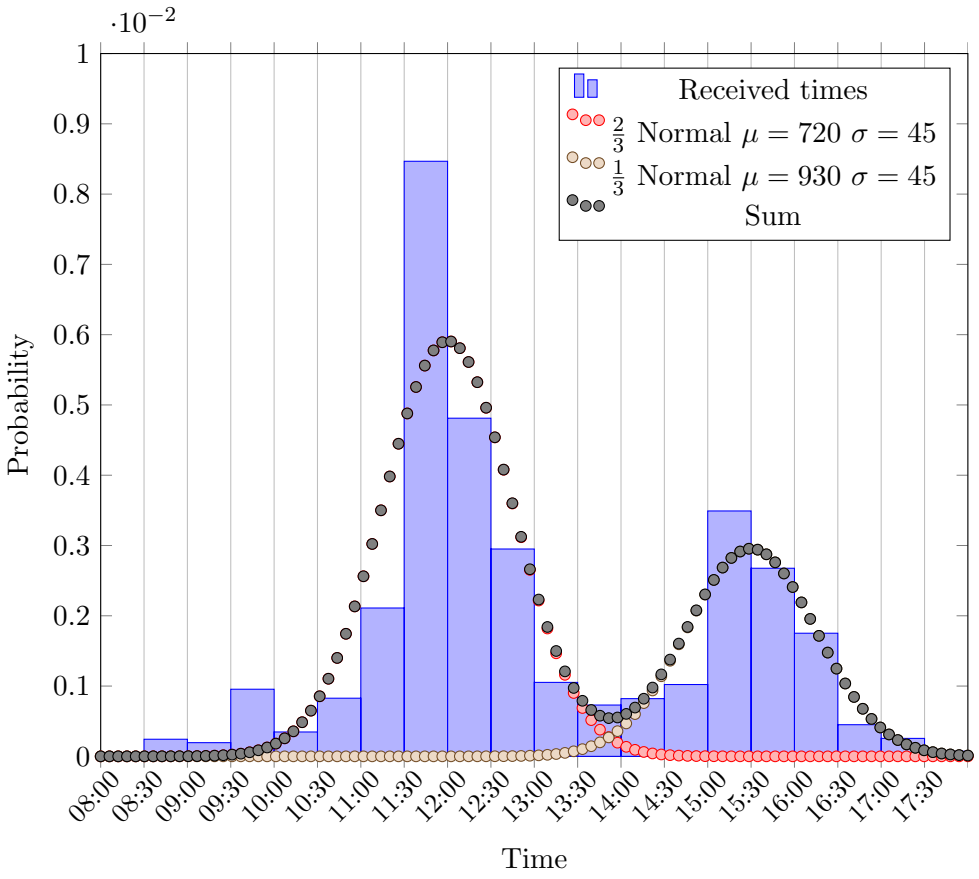


Fig. 6.2: Real-world consignment received times, for consignments where this information is present, and approximating distribution.

Many of the real data, and all generated backhauls, are missing consignment received values. For these consignments we have to generate an appropriate received time. To do this, we model the received time as the sum of two normal distributions centred around the observed peaks shown in Figure 6.2, the first has a mean of 720

minutes (12:00) and the second, 930 minutes (15:30). Both have standard deviations of 45 minutes. We weight the two distributions as $\frac{2}{3}$ of the first distribution, plus $\frac{1}{3}$ of the second. The result, shown in Figure 6.2, matches adequately the distribution of known consignment received times.

We must then decide on what day each consignment occurs. To model the distribution of received consignments over the course of a week the probabilities shown in Table 6.1 are used. An example output comparing 5480 real received times to 5480 generated received times is shown in the appendix, Figure A.5.

Tab. 6.1: Modelled probabilities for received day.

Day	Sun	Mon	Tue	Wed	Thu	Fri	Sat
Probability	$\frac{1}{105}$	$\frac{2}{21}$	$\frac{2}{21}$	$\frac{4}{21}$	$\frac{8}{21}$	$\frac{4}{21}$	$\frac{4}{105}$

6.1.3.2 Pickup Time

Figure 6.3 shows a relationship between the consignment-received time and the corresponding pickup-scheduled time for all of the consignments where this data is valid (14,382 records, ≈53%). We refer to the length of time between consignment-received and pickup-scheduled as the planning window. We observe spikes at 24, 48, 96 hours, relating to whole days of planning, though we can see a clear underlying trend. An inverse Gaussian approximating function is used when generating additional data as it produces planning windows that are similar in distribution to the original data. The smoothing of full day peaks would have the effect of spreading our data out slightly more evenly throughout a day; however looking at Figure 6.4, we see that pickups occur at all times of the day so this spreading is not thought to be an issue.

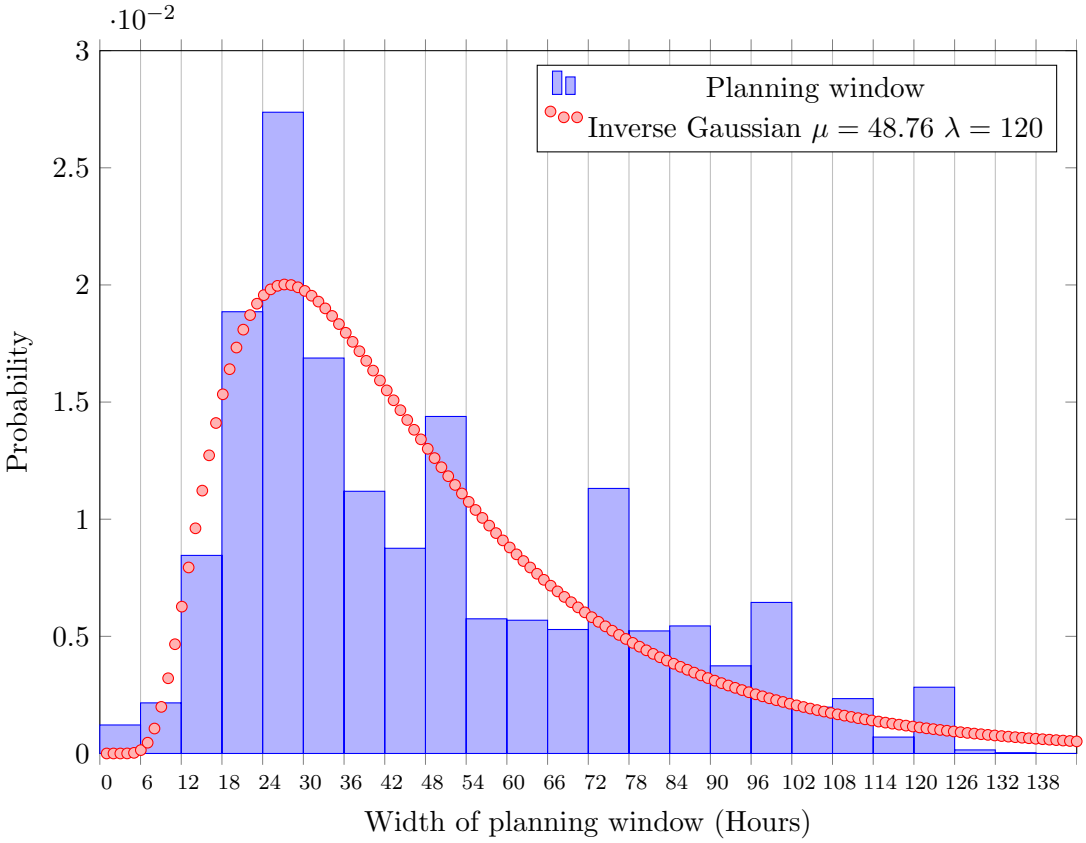


Fig. 6.3: Real-world planning windows (planned pickup time minus received time) and an approximating inverse Gaussian function.

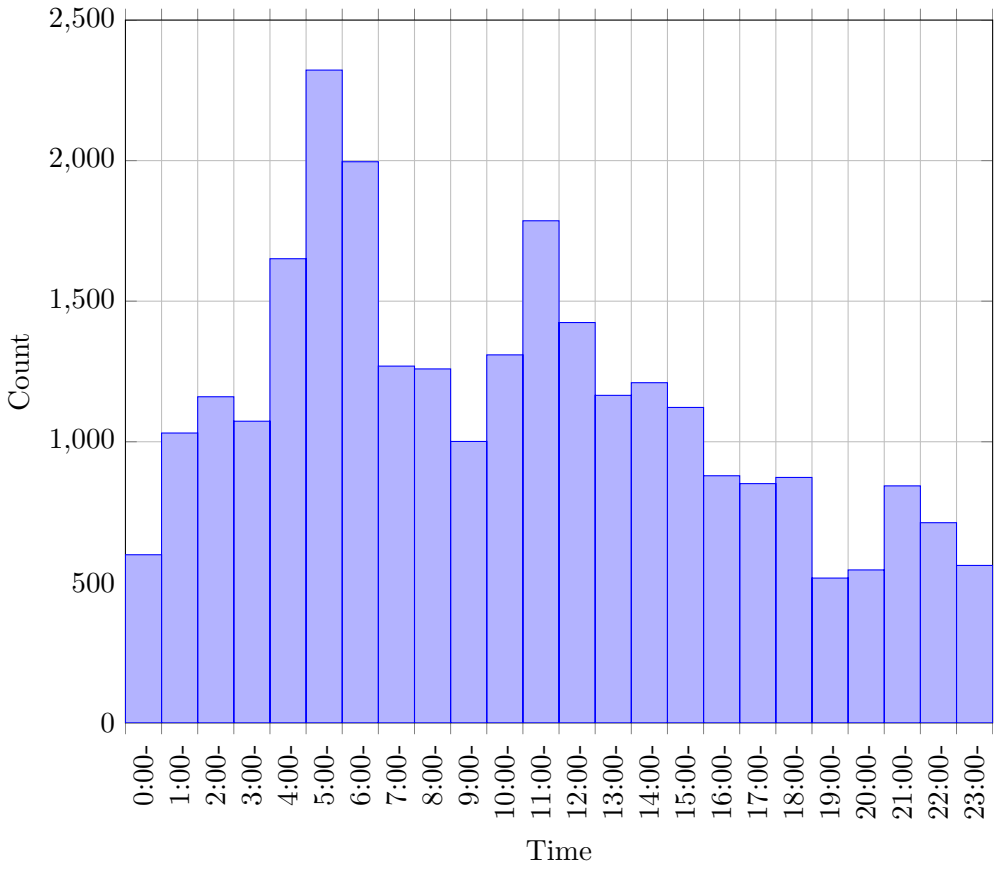


Fig. 6.4: Real-world distribution of pickup times.

6.1.3.3 Service Time

Figure 6.5 shows the valid pickup and delivery service times from the Transfaction Ltd. data set. We observe that in general, delivery (unloading) times are longer than pickup (loading) times. After observing that the distributions are most closely fit by the inverse Gaussian distribution we have chosen parameters for two different curves. The first, with mean 85 and shape parameter 210 fits the pickup loading times well, while a second inverse Gaussian, with mean 100 and shape parameter 300, provides a good fit for the delivery unloading time. Pickup and delivery service times are generated in accordance with these distributions for all backhauls and for linehauls for which data is missing.

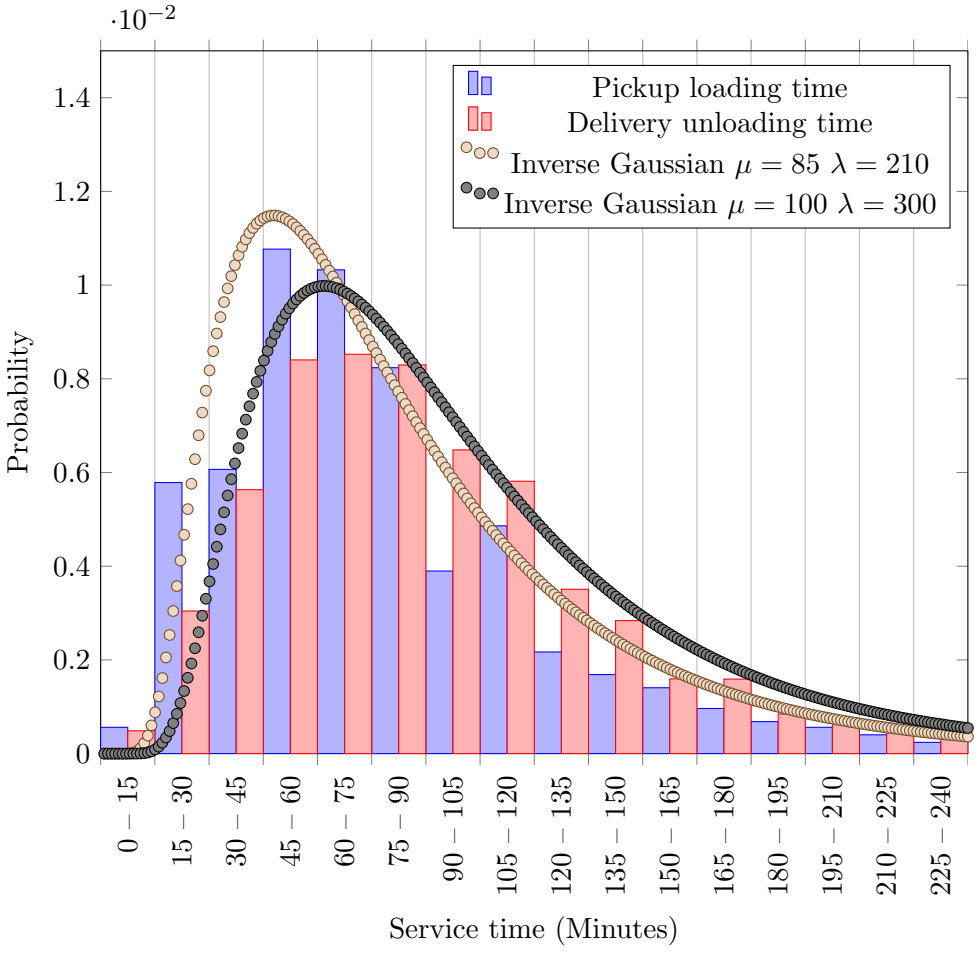


Fig. 6.5: Real-world pickup and delivery service times.

Figure 6.6 shows that there is no correlation in the real data between loading and unloading times. Therefore, it is unsafe to use the distributions to estimate the load of a consignment. Figures 6.5 and 6.6 both show (un)loading time values between 0 and 240 minutes; values outside of this range are present in the Transfaction Ltd. data but are considered to be inaccurate - notably those which have negative service times. In cases where we cannot trust the data, service times are drawn from the distributions shown in Figure 6.5.

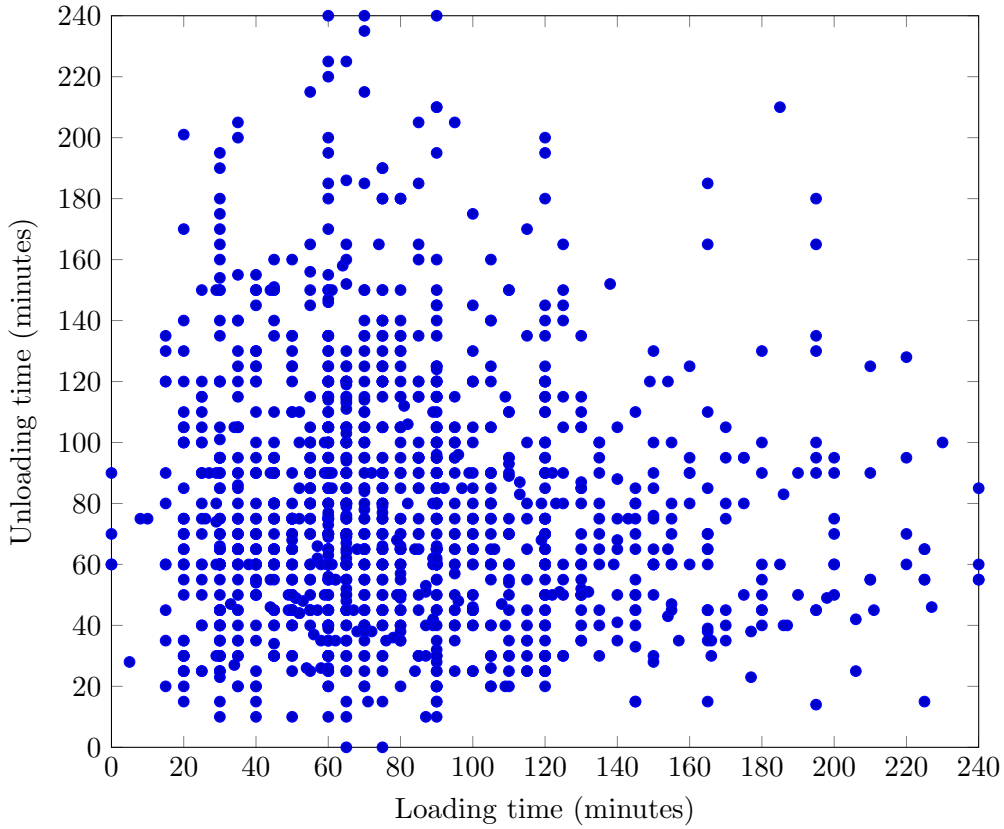


Fig. 6.6: Real-world loading vs unloading service times.

6.1.3.4 Journey Time

Given that companies such as Transfaction Ltd. have objectives with time based components, we have to be able to estimate the time taken to travel between any of the pickup and delivery locations in our data set. We plot journey time as a function of distance, for consignments which have the required information, in Figure 6.7. Large variabilities in journey times can be seen however, a positive correlation is clearly apparent. For our purposes, a best case average speed of 56km/h and straight line distances are assumed for all our experiments, since modelling real world traffic flows is outside the scope of this research. Since internally we map distance to time instead of using the distance directly, it would be possible to incorporate more accurate measures into this step. It should also be noted that there are some clearly erroneous points in this data set: the 90km/h line in Figure 6.7 is the European speed limit for 44-tonne articulated trucks; the data points to the left of this line represent arriving impossibly early.

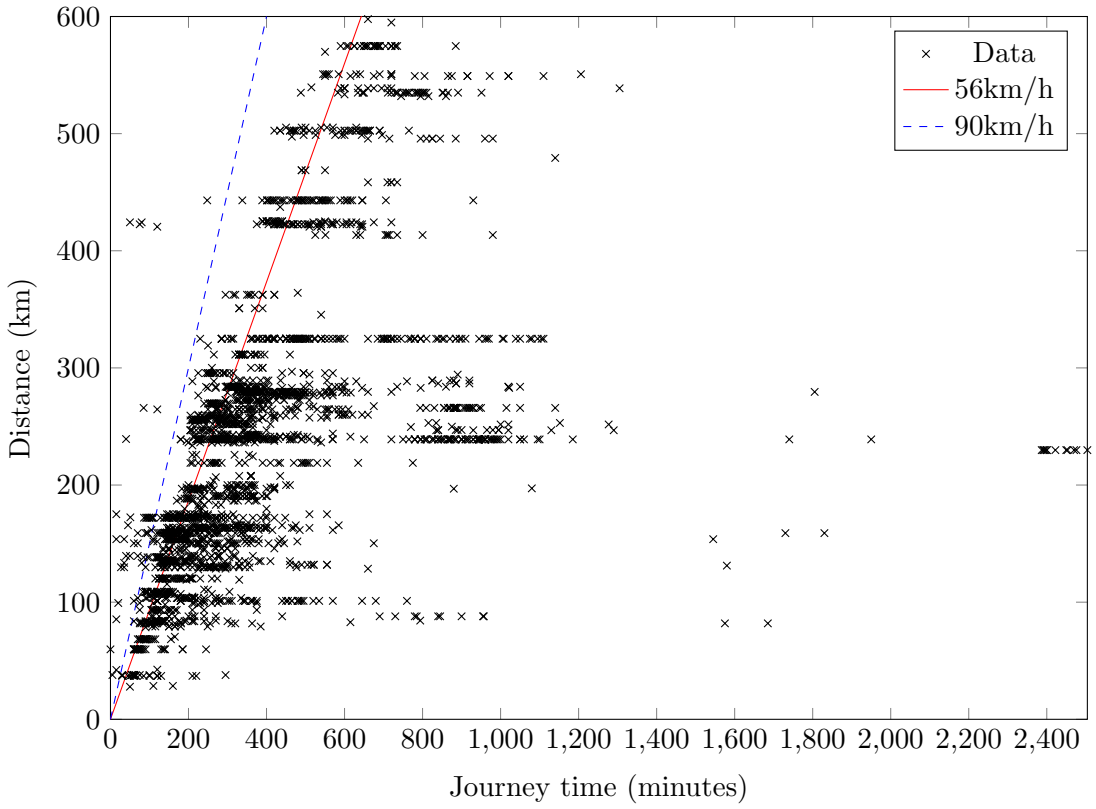


Fig. 6.7: Real-world time vs distance and approximating function.

6.1.3.5 Delivery Time

In the Transfaction Ltd. data set, there are too few delivery times for us to model and generate missing values. Instead of trying to emulate the real situation we take a pragmatic approach and assign delivery times on the assumption that we reach the pickup at the earliest possible time, then add to this the pickup service time and the journey time from pickup to delivery. The delivery time is calculated as shown in Figure 6.8, where the planning window is also shown.

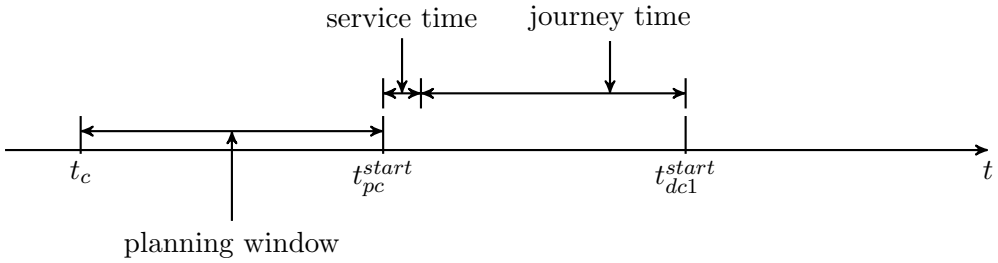


Fig. 6.8: Estimated delivery time t_{dc1}^{start}, set to the known pickup time t_{pc}^{start} plus the known pickup service time and the journey time from pc to $dc1$.

Subsequent delivery arrival window start times are calculated in a similar manner using the previous delivery's start, service and journey from times.

6.1.3.6 Arrival Time Window

The arrival time window is the period in which no waiting or delay penalties are incurred by a truck servicing either a pickup or delivery request (Section 4.4.3). The arrival time window is the same for all pickups and deliveries in a consignment, and is visualised in Figure 6.9

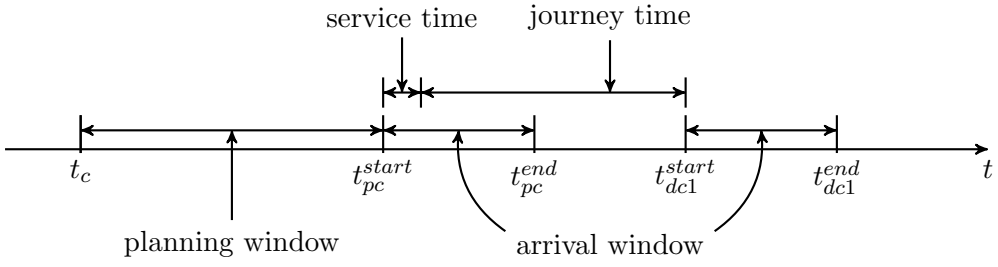


Fig. 6.9: Planning and arrival windows for a consignment c with one pickup pc and one delivery $dc1$. Labelling as described for Figure 6.8

6.1.3.7 Delays

A delay is the time between the end of the window in which a pickup or delivery is required and the actual time if the vehicle arrives late. Delay varies based on the schedule used and can be used as a measure to compare different schedules. Figure 6.10 shows the delays for pickup and delivery requests in the original, manual, schedule provided by Transfaction Ltd.. The plots ignore outliers that are delivered days before or after their deadline. In the real data, approximately half of pickups are delayed while a majority of deliveries arrive on time.

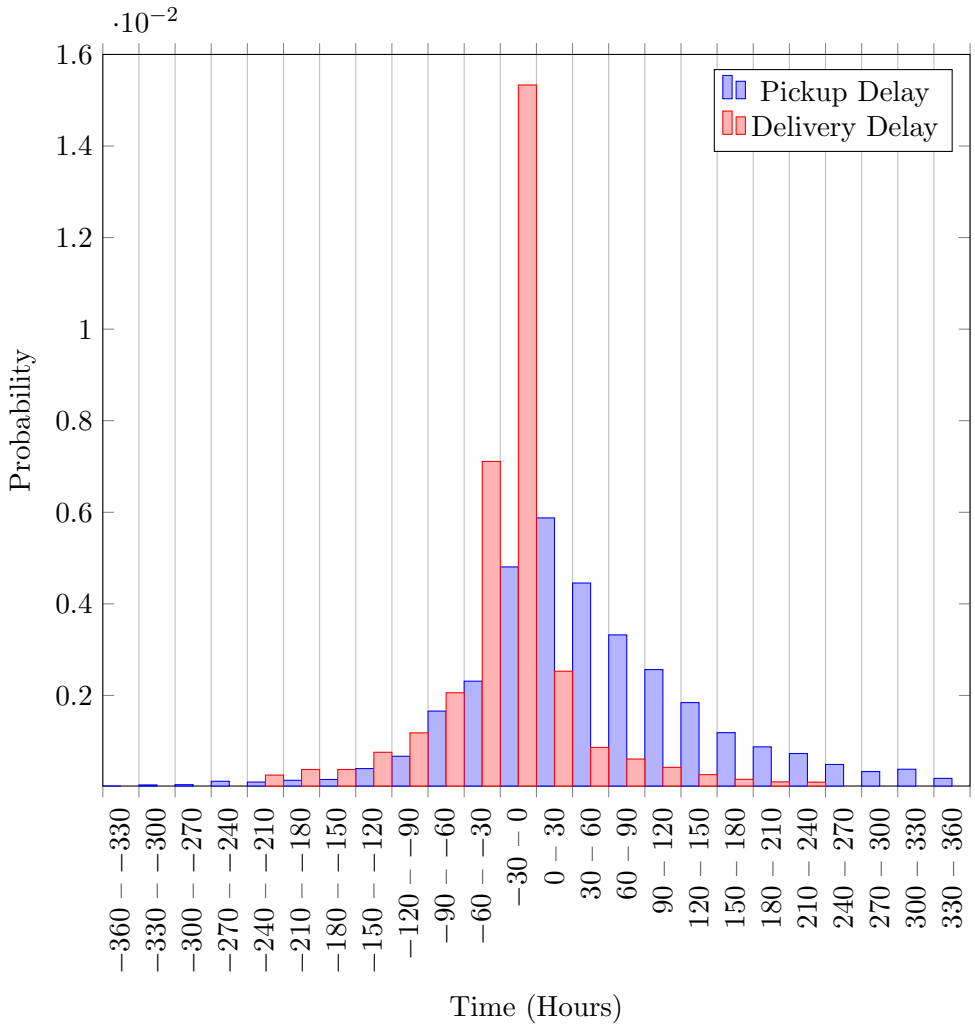


Fig. 6.10: Real-world pickup and delivery delay.

6.1.4 Consignment Generation Process

Pulling together our data cleaning and generation, Algorithm 6.1.1 presents the consignment generation process used for all data used in this thesis. In Algorithm 6.1.1, Uniform(x, y) produces a uniformly distributed random integer in the range x, y inclusive, and IGaussian(x, y) produces a real number with a probability based on an inverse Gaussian distribution with $\mu = x, \lambda = y$.

Algorithm 6.1.1 Data Preprocessing

```
1:  function CLEAN CONSIGNMENT DATA
2:      Consignment c
3:      c.start day ← Proportional Select                          ▷ Table 6.1
4:      c.received time ← 2/3(Normal(720, 45)) + 1/3(Normal(930, 45))
5:      c.p ← from data
6:      c.p.load ← Normal(50%, 5%)
7:      c.p.start time ← c.received time + IGaussian(48.76, 120)
8:      c.p.service time ← IGaussian(85, 210)
9:      c.D ← from data
10:     GenerateDeliveries(c)

11: function GENERATE BACKHAUL
12:     Consignment c
13:     c.start day ← Proportional Select                          ▷ Table 6.1
14:     c.received time ← 2/3(Normal(720, 45)) + 1/3(Normal(930, 45))
15:     c.p ← new pickup
16:     c.p.location ← Random Delivery Location
17:     c.p.load ← 30%
18:     c.p.start time ← c.received time + IGaussian(48.76, 120)
19:     c.p.service time ← IGaussian(85, 210)
20:     c.D ← new delivery                  ▷ only one delivery for backhauls
21:     c.D.location ← Corresponding Pickup Location                ▷ see text
22:     GenerateDeliveries(c)

23: function GENERATEDELIVERIES(Consignment c)
24:     Previous Event e = c.p
25:     for all Delivery d ∈ c.D
26:         journey time ← e.timeTo(d.location)
27:         d.start time ← e.start time + e.service time + journey time
28:         d.service time ← IGaussian(100, 300)
29:         d.load ← c.load / n
30:         e ← d
```

6.2 Department Store Case Study

The department store case study is a small real-world PMDP data set comprising 387 consignments spread over the south east of the UK. This data set represents a typical one week planning period for the department store chain. The location and time values for this data set are of high quality and are used directly. There is no load size information associated with this data set but we make assumptions based on the number of deliveries in each consignment. The distributions set out in Section 6.1.2 are used to assign a fictional load to each consignment.

The department store data set is especially useful as we have the real-world schedule to compare our results to. A summary of the existing schedule is presented in Chapter 8, Section 8.2.

6.3 A Dynamic Problem

In a competitive logistics company, scheduling needs to be a dynamic process, as discussed in Section 2.2.6, that happens in real time. A static model would not work competitively in a real-world problem where consignments arrive continuously throughout the working day and are expected to be scheduled quickly in order to generate a competitive quote. A static model would have to be either regenerated every time a new consignment enters the system, leading to difficulties matching the existing schedule to the new schedule or solved separately, sub-optimally. The dynamic model I have implemented is a two phase process. Phase one occurs when consignments enter the system, consignments are inserted greedily into an appropriate vehicles schedule on a first come first served basis. Phase two, optimisation, happens continuously while no new consignments arrive. Optimisation can take many forms, as discussed in the literature (Chapter 3) or as developed here (Section 5.3).

6.3.1 Discrete Event Simulation (DES)

DES (Pidd, 1998) is used to simulate the dynamic receipt of consignment requests, in silico, for testing purposes. To simulate a real consignment scheduling scenario, each consignment is treated as an independent event and must be scheduled with knowledge

only of earlier consignments. The order in which consignments are observed is based on their received time t_c. Once sorted into ascending order by received time, the first phase of the dynamic model can proceed, with consignments arriving throughout a simulated period of time.

After each insertion, a fixed amount of CPU time is used to conduct the optimisations of phase two. This is an acceptable simplification as the experiments run over a considerably shorter time period than in reality, where ample CPU time would be available between consignment arrivals.

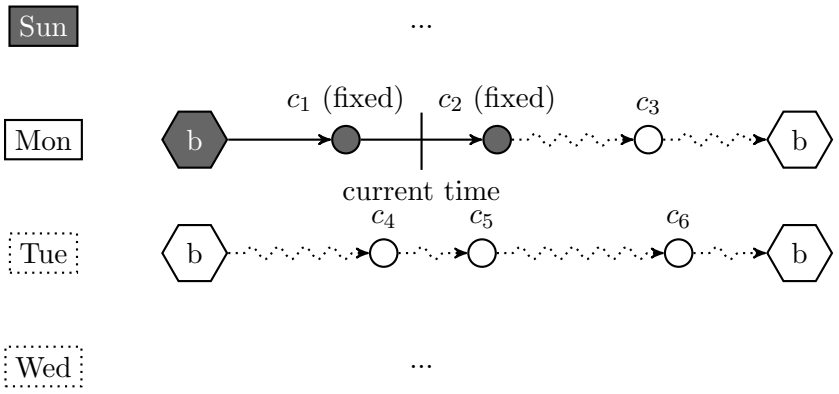


Fig. 6.11: An example daily schedule - Each truck's schedule is a list of events that may represent a pickup, delivery or return to base. Here, dark nodes are locked in place; new requests cannot be inserted before these and they cannot be moved during optimisation.

In DES, we keep track of simulation time (an internal representation of current time, stored so that requests which in reality would have already happened cannot be modified by our optimisation procedure). If the scheduled start time of any request is before the current simulation time, it is marked as "fixed", as shown in Figure 6.11. In reality, this would be a consignment that is already being serviced. Additional requests cannot be inserted before these fixed requests, and the routing of a fixed request cannot be altered in any optimising moves.

Once all consignments have been inserted and after a final pass of optimisation, the simulation is complete and performance data for the optimisation method or for the effect of test parameters can be gathered.

6.4 Comparing Heuristics for the PMDP

Our small real-world data set comprises 387 consignments pairs spread over the south east of the UK. The data provided by Transfaction Ltd. is from a large department store chain and comprises deliveries for a one-week period. Due to the distribution of store locations, a larger number of routes is required for full service, in comparison to similarly sized benchmark problems. In addition, for the real-world problem we simulate consignments arriving during scheduling in a dynamic fashion (Section 6.3), and allow late arrival at locations using "soft" end of time windows. We use a cost based fitness function (described in Chapter 4) as the objective rather than the number of routes and distance.

We compare VNDM, QL, BEBO and RD as before, comparisons to the state-of-the-art for PDP cannot be made on our PMDP data without re-implementing, modifying and parameter tuning algorithms which in many cases are not in the public domain. We appreciate that many more techniques from PDP solvers could be used or adapted for use on the PMDP, however, doing so would be out of scope for our research, our aim is to find an approach capable of finding and exploiting opportunities for cost saving in order to carry out the business design research presented in Chapters 7 and 8. The real-world data set is run with the best parameter settings of each method (Section 5.4.1) for 40 minutes of CPU time, and repeated 100 times.

Figure 6.12 shows the average performance over CPU time. From discussions with our industrial partner Transfaction Ltd., current manual scheduling procedures are most closely approximated by our initial greedy insertion procedure, the cost of which is represented by the starting figure of around £33,500 in this example data set. Clearly, utilising any of the reviewed computational methods results in large savings to delivery cost. On average, and over any amount of running time, VNDM can find better solutions for the real-world problem than the tested alternatives and is relatively simple to implement. We can also see that BEBO produces results that are competitive if run-times are kept below 10 minutes but that it gets stuck at a worse level than other approaches after this.

Figure 6.13 shows the distribution of final cost achieved by each hyper-heuristic

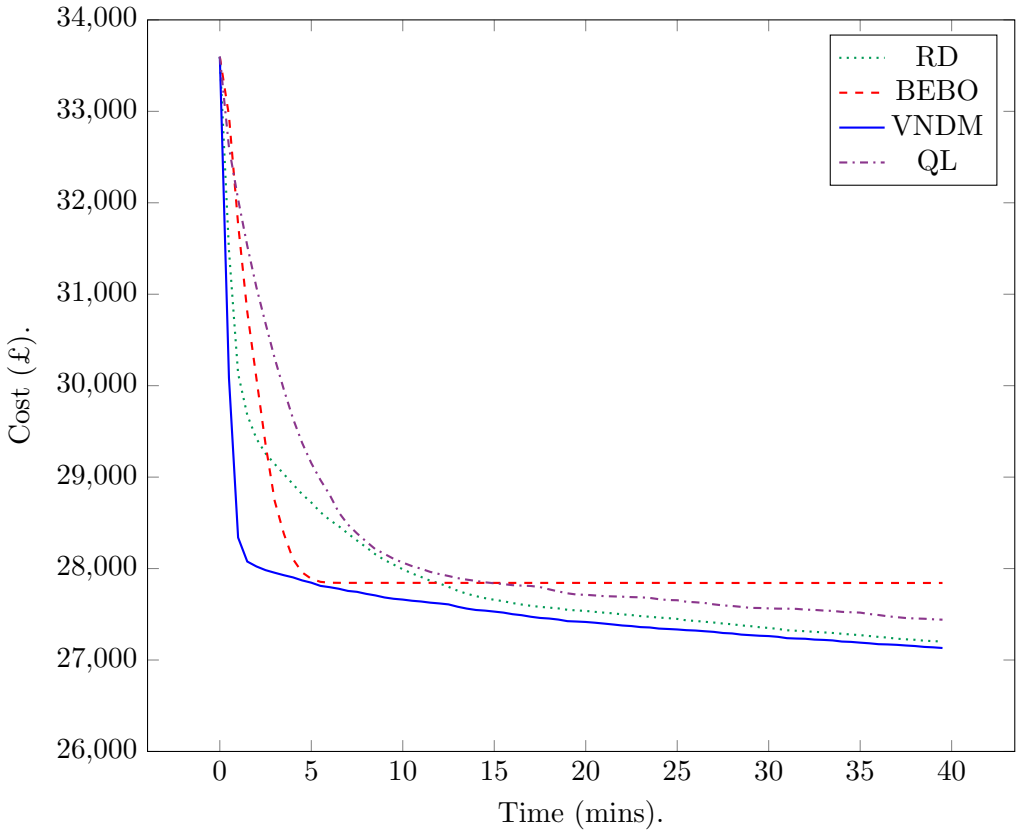


Fig. 6.12: Average performance over CPU time for a real data set, note that BEBO converges after about 5 minutes and then no longer produces any improvements.

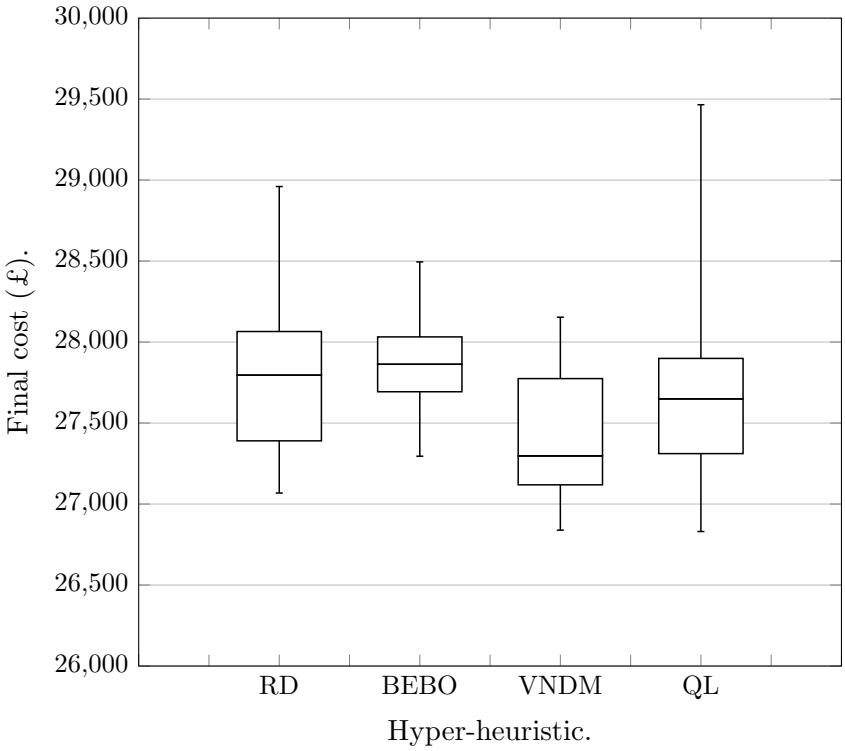


Fig. 6.13: Distribution of final cost across 100 runs.

across the runs. There is considerable overlap in the plots, though VNDM can be seen to outperform the alternatives, having lower minimum, maximum and quartile costs. It is also clear that BEBO is unsuitable for the PDP, as it produces the highest average final cost and is generally outperformed by RD. QL looks slightly better in these results than in Figure 6.12 as its best and many of its results are competitive, but it also produces the most expensive schedules, hurting its average.

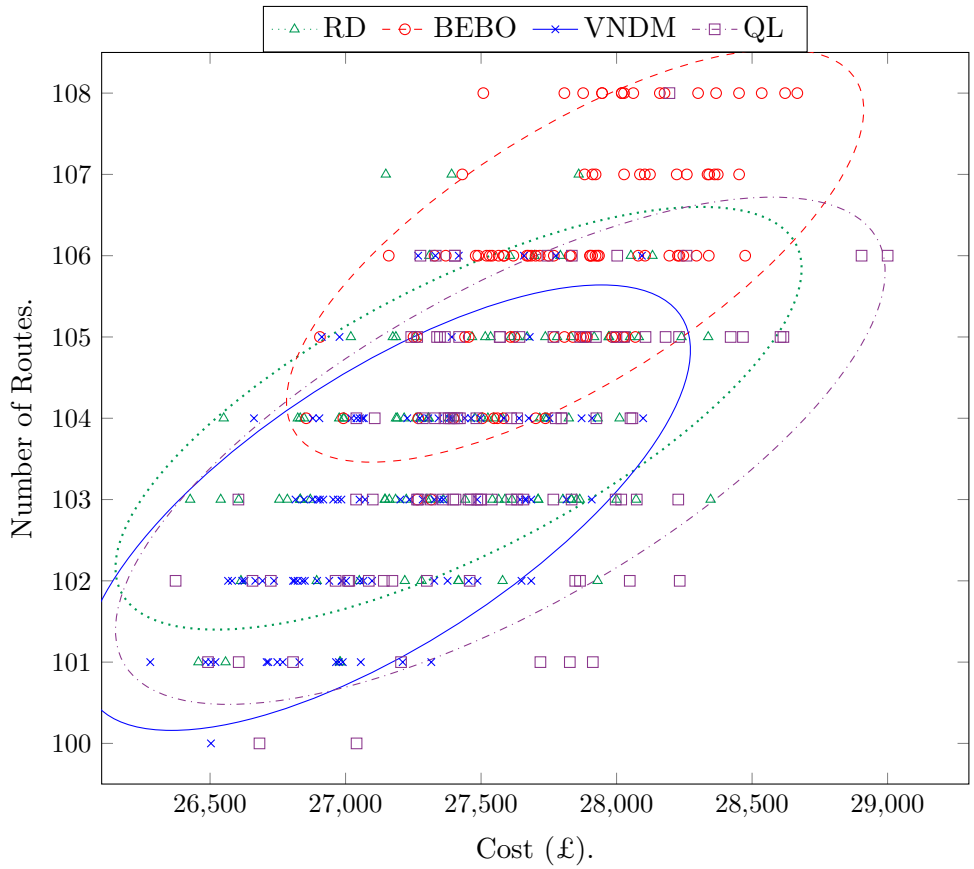


Fig. 6.14: Number of routes versus cost for real data set. Ellipses represent two standard deviations from the mean of each method.

Figure 6.14 plots the results as number of routes against cost. We can see that, for our real-world data set, there is a strong correlation between the number of routes and cost, but that the cheapest solutions do not necessarily have the fewest routes. Approaches that aggressively minimise for number of routes may not be able to perform as well as the approaches here from an operational cost perspective. The real-world results most closely resemble the LC 1-4-z benchmark instances.

Though we have only provided results for a single real-world data set, since VNDM

proved competitive with state of the art approaches on many small benchmark instances, we can say with some confidence that VNDM is a suitable choice for our real-world problem, even though it has undergone no parameter tuning to specifically fit the real-world data. In the following chapters we will present many different results using VNDM for realistic problems which provide further evidence for the suitability of the heuristic.

6.5 Discussion and Summary

This chapter presented the data, provided by Transfaction Ltd., needed to perform the analysis in this thesis. Assumptions and limitations of this data are outlined, along with methods for cleaning incomplete data and for generating additional and missing values for backhaul consignments. This allows us to generate realistic instances of a variety of sizes for our experiments, the generated data have been assessed by logistics experts in Transfaction Ltd.

DES, capable of simulating the arrival of consignments in a realistic manner, for testing scheduling strategies in a real-time environment, is described. Using DES I have shown VNDM to perform better than alternative hyper-heuristics in a dynamic real-world situation. Combined with the results from Chapter 5, VNDM is demonstrated to be not overfitted to benchmarks or the real-world data set. We have shown that the traditional PDP objective of minimising the number of routes in priority to total distance does not always produce the cheapest solutions in a real-world problem. The balance between vehicle maintenance costs and distance based running costs should be considered simultaneously.

We have not considered how robust to unforeseen circumstances, such as vehicle breakdowns, the schedules generated are, though in reality there are clauses written into driver working time rules which allow overtime for these occurrences (Department for Transport, 2016).

7

Cooperation and Competition

The research in this chapter has been published in (Mourdjis et al., 2016b). Section 7.3 presents results submitted to a special edition of "Communications in Computer and Information Science" (CCIS) published by Springer. This chapter investigates cooperation and competition between haulage companies in a dynamic PMDP where consignments represent less than full truck loads and vehicles may be loaded with more than one consignment providing LIFO constraints are satisfied. The model of the system, objective function and pricing information is that described in Chapter 4. All consignment data used to produce these results has been drawn from Transfaction Ltd.'s real-world UK retailer data set, described in Chapter 6.

7.1 Introduction

Logistics is a highly competitive industry; large hauliers use their size to benefit from economies of scale while small logistics companies are often well placed to service local clients. To obtain economies of scale, small hauliers may seek to cooperate by sharing loads. With over six thousand hauliers in the UK alone (Dff International Ltd., 2015), competition is fierce. Hauliers face the orthogonal demands of short notice from customers, an expectation of low-cost service, and environmental sustainability concerns (McLeod et al., 2012; Nahum, 2013; Demir et al., 2014). Because larger carriers can leverage economies of scale to benefit in routing and scheduling, competition is getting ever stronger. If smaller carriers could work together, they could increase scheduling

efficiency, save on mileage costs, and improve flexibility. Preliminary research into cooperation has been published in (Mourdjis et al., 2014).

A driving force behind our research is that huge savings are possible by combining consignments that are less than the capacity of a vehicle. In the UK supermarket delivery market for example, Transfaction Ltd. estimates that $\frac{1}{3}$ of vehicle miles are travelled empty. Discounting weight constraints, 44-tonne articulated trucks have a capacity of 52 goods pallets. Deliveries, on the other hand, typically lie within a range of 8-47 pallets with an average of 18[1]. There is clear scope for optimisation here.

In this chapter we quantify the savings possible when carriers distributed across a country outsource some of their customer consignments to other carriers, working either independently or as a group. In partnership with Transfaction Ltd., we propose realistic cost and revenue functions to investigate how companies of different sizes could cooperate to both reduce their operational costs and to increase profitability in a number of different scenarios.

All the results in this chapter are based on the simulation of one dynamic scheduling week, with optimisations limited to 5 minutes of CPU time. Each configuration in this chapter is run 30 times and 95% confidence intervals are presented in the results. A heterogeneous cluster of Intel Xeon based servers, totalling 72 cores and 120GB of RAM, was used. The results presented in this chapter thus represent thousands of hours of CPU time.

7.2 Cooperation

In collaboration with Transfaction Ltd., we have access to real scheduling data and manually-scheduled consignments for small UK hauliers (referred to as *real data*). As described in Chapter 6, the real data are insufficient, in quantity and quality, for our scheduling research, but provide us with indicative distributions and other information, from which we generate larger, realistic, data sets on requests and consignments (referred as *generated data*). We generate 100 scenarios from a data set of 27,153 real-world consignments. The scenarios are built by selecting 200 real consignments at random from this set and building pairs of consignments representing outbound linehaul and

[1] Based on information provided by Transfaction Ltd.

return backhaul legs. Each consignment consists of at least two requests.

Initially, each haulage company (carrier) is assumed to have an unlimited number of vehicles and is represented by a depot, randomly located within the area encompassing the consignments. Consignments are assigned to the carrier, from the set of carriers with the fewest consignments, that is geographically closest to the midpoint between a consignment's pickup and final delivery locations. Thus, the initial schedule systematically distributes consignments evenly across many carriers. The dynamic arrival of requests is scheduled using DES (Section 6.3.1).

7.2.1 Simple Cooperative Strategies

The first set of results compares the average per request costs for five carriers, exploring the effect on one carrier (the sample) under four different configurations of cooperation with the other four carriers. *All Contracted* has each consignment assigned to a specific carrier. Optimisation is only possible between vehicles belonging to the same carrier. *Out-sourcing* starts with a competitive model, but allows re-assignment of consignments from the sample to any of the other carriers, if cost savings can be made. *Out-sourcing to coop(erative)* adds the out-sourcing model for the sample carrier into a model in which the other carriers can exchange consignments if savings can be made; the sample carrier does not accept any additional consignments. Finally, the *cooperative* model initially assigns all consignments to individual carriers (as in *Contracted*) but allows unrestricted re-allocation during optimisation, if cost savings are possible.

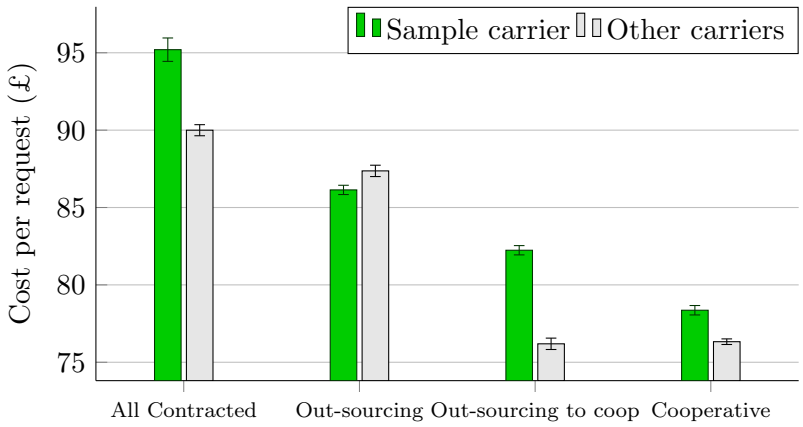


Fig. 7.1: Average cost for a single carrier (sample carrier) and a group of carriers (other carriers), with four different models of cooperation.

The costs presented in Figure 7.1 show that for the sample carrier, an average 9% saving can be made by out-sourcing to the four other carriers, whilst the configuration that allows other carriers to also cooperate results in average savings of nearly 14%, because the cooperation allows more efficient routing across the carriers. If the sample carrier also cooperates in efficient scheduling, the total average saving for the sample carrier rises to 18%. Cooperation is also beneficial for the other carriers: accepting consignments from the single carrier can produce benefits of 3%, whilst cross-group cooperation produces savings averaging 15%.

The results shown should drive all carriers towards cooperation. Competition favours carriers with the lowest costs; the sample carrier achieves this in configuration 2, by outsourcing to other carriers who are not cooperating. However, rational competitors would be expected to copy this behaviour, moving the system towards a reallocation of consignments as seen in configuration 3; here, the competitors are cooperating, and the sample carrier is at a competitive disadvantage. However, if all carriers cooperate, as in configuration 4, the lowest costs for all carriers are observed.

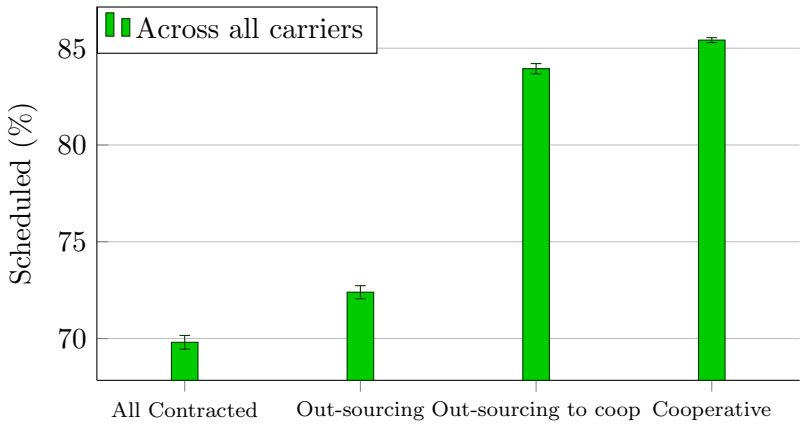


Fig. 7.2: Percentage of assigned consignments serviced across the four different models of cooperation.

Increasing cooperation allows a greater number of consignments to be handled. Figure 7.2 shows that the schedule in which all carriers operate alone covers on average less than 70% of their assigned consignments. However, the fully cooperative model can schedule over 85% of consignments. (Note that random scenario generation means that there is no guarantee that all consignments are feasible given the number of carriers, their locations and that even with an infinite number of vehicles, some consignments

are too far apart to be serviced whilst adhering to driver working time rules: since we do not consider driver sleeping arrangements and all routes must begin and end at the depot, these consignments are impossible in our current model.)

Tab. 7.1: Percentage of the sample carrier's consignments re-allocated in different configurations.

Config.	Cooperation Mode	Re-allocated
1	Competitive	0%
2	Out-sourcing	65.6%
3	Out-sourcing to coop	67.2%
4	Cooperative	57.2%

Table 7.1 shows the percentage of consignments that are re-allocated from the sample carrier in each configuration. Both out-sourcing and out-sourcing to a cooperative allow almost two-thirds of the carrier's consignments to be assigned to others: because our scheduling algorithm minimises cost, these re-allocations can be interpreted as being carried more cheaply, due to more efficient use of resources, when assigned to other carriers. We are most interested in the percentage of consignments that are re-allocated away from the sample carrier. When outsourcing and cooperation are combined (configuration 3), the sample carrier's re-assigned loads are most cost-effective, as, in this configuration, the other carriers can also re-allocate loads among themselves (but not to the sample carrier). In the fully cooperative model, the sample carrier's consignments are less cost-effectively reassigned than in other reallocation configurations. However, the overall cost-effectiveness of the 5 carriers is significantly better than in other configurations: 62.5% of other carriers' consignments were reallocated in this model, leading to the reduction in cost observed for cooperation in Figure 7.1. These results also strongly support the contention that savings can accrue to small hauliers who cooperate to carry each others' consignments efficiently.

7.2.2 More Group Configurations

We seek to further investigate the effects of different sized groups of carriers on both cost and network capacity. Using the same 100 scenarios as investigated previously, we now investigate how efficiently 10 carriers can service the consignments, split into a number of different group configurations. Cooperation is allowed within but not

between these groups. In the *Competing* configuration each of the 10 carriers works independently, in the second configuration, carriers work in *Pairs*. In *1 vs 3s*, one carrier, the sample, is compared against 3 groups of 3 carriers. In *5 vs 5*, *3 vs 7* and *1 vs 9* the 10 carriers are divided into 2 groups of differing sizes accordingly. In the final configuration, *Cooperative*, the 10 carriers work together.

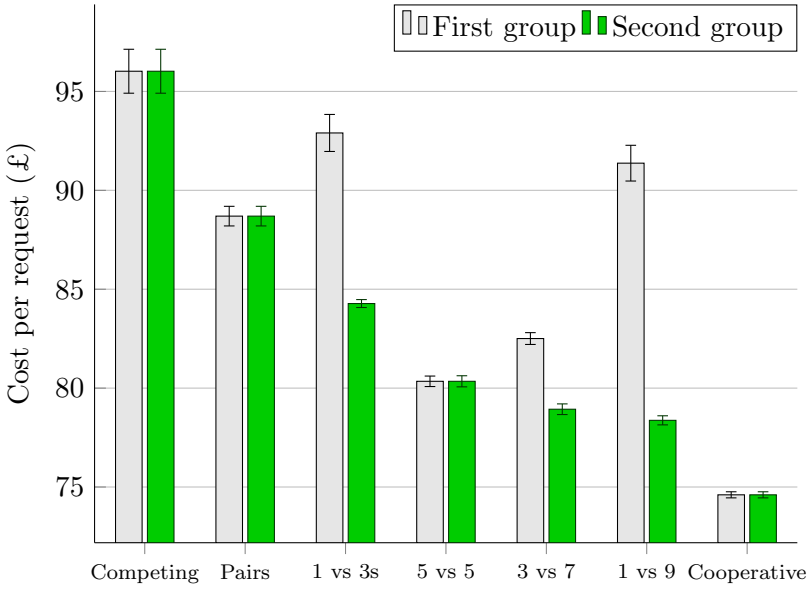


Fig. 7.3: Cost per request for different carrier group configurations.

Figure 7.3 confirms our earlier findings that working as a group can substantially reduce costs and additionally shows that larger groups can attain bigger cost reductions than smaller groups.

Figure 7.4 shows again the increase in network capacity made possible through cooperation. It is also clear that the largest savings are made quickly: just pairing with one other carrier can increase the number of scheduled deliveries from 72% to 80%. In each configuration, consignments are divided equally between groups, not carriers, such that, for example in the *1 vs 3s* configuration each group of carriers is assigned 100 consignments out of 400 but in the *1 vs 9* configuration, each group is assigned 200 consignments. Because of this, carrier 1 has more choice in the 1 vs 9 configuration and can achieve slightly better results than in the 1 vs 3s configuration, however the number of consignments that can actually be served is dramatically reduced.

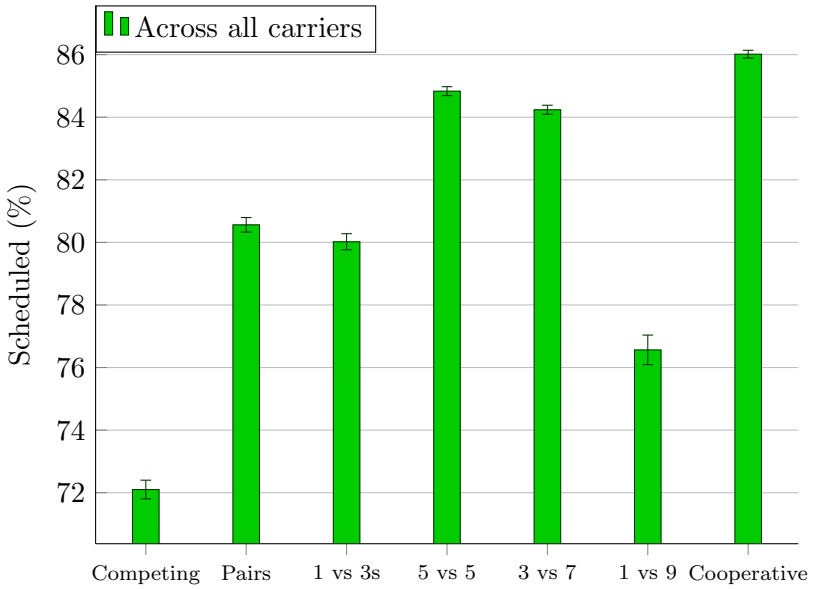


Fig. 7.4: Percentage of scheduled consignments for different carrier group configurations.

7.2.3 Carrier Group Size

Extending our analysis, we seek to identify if there are diminishing returns for increasing the number of carriers in a cooperative group. Here we assume a single cooperative group and modify the number of carriers. The consignments arrive identically in all cases.

Figure 7.5 shows how both the cost per request and the percentage of consignments scheduled improve as the size of a cooperative group increases. Though there are linear savings evident above 10 carriers, the majority of benefit is found between 1 and 5 carriers. These results must be qualified by stating that our consignments cover the UK and our carriers are randomly located across this area; since distance costs are a dominant factor in real-world pricing; if larger distances are involved, for instance across Europe, America or Asia, a larger number of well distributed carriers would likely be necessary to produce these savings. These results can be thought of more as suggesting that 10 major transport hubs is sufficient for efficient vehicle routes in the UK.

So far, we have assumed that all companies work together to reduce total cost and that an infinite number of vehicles is available at each carrier location; in practice there will be a limited supply of vehicles at each carrier and therefore multiple carriers in the

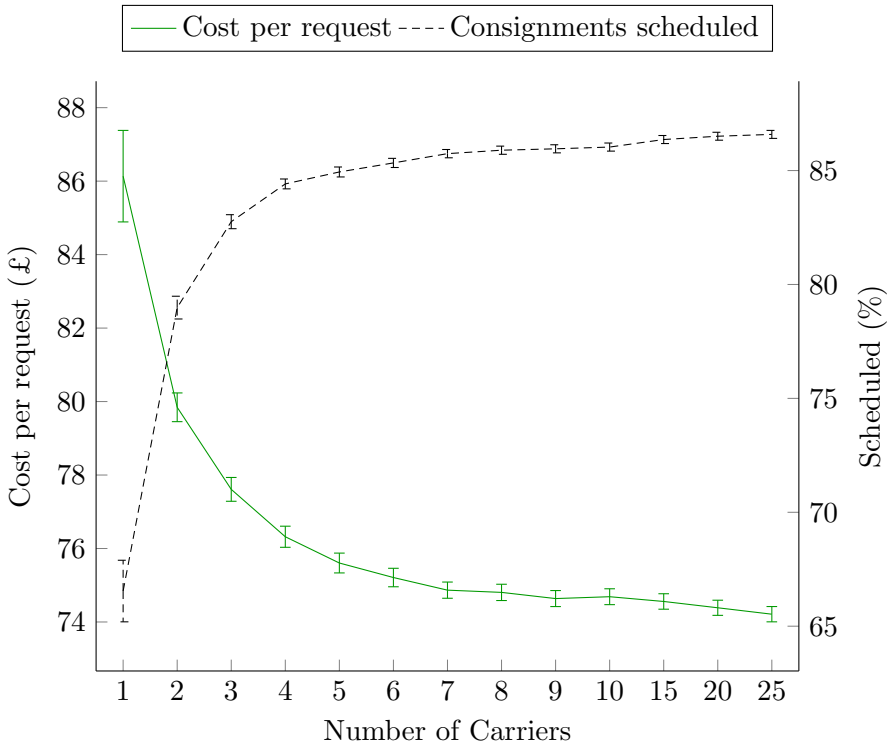


Fig. 7.5: Effect of carrier group size, the cost per request and number of consignments successfully scheduled versus the number of carriers working together.

same area would need to work together. The following section investigates cooperation in resource constrained situations.

7.3 Competition

In the real world, delivery companies compete with one another and are unlikely to work together unless it is beneficial to themselves. If companies are resource constrained they are still likely to be able to profit by outsourcing less profitable consignments to other companies, and instead servicing more convenient consignments. We use the same 100 scenarios each with 10 carriers and 200 consignments (assigned as before). However, companies now have a fixed number of vehicles. If a company cannot satisfy a consignment assigned to it, instead of creating additional vehicles, the customer is re-assigned to a random company that can service it. This means that a better utilisation of assets will lead to more customers for a given company. We assume that companies will not share their consignments if it results in them losing money, therefore, when cooperation is allowed between two companies, the company originally assigned a

consignment always receives the profit it would make. For a different company to fulfil this consignment, it must yield sufficient profit to pay off the original company and still cover the associated delivery costs. To do this, we introduce a simple model for consignment revenue, enabling us to estimate carrier profitability. The revenue model for a consignment is:

$$\text{Revenue}(c) = l_{p_c} \sum_{r \in D_c} nn_{r-1,r} \tag{7.1}$$

where the revenue of consignment c is a linear function of the total distance between all requests in the consignment (the summation) multiplied by the pickup load l_{p_c}. This is good enough for our uses as it is the same for any carrier and invariant of when the consignment was delivered (this is still handled by the delay penalty introduced in Section 4.4.3). A company's total profit is the revenue of all the consignments it delivers minus the total cost of serving these, as specified in Section 4.4.

We now consider variants of the scenarios previously investigated, but, in each case, the number of vehicles is fixed at 40. We consider the impact of cooperation in scenarios with different distributions of these vehicle between the 10 companies, to simulate different competitive environments.

7.3.1 Equally Sized Companies.

First, to validate our previous findings the 10 companies are set to have equal size, with 4 vehicles each. As expected, Figure 7.6 shows that cooperation increases the profitability of all companies by around 20 to 25%.

7.3.2 Differently Sized Companies.

The ten companies are now given different numbers of vehicles, set to: "2, 2, 3, 3, 4, 4, 5, 5, 6 and 6" respectively.

Figure 7.7 shows the increased profitability of the first three companies when they work together as a cooperative assuming that all other companies continue to work independently. Profit increases of 12-18% demonstrate that even the smallest companies benefit from cooperation.

Looking at the group of heterogeneously sized companies in more detail, Figure 7.8 shows how company size affects both raw profitability and the benefit of cooperation.

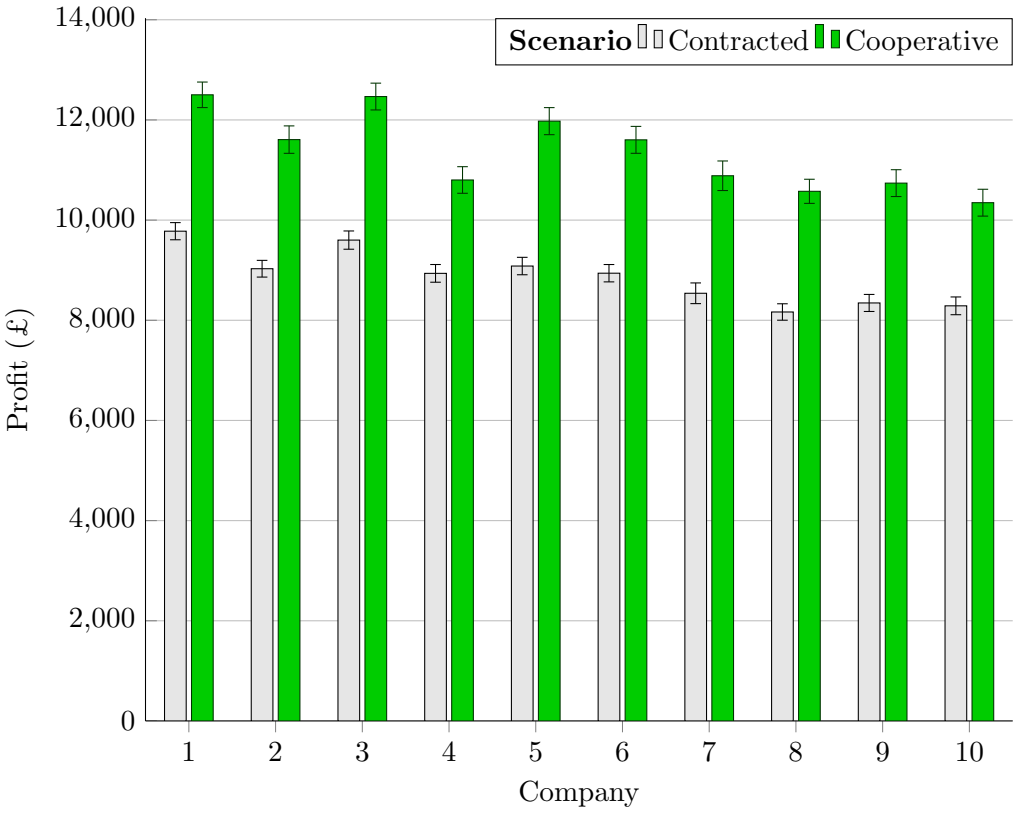


Fig. 7.6: Effect of cooperation on the total profits for ten equally sized companies.

Larger companies are able to produce more optimal routes and service more customers, generating more profit. When all parties cooperate, the profits for companies of all sizes increases. We can see that, as a percentage, small companies stand to gain the most from working cooperatively, with gains of up to 50%. Compared to the 12-18% result, above, it is again clear that more companies working together produces better results.

7.3.3 Large vs Small Companies.

In this scenario we compare the profitability of large and small companies competing in the same market. The ten companies are now set to: "8, 2, 2, 2, 2, 2, 2, 2, 9, and 9" vehicles respectively. Figure 7.9 shows that, initially (when no companies are cooperating), the 2 largest companies produce the most profit. When the small companies work together they can increase their profitability and reduce the profits of the larger companies. Finally we observe that if the first large company joins the cooperative it can massively outperform its competitors. Other companies' profits fall, and the

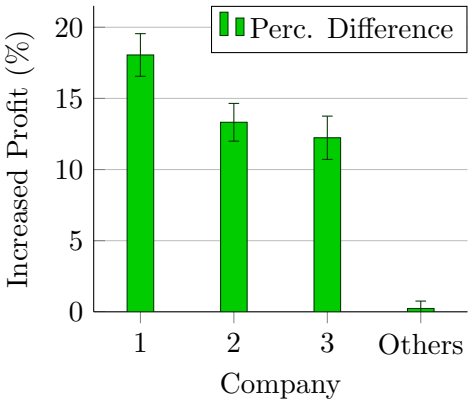
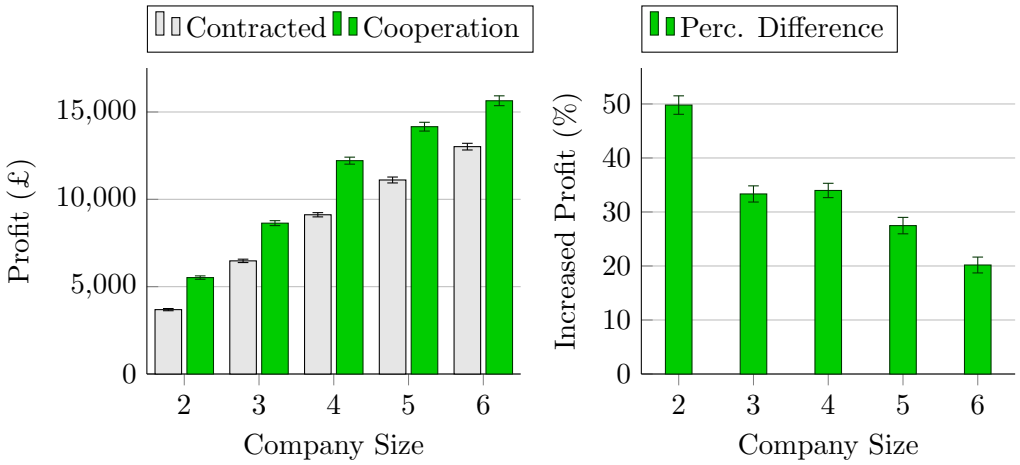


Fig. 7.7: Effect of cooperation between small companies.



(a) Profit across companies of varying size *(b)* Profit increases across companies of varying size

Fig. 7.8: Effects of cooperation across different sized companies, absolute and percentage change of profit.

cooperative can more effectively handle consignments (so consignments are not stolen by the large companies).

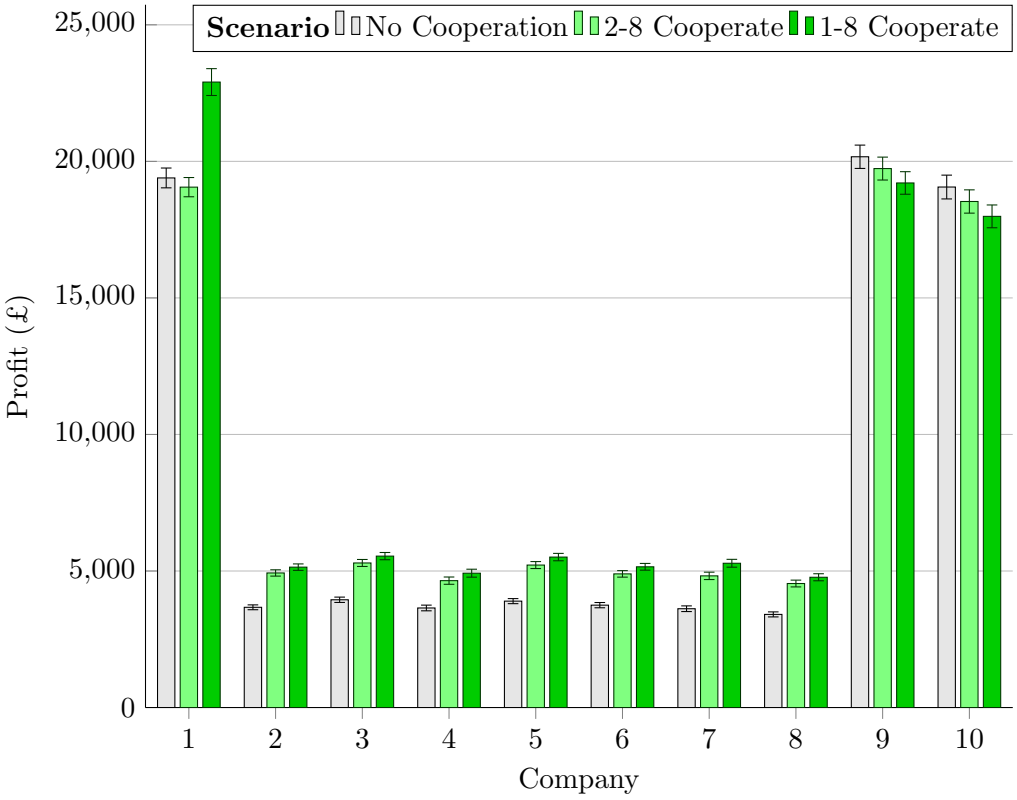


Fig. 7.9: Effect of cooperation between large and small companies. Total profit in a scenario where company 1 has 8 vehicles. Companies 2-8 have 2 vehicles each. Companies 9 and 10 have 9 vehicles each.

7.4 Discussion and Summary

Cost estimations from the RHA (Dff International Ltd. (2014), see Section 4.4) have been used to explore the pricing and marginal costs of delivering consignments. In reality, delivery companies charge what they can get away with for consignments; if there is little competition, prices rise. Conversely, a competitive market leads to lower prices. We have shown that cost savings of 15% to 18% are possible when hauliers cooperate. Cooperation also increases the capacity of a group of hauliers, by as much as 21%. The benefits of cooperation see diminishing returns above 10 separate carrier locations working together assuming sufficient numbers of vehicles to meet demands. Larger cooperatives will always have lower operating costs than smaller ones as they are able to more efficiently schedule their consignments to the most optimal company

locations.

We have carried out further investigation into how savings from cooperation could be turned into increased profit in resource constrained problems with a fixed number of vehicles. We propose that the revenue from a customer be modelled as a linear combination of distance and load and define company profit as the sum of revenues over all delivered consignments minus the costs associated with delivering these loads. We consider that each company aims to maximise its own profit by only reassigning customers when a cooperating company can pay off the original company's profit and still cover its delivery costs. The cooperating company makes the cost saving as its profit on such consignments. We have shown that this more realistic model of cooperation still leads to increased profits for all cooperating parties in a variety of different scenarios with differing company sizes. A particularly interesting result is that competing large companies stand to significantly benefit by cooperating with a group of smaller companies. Benefits of cooperation scale with the number of companies in the cooperative but generally lie within 15-20%.

So far, this research only investigates the marginal costs associated with deliveries, profit margins, loss leading and other marketing techniques are outside the scope of this research. We do not consider issues of vehicle reliability, for example, who pays the costs associated with missed delivery slots and what effect this has on customer perceptions. We have not considered the fixed costs associated with carrier-owned vehicles in this research; implementing the strategies outlined in this paper may result in reduced usage of carrier owned assets as cooperation allows for an increase in capacity, allowing the same fixed cost assets to be more productive, assuming there is sufficient demand for service.

The following chapter investigates another potential area of savings, the impact that the amount of planning time and the width of arrival time windows have on the cost of delivering consignments.

8

The Value of Time and Timely Information

8.1 Introduction

Just-in-time business processes are increasingly common and bring additional complexity to logistics planning and scheduling. To efficiently manage these processes it is important to understand the impact of time windows and timeliness of information on costs. In co-operation with Transfaction Ltd., we investigate dynamic scheduling of shared loads for long distance truck haulage in the UK, modelled as the pickup and multiple delivery problem (PMDP), introduced in Chapter 4. Since a significant percentage of journeys involve vehicles with partially or completely empty loads, combining these journeys, where appropriate, can enable large savings that are often missed in practice. We use our variable neighbourhood descent with memory (VNDM) heuristic, presented in Chapter 5, which we have shown to be capable of producing large cost savings in acceptable time for both real-world and synthetic instances (Section 5.4).

This chapter analyses the effect of altering the length of planning and arrival windows on a number of important metrics for schedule quality, both in the department store chain data set (Section 6.2) and in a large group of UK retailers (Section 6.1). The key findings are that savings of up to 20% are possible by increasing these time windows, and that planning beyond 12 hours in advance offers diminishing returns, as

sufficiently good routes can be built with this much notice. As logic would suggest, wider arrival windows always enable shorter, cheaper routes as there are more opportunities to combine geographically close consignments. Overall, our results provide a framework for understanding the time value of information (Cowling and Johansson, 2002) in practice, and the cost savings due to relaxation of pickup and delivery windows so that hauliers can understand the discount that might be offered to customers for providing timely information and wider time windows.

A significant real-world impact of this work is to quantify the time value of information and the savings that are possible by extending arrival time windows. We also show that current delay penalties offer little incentive for on-time delivery, especially when considering the large cost savings possible by widening planning or arrival windows. A discussion of how consignments may be discounted is presented in Section 8.7.

8.1.1 Parameters for Analysis

For these experiments we utilise two different real-world data sets, the first is sourced from a UK department store chain (introduced in Section 6.2) and is the same as that used in Chapter 5 (Sections 8.2 and 8.3). The second is drawn from the large data set of UK retailers presented in Section 6.1 and includes some generated data (Section 8.4 and 8.5). The department store data is a smaller but more complete existing schedule where direct comparisons to the manual scheduling used by the department store can be made.

Since we investigate the effect of time windows on truck scheduling we consider altering the planning and arrival windows of consignments in each scenario. The planning window is the amount of warning between a consignment's details being received and the start of its pickup request time window. The arrival window is the time between the start and end of the time window at every request, defining when it is acceptable to arrive at that request (Figure 4.3, Section 4.2.1, page 74). We have chosen to investigate values for both planning and arrival window which cover the wide range of practices observed in real-world retail haulage problems, ranging from 30 minutes to 168 hours (7 days) in appropriate steps.

For each data set, we first consider the effects if we are able to modify all con-

signments; such that all consignments have the same planning and arrival windows, We term this *homogeneous*. We then investigate whether the observed improvements are still possible if only a small proportion of the consignments are modifiable. We term the second group of experiments *heterogeneous*, as the consignments do not all have the same planning and arrival windows. Results for the homogeneous scenarios are presented in Sections 8.2 and 8.4, while heterogeneous scenarios are presented in Sections 8.3 and 8.5.

We consider a typical week's worth of scheduling; where all consignment's requests have start times that fall within a one week period. As the planning and arrival windows of the requests expand, the planning horizon of our experiment, the length of time from the beginning to the end of the experiment, also grows such that for our initial week's worth of consignments the planning horizon is equal to one week + planning window + arrival window. In the extreme case of 7 day planning and 7 day arrival windows, the test period would cover 3 weeks. Consignments would only be serviced in weeks two and three.

We analyse the effect that changing these parameters has on the following metrics of schedule quality:

Distance the total distance in kilometres of all truck-routes throughout the planning horizon.

Delay the average time in minutes that each request is delayed (note that this is an average, so the direct cost of delay cannot be gathered from these charts due to the stepwise delay penalty function described in Section 4.4.3)

Utilisation the average load of trucks in the schedule: $\frac{\sum ll_{rk} * nn_{ru}}{\sum nn_{ru}} \forall r \in R$ where ll_{rk} is the load of the scheduled truck k after visiting request r and nn_{ru} is the distance between requests r and u (for example if a truck-route has two legs, the first at 50% load for two kilometres and the second empty for one kilometre, the utilisation of the truck is 33%).

Cost per request the average cost in pounds for each request (using parameters discussed in Sections 4.4.1, 4.4.2 and 4.4.3). Note that consignments are made of at least 2 requests.

8.2 Scenario 1 - Department Store Case Study

The first scenario is a set of consignments for a large department store chain based in the UK. The data set comprises 387 consignments over a one-week period for which we have the real-world schedule, it is introduced in Section 6.2 and is the same as that used in Chapter 5.

A summary of the existing schedule for scenario 1 is presented in Table 8.1. Truck utilisation in the original, static, greedy schedule is around 40% with a per-request cost of £43.06. In comparison to this our optimisation is able to produce schedules of similar cost (albeit with higher delay) even with the tightest of time windows (30 minutes) in a dynamic scheduling environment.

Tab. 8.1: Manual versus optimised schedules, example results for the static, manual schedule and optimised, dynamic schedule when planning and arrival windows are both set to 30 minutes (30M), 4 hours (4H) and 7 days (7D).

Metric	Manual	30M	4H	7D
Distance (km)	21,199	25,802	20,619	17,243
Delay (mins)	0	30.0	9.5	0
Cost per request (£)	43.06	44.07	37.48	33.86
Utilisation (%)	39.84	33	42.20	49.78

Table 8.1 also shows that utilisation, the average load of all truck kilometres in each set of parameters, increases with both planning and arrival window length, ranging from 33% when windows are shortest to 50% when both planning and delivery windows are set to 7 days. These results suggest that better information and optimisation brings environmental and cost benefits.

Figure 8.1 shows the very marked effect on delay of tighter arrival windows. We also see that in some cases, a relaxed planning window results in slightly increased delays: This is likely due to trading delay for shorter, cheaper routes. However, even in the highest case this delay is only 34 minutes per request, which is below the threshold for any delay penalty. The optimisation phase has traded "free" delay for a reduction in distance that has an observable impact on cost.

Figure 8.2 shows the expected relationship between cost and planning / arrival windows. The relationship between the distance travelled and these windows is very similar (Figure 8.3). We conclude that using the cost structure set out in the RHA

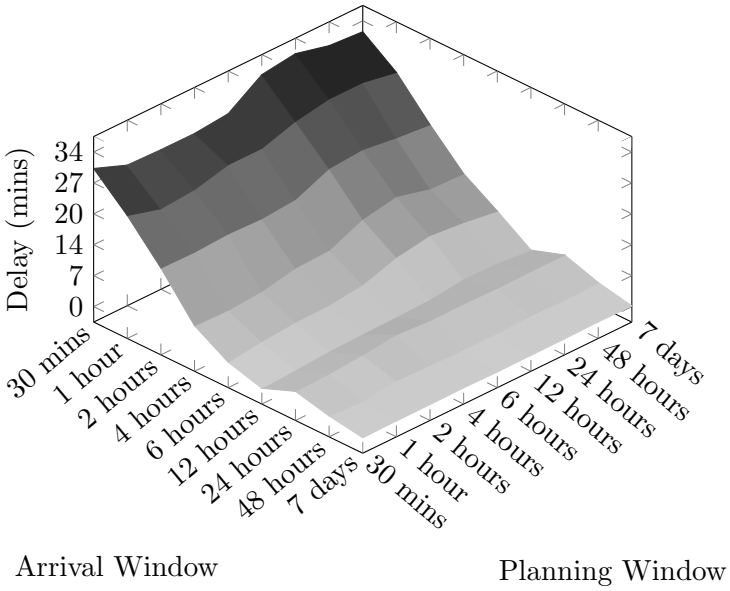


Fig. 8.1: Scenario 1, average delay per request. Delay starts incurring cost penalty at 2 hours (120 mins)

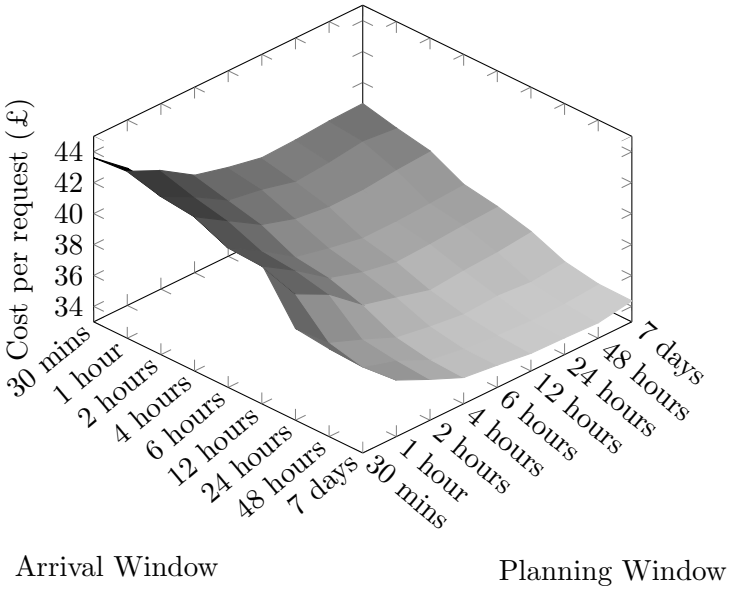


Fig. 8.2: Scenario 1, average cost of servicing one request (a consignment is made of at least two requests)

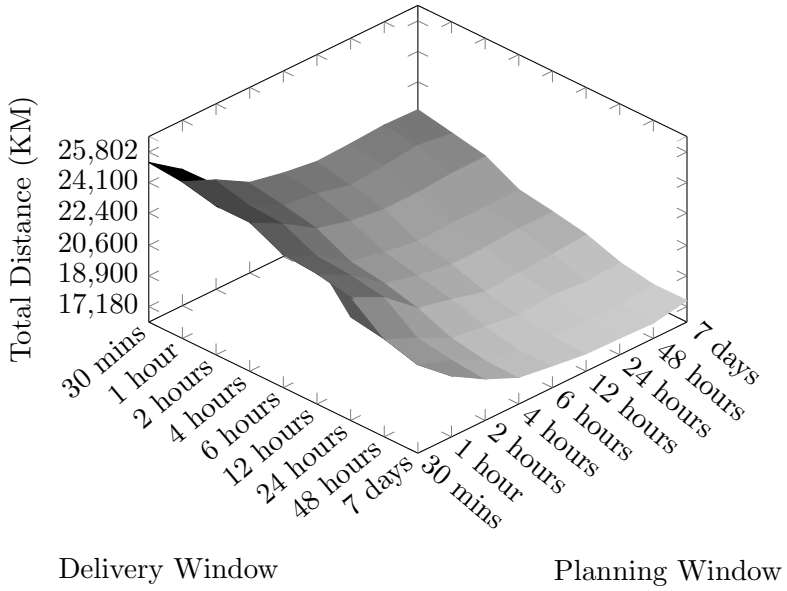


Fig. 8.3: Scenario 1, total distance of solutions.

tables (Section 4.4), and using a realistic delay cost function, journey distance (ranging from 17,180 to 25,802km) dominates the cost of road haulage. Comparing 4H and 7D in Table 8.1, we see a substantial mileage and cost reduction when using longer planning windows. This is particularly marked beyond four hours, as this provides enough time to insert most consignments optimally in terms of distance. A slight upward trend in cost is evident once planning windows exceed 12 hours (Figure 8.2); this can be explained by the CPU time limit imposed on the optimisation step. As the planning window is increased the search space grows and many more consignments are modifiable at any time in the simulation. Much longer CPU runs do yield improvement for the more relaxed problem instances. However, to keep the time for running experiments manageable, a lower time limit was chosen here; the optimisation is unable to converge in the given time when faced with the larger search space.

To clarify the impact of a wider planning window on cost, shown in Figure 8.2, a series of cross-sections through the surface plot have been made at 30 mins, 6 and 48 hours; these are presented in Figure 8.4. This plot is based on 30 runs and illustrates the substantial savings resulting from having at least 12 hours' notice of a consignment.

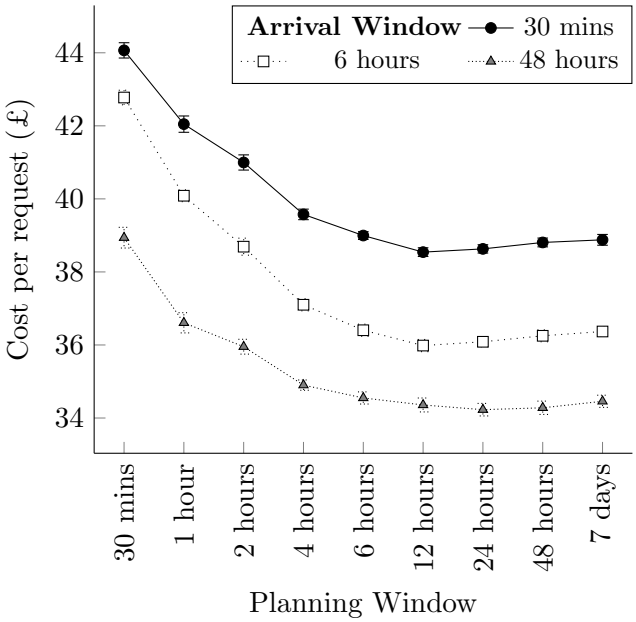


Fig. 8.4: Scenario 1, cross sections of cost per request.

8.3 Scenario 2 - Department Store, Heterogeneous Consignments

The results so far represent a homogeneous case where all consignments have the same planning / arrival windows. In the heterogeneous case, however, the cost reductions that follow from advance information or larger time windows only apply to a small fraction of consignments. The following experiments investigate the effect of controlling a 10% sample of the consignments, where the remaining 90% have planning and arrival windows selected uniformly at random from the set of all time windows.

Analysing results for the 10% of controllable consignments, Figure 8.5 shows a small but significant increase in utilisation, from 38.5% to 40.5%, as planning and arrival time windows widen.

Another interesting aspect of this experiment is that the delay and cost of the sampled consignments (shown in Figures 8.6 and 8.7 respectively) follow the same trend as for the homogeneous case, with significant cost savings as the planning and arrival windows are relaxed. However, the benefit of an increased arrival window is diminished once the planning window exceeds 6 hours; this is sufficient in this scenario to produce schedules without significant delay.

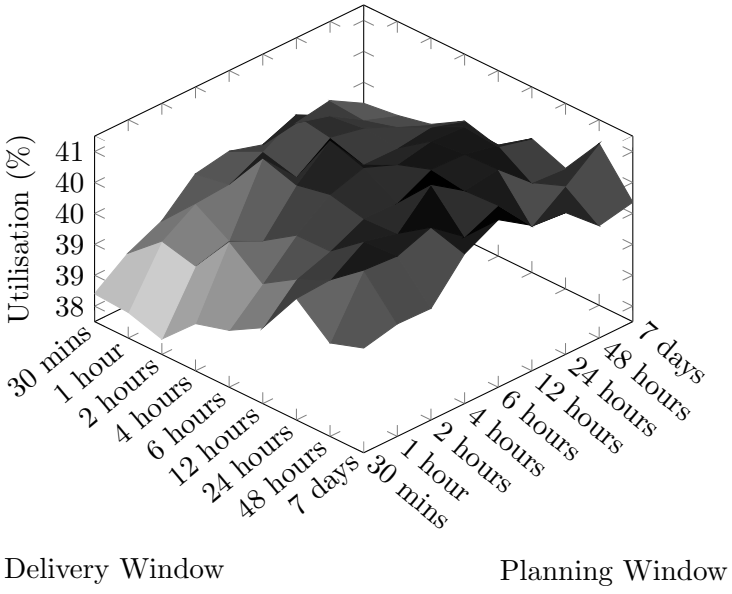


Fig. 8.5: Scenario 2, utilisation across time windows.

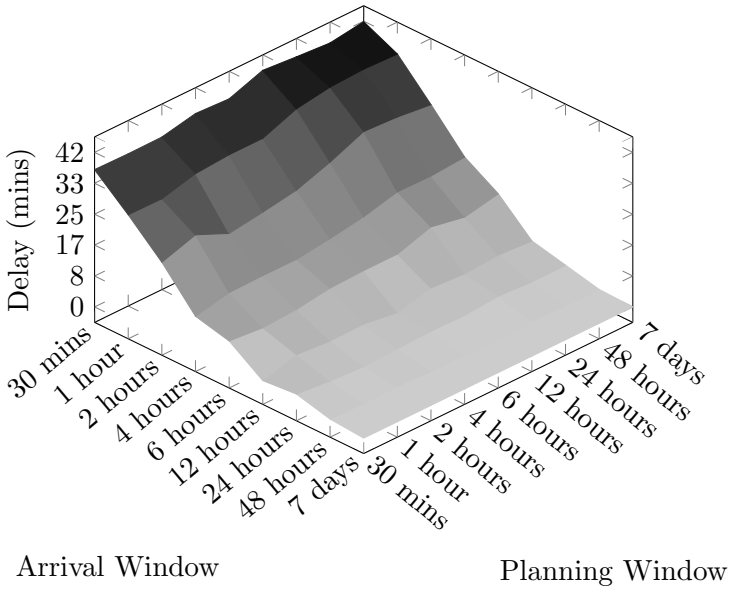


Fig. 8.6: Scenario 2, average delay per request.

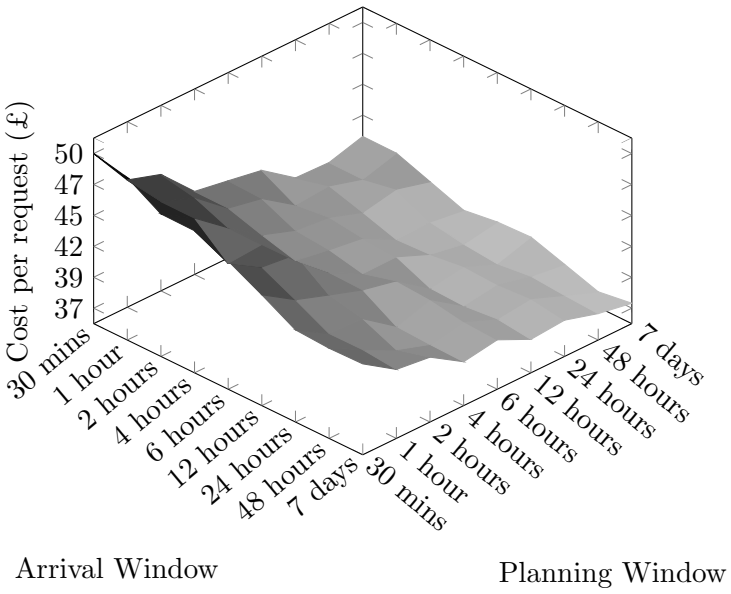


Fig. 8.7: Scenario 2, average cost of servicing one request.

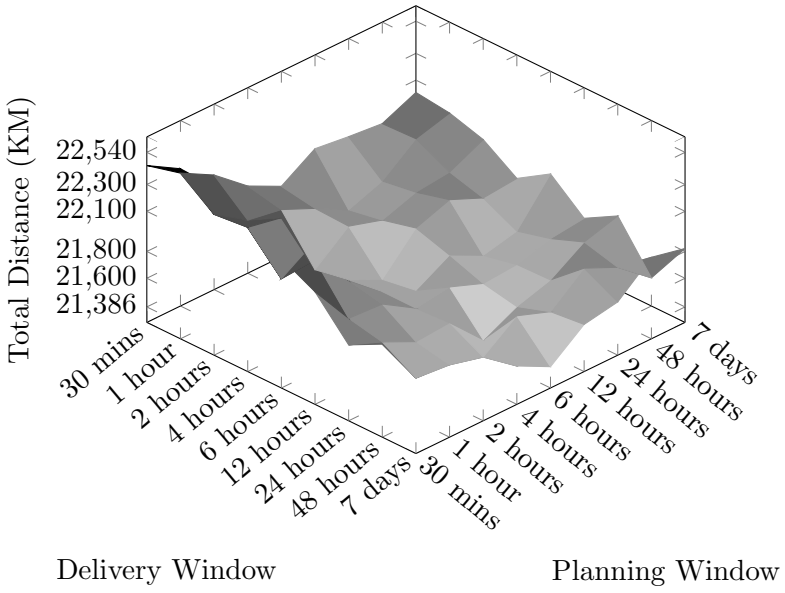


Fig. 8.8: Scenario 2, total distance of solutions.

In this heterogeneous setting, the total distance changes much less dramatically than in the homogeneous case, ranging from 21,386 to 22,540 kilometres as shown in Figure 8.8. Distance also does not seem to be as important in determining the per request cost as in the homogeneous case.

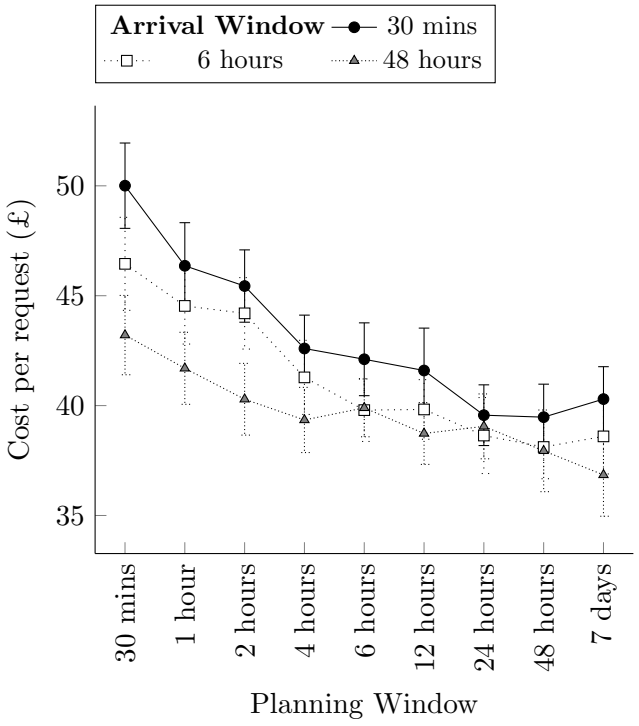


Fig. 8.9: Scenario 2, cross sections of cost per request at 30 minute, 6 hour and 48 hour arrival windows.

Figure 8.9 shows a cross section of Figure 8.7 so that the effects of planning and arrival windows can be observed in more detail. In the heterogeneous case, increased planning windows have a greater effect on cost than wider arrival windows.

The uncontrolled consignments' delay and cost do not vary greatly with modification of the controlled consignments' time windows, and are around 15 minutes and £39 respectively in all cases. A more detailed examination of the 90% of uncontrolled consignments shows very small effects as the planning and arrival windows of sampled consignments are altered. These fluctuations are as much due to the stochasticity in the optimisation procedure as to the underlying data. In the worst case, however, costs are increased by approximately £1 per request. Delays vary by less than two minutes.

8.4 Scenario 3 - UK Retailers Case Study

The third scenario is a larger, partially generated, data set drawn from three carriers and 220 hauliers working for UK retailers as described in Section 6.1. We generate 100 separate trial sets of consignments "instances" from a data set of 27,153 real-world consignments. Each instance comprises 400 consignments, built by selecting 200 real consignments at random from this set and building pairs of consignments representing outbound linehaul and return backhaul legs. In the real data, time windows are often missing. Data is generated where needed as in Chapter 6. There is no implied precedence between a consignment and its associated backhaul (a backhaul may be scheduled before its linehaul); this is acceptable as, in practice, routes are repeated week after week. A backhaul that occurs before its logical linehaul is the backhaul for this same linehaul from an earlier period. Loads for the linehauls and backhauls are assigned in the same way as for the real case, described in Section 8.2.

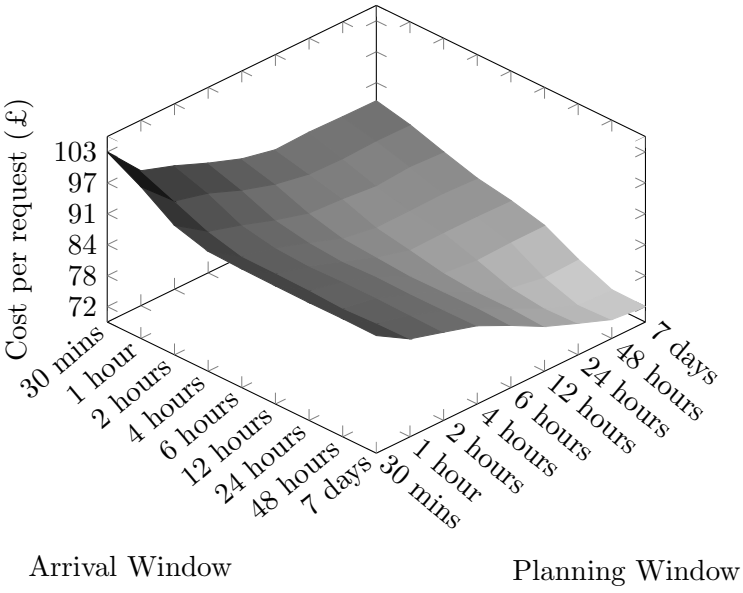


Fig. 8.10: Scenario 3, average cost of servicing one request.

The requests in scenario 3 and 4 cover a wider area of the UK than those in scenarios 1 and 2, and are therefore more spread out. This difference is reflected in the higher average cost per request shown in Figure 8.10 and the greater total distance shown in Figure 8.11. With the larger distances between requests in this scenario, increasing the planning window can have a larger effect on cost than increasing arrival windows.

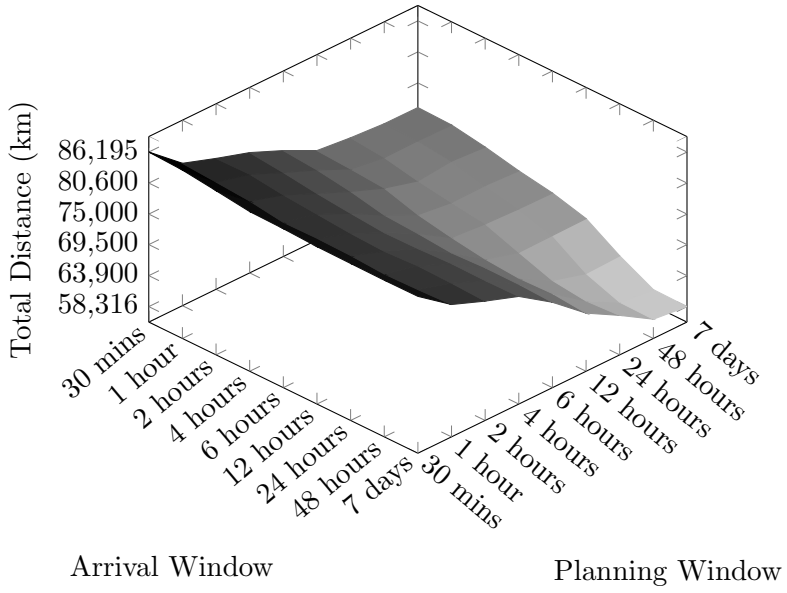


Fig. 8.11: Scenario 3, total distance of solutions.

Larger planning windows can enable more effective combinations of consignments (regardless of the size of arrival time window).

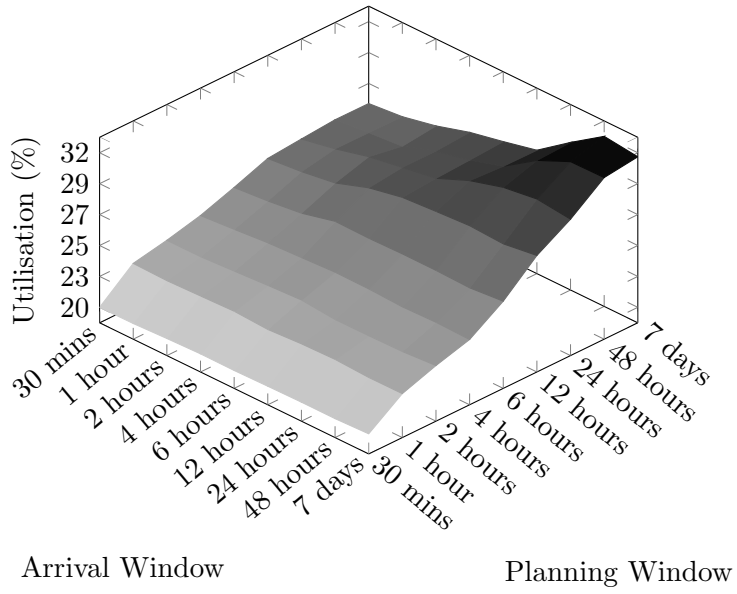


Fig. 8.12: Scenario 3, utilisation across time windows.

Figure 8.12 shows a very strong correlation between increased windows and utilisation; it clearly shows that increasing arrival windows alone is not beneficial but must be combined with additional increased planning windows.

Figure 8.13 shows the impact on delay. Here increasing the arrival window has a

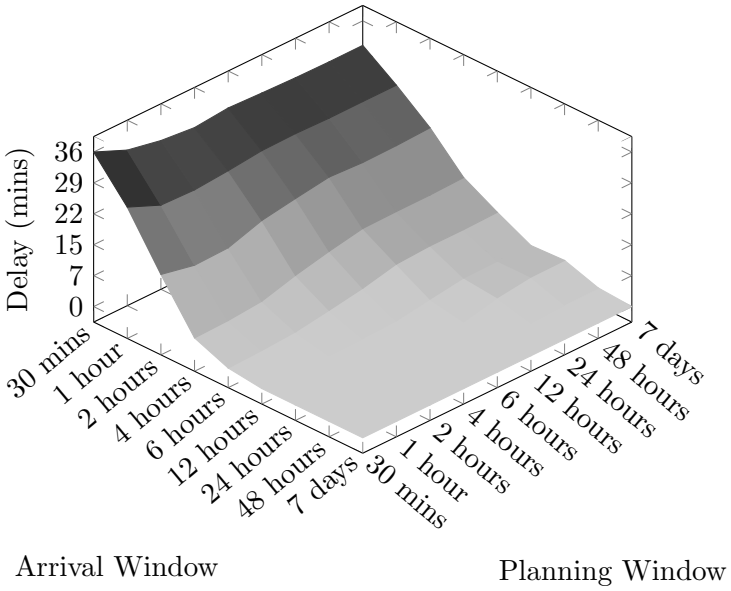


Fig. 8.13: Scenario 3, average delay per request.

direct impact without necessarily changing the schedule at all. The benefit of increasing the arrival window tapers off after the 6 hour mark, again suggesting that this amount of time is sufficient to guarantee timely delivery. Increasing planning windows does not affect the overall delay; as the optimisation process always favours cheaper solutions this implies that it is cheaper to have shorter, delayed routes than longer on-time routes, given the cost structures outlined in Section 4.4.

8.5 Scenario 4 - Retailers, Heterogeneous Consignments

Scenario 4, is a heterogeneous version of scenario 3. Figures 8.15 to 8.17 show results for the 100 UK retailers instances, described in Section 8.4, when considering changes to only 10% of consignments. The differences with the homogeneous case are in line with those observed between scenarios 1 and 2. In the heterogeneous case, the same patterns are observable as in scenario 3 but the range of each is reduced; delay peaks at 25 minutes, per request cost ranges from £76 to £99, total distance is from 69 to 73 thousand kilometres and utilisation is from 24 to 26%.

Figures 8.14 and 8.16 again show the correlation between solution length and per request cost with plannning windows showing slightly more impact than arrival windows. Figure 8.17 likewise shows more impact from planning than arrival windows, with 7

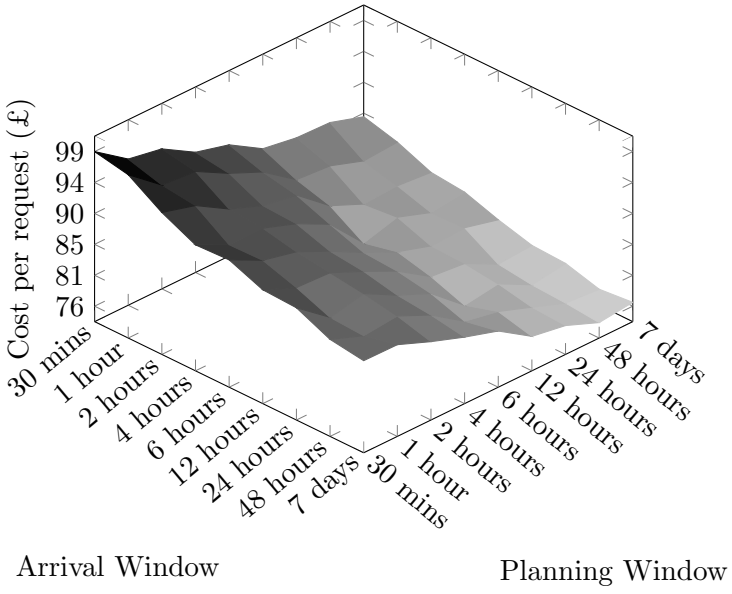


Fig. 8.14: Scenario 4, average cost of servicing one request.

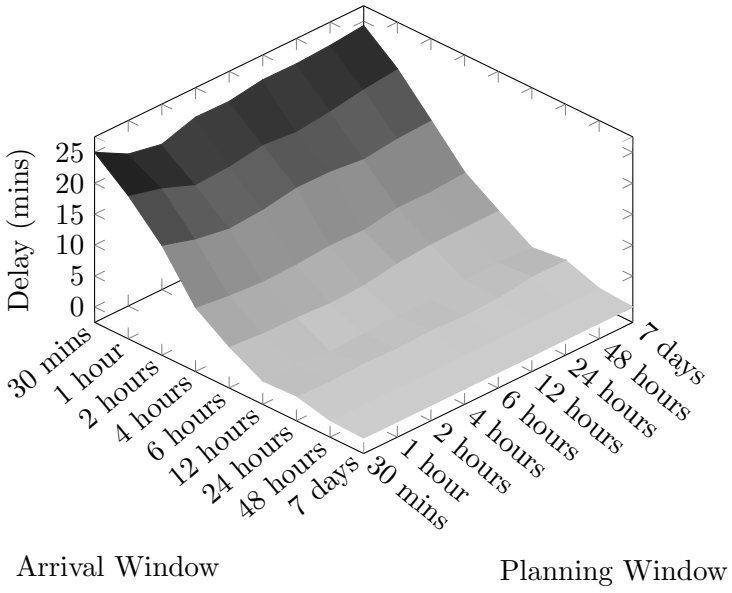


Fig. 8.15: Scenario 4, average delay per request.

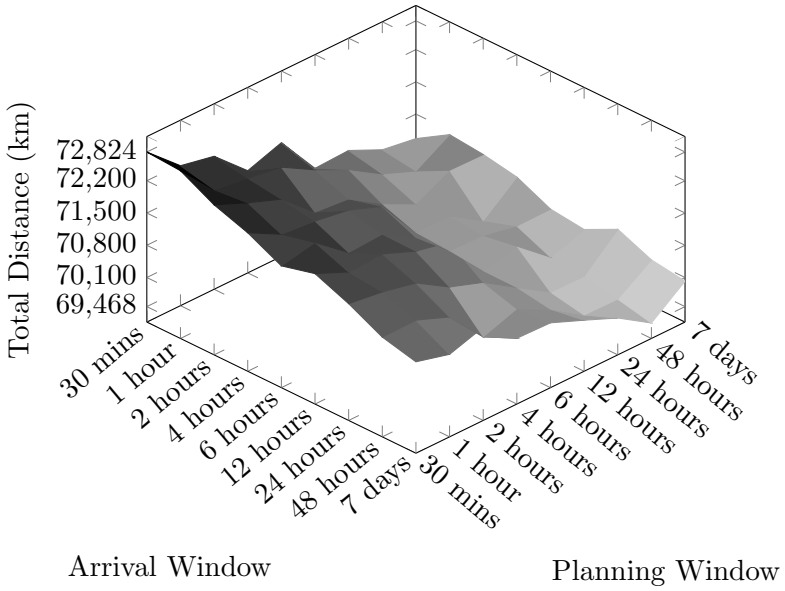


Fig. 8.16: Scenario 4, total distance of solutions.

day planning there is almost no difference in utilisation across all arrival windows.

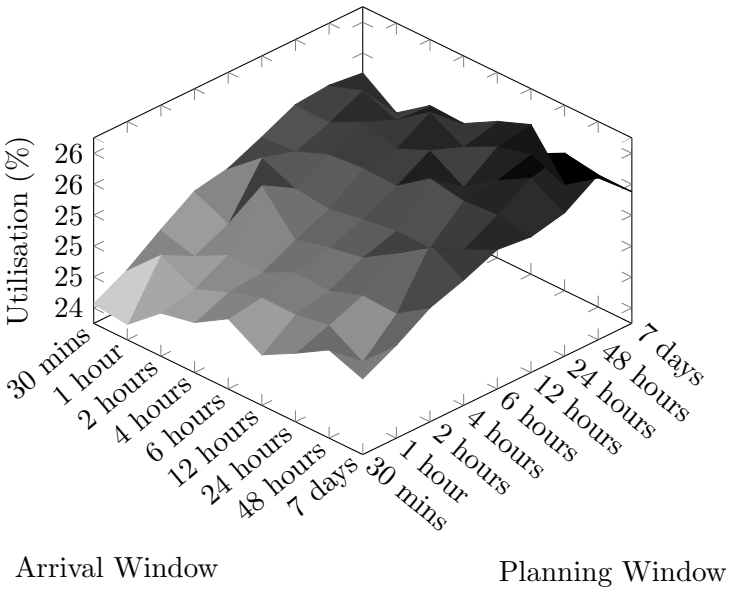


Fig. 8.17: Scenario 4, utilisation across time windows.

8.6 Impact of Time Windows

To evaluate the impact of time windows in general, the following results for each scenario have been scaled, taking the cost with 30 minute arrival time windows as the baseline (100%).

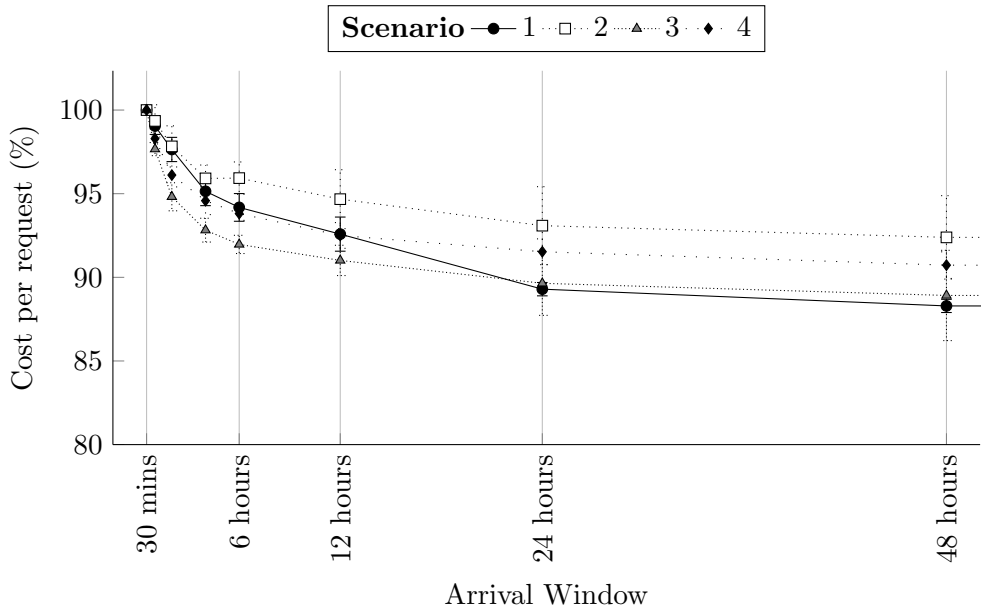


Fig. 8.18: Impact of increasing arrival window from 30 mins to 48 hours across 4 scenarios.

Figure 8.18 shows that, once the arrival window reaches 12 hours, the cost per request is reduced by approximately 10%. Further improvements beyond this are possible but not as significant.

Figure 8.19 shows similar results when increasing planning windows. Moving from 30 minutes to 12 hours can produce cost savings of between 10 and 15%, depending on the scenario, suggesting that discounts could be offered to encourage customers to give more notice of upcoming consignments. The reduction in cost by increasing the planning window does not change greatly after 12 hours. As in Section 8.2, this could be due to the cut off CPU time. However, relaxing the CPU time limit sees only slight improvements, suggesting there are few additional savings to be made beyond the 12 hour planning window. Intuitively this makes sense, as, in the case of having no better solution, more time to plan is not going to make any difference.

The relative importance of increasing arrival or planning windows is similar across all tested scenarios. However, improvements from increasing the planning window are

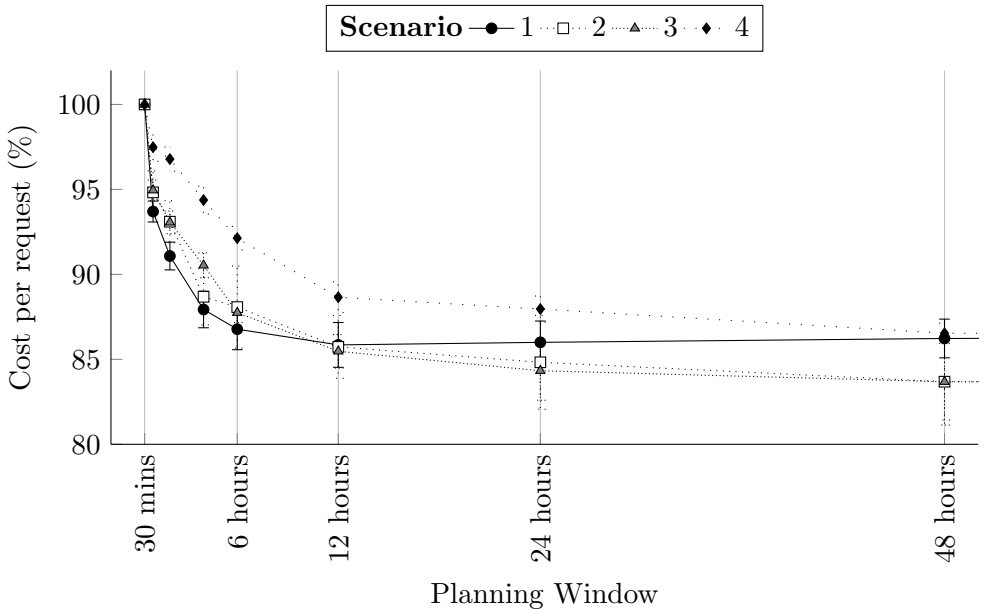


Fig. 8.19: Impact of increasing planning window from 30 mins to 48 hours across 4 scenarios.

significantly larger when viewed from this perspective. It would be possible, given these figures, to offer discounts of 10 to 15% on consignments if the planning window was increased from 30 minutes to 12 hours, with a further 5 to 10% discount offered if arrival windows were relaxed in the same manner. Discounts that could only be offered in combination with efficient, automated scheduling.

8.6.1 Degree of Dynamism

The effective degree of dynamism for a problem with time windows (edod-tw) (Larsen, 2001) can be calculated as:

$$\textit{edod-tw} = \frac{1}{R_{tot}} \sum_{i=1}^{R_{tot}} (1 - \frac{tt_i^{response}}{T}) \tag{8.1}$$

where $tt_i^{response}$ is the response time (planning plus arrival window ($tt_i^{planning} + tt_i^{visit}$)), T is the planning horizon (the length of time between receiving the first request and the latest end time of any request) and R_{tot} is the total number of requests. Edod-tw can range in value from 0, a static problem, to 1, a completely dynamic problem. There is a large spread of problem type in the homogeneous case (Table 8.2), from 0.333 when both planning and arrival windows are 7 days to 0.994 when both are 30 minutes. In the

heterogeneous case (Table 8.3) these values are 0.828 and 0.894 respectively, a much smaller range. In both cases, the edod-tw values follow the same trend, with larger windows producing more dynamic problems.

	0.5	1	2	4	6	12	24	48	168
0.5	0.994	0.991	0.985	0.974	0.963	0.931	0.873	0.776	0.499
1	0.991	0.988	0.982	0.971	0.960	0.928	0.870	0.774	0.499
2	0.985	0.982	0.977	0.966	0.955	0.923	0.866	0.771	0.497
4	0.974	0.971	0.966	0.955	0.944	0.913	0.857	0.764	0.494
6	0.963	0.960	0.955	0.944	0.933	0.903	0.848	0.757	0.491
12	0.931	0.928	0.923	0.913	0.903	0.875	0.824	0.737	0.483
24	0.873	0.870	0.866	0.857	0.848	0.824	0.778	0.700	0.467
48	0.776	0.774	0.771	0.764	0.757	0.737	0.700	0.636	0.438
168	0.499	0.499	0.497	0.494	0.491	0.483	0.467	0.438	0.333

Tab. 8.2: Edod-tw for homogeneous scenarios (1 and 3), axes are planning and arrival windows for all requests

	0.5	1	2	4	6	12	24	48	168
0.5	0.894	0.894	0.894	0.894	0.893	0.892	0.890	0.885	0.861
1	0.894	0.894	0.894	0.894	0.893	0.892	0.890	0.885	0.861
2	0.894	0.894	0.894	0.893	0.893	0.892	0.889	0.885	0.861
4	0.894	0.894	0.893	0.893	0.893	0.891	0.889	0.884	0.861
6	0.893	0.893	0.893	0.893	0.892	0.891	0.889	0.884	0.860
12	0.892	0.892	0.892	0.891	0.891	0.890	0.888	0.883	0.859
24	0.890	0.890	0.889	0.889	0.889	0.888	0.885	0.880	0.857
48	0.885	0.885	0.885	0.884	0.884	0.883	0.880	0.876	0.852
168	0.861	0.861	0.861	0.861	0.860	0.859	0.857	0.852	0.828

Tab. 8.3: Edod-tw for heterogeneous scenarios (2 and 4), axes are planning and arrival windows for the 10% of sampled requests

All requests have start times that fall within a one week period. As the planning and arrival windows of the requests expand, the planning horizon of our experiment, the length of time from the beginning to the end of the experiment, also grows such that, for our initial week's worth of consignments, the planning horizon is equal to one week + planning window + arrival window.

Revisiting scenario 1, our department store data, and plotting edod-tw against average request cost (Figure 8.20), we can see that in general, less dynamic problems, with lower edod-tw values, have lower service costs than more dynamic problems. However, there are multiple, different results for individual values of edod-tw. This is due to both

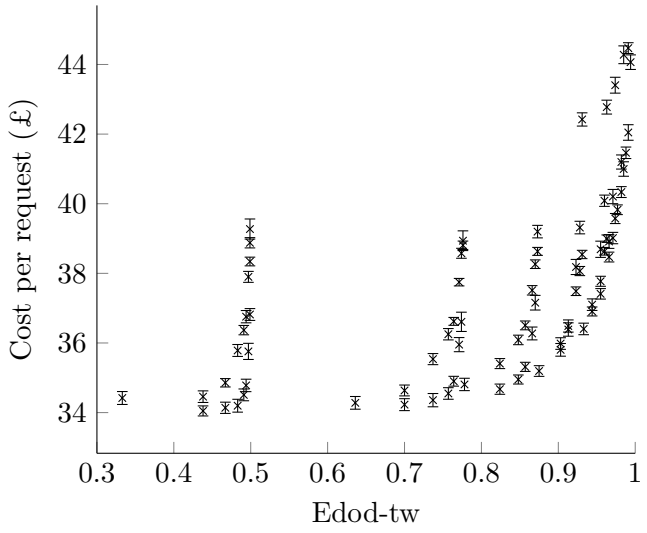


Fig. 8.20: Edod-tw versus average request cost in scenario 1.

the arrival and planning windows having the same effect on edod-tw. For example, a planning window of 6 hours and an arrival window of 2 hours yields the same edod-tw as a planning window of 2 hours and an arrival window of 6 hours but results in a different per-request cost. Due to this we must conclude that, on its own, edod-tw does not capture enough information about a problem to determine the difficulty of solving it or the costs involved.

8.7 Discussion and Summary

The conclusions from studying the effect of time windows can be summarised as follows.

- Once the arrival window reaches 12 hours, the arrival delay factor becomes negligible and per-request cost is reduced by approximately 10% compared to the manual strategy. It would be possible, given these figures, to offer discounts of up to 10% on consignments if required arrival windows were increased from 30 minutes to 12 hours.

- Increasing the planning window (from notification of a consignment to pickup) from 30 minutes to 48 hours can produce cost savings of between 15 and 20%, depending on the corresponding arrival window. This suggests that a discount could be offered as an incentive for customers to give more notice of upcoming consignments.

- The relative importance of increasing arrival or planning windows is similar for all tested scenarios. Looking at our average results over 100 retailer instances, it would seem reasonable to offer discounts of 10% for clients who are prepared to accept planning or arrival windows increased from 30 minutes to 6 hours, with further discounts of up to 30% if both planning and arrival windows are 48 hours.

- The current industry standard for pricing delay is not significant enough to encourage on-time delivery. Currently, a haulage company seeking to maximise profits would be best served producing shorter routes and arriving at destinations late (albeit within the one hour grace period). This conclusion is drawn from our results, observing that total distance charts closely resemble the cost charts.

- As planning and arrival windows increase it becomes possible to combine consignments, resulting in increased utilisation of assets and a reduction in costs. These benefits are realisable even if only a small portion of consignments adopt the increased windows proposed, as we observe similar trends when only modifying 10% of the consignments in a scenario.

- Relaxing the CPU time limit sees only slight improvement suggesting there are no additional savings to be made in our scenarios beyond a 12 hour planning window. The reduction in cost does not increase significantly when the planning window exceeds 12 hours. Intuitively this makes sense, as, in the case of having no better solution, more time to plan is not going to make any difference.

- Less dynamic problems, with lower edod-tw values, in general have lower associated service costs, as expected, than more dynamic problems. However, a direct relationship between edod-tw and cost is not possible as two planning and arrival window pairs may produce the same edod-tw but lead to different costs.

This chapter presents a number of interesting findings on the effect that changing time windows has on various aspect of schedule quality; that increasing time windows reduces distances, delays and costs whilst increasing utilisation is an intuitive result, as is that increasing planning windows can produce larger savings than arrival windows.

That the delay cost function used in industry offers no incentive for on time service is a new observation and suggests that companies should perhaps charge more for missing arrival deadlines.

We have also shown that cost savings are possible even if only a small portion (10%) of consignments are altered, meaning that it could be possible to profitably implement a multi-tiered pricing model to encourage clients to either give more warning of upcoming consignments or accept wider arrival windows, assuming an efficient automated scheduling approach.

A study into how cost savings could be distributed between customers and hauliers is outside the scope of this research; this should also consider potential impacts on things like environmental pollution. The cheaper solutions that are discussed in this chapter are correspondingly shorter, as distance-related costs are dominant, it is intuitive to assume that shorter routes are better for the environment but this is not necessarily true, especially in cities. In practice a more realistic model of travel time and pollution would be required to guarantee these savings.

9

Conclusions

My collaboration with Transfaction Ltd. has produced a number of interesting and industrially relevant conclusions. This chapter begins with a summary of the key findings evidenced in this thesis before introducing a number of places where this research could be extended if future work were to be undertaken.

9.1 Key Findings

I have detailed the PMDP as a model for a real world, supplier-to-customer haulage problem as exists in truck routing in the UK (Chapter 4, page 71). The PMDP shares a number of similarities with the PDP but introduces multiple deliveries for each consignment which must be serviced in a specific order. In addition to common extensions to the PDP such as soft time windows and driver working hours, the PMDP also includes LIFO loading constraints due to the rear access nature of trucks and loading bays. Cost estimations from the RHA (Dff International Ltd., 2014) have been used to explore the pricing and marginal costs of delivering consignments.

We introduced the VNDM hyper-heuristic (Chapter 5, page 83) to solve PMDP and have shown it to be a good choice for our real world case studies, as well as being competitive with best known solutions for small benchmark instances of the PDP (Li and Lim, 2003). We have shown (Section 5.4, page 92) that, in limited CPU time, VNDM outperforms BEBO, QL and RD on many of the 100, 200 and 400 customer static benchmark instances, as well as in a dynamic real-world situation. This result

is demonstrated with a variety of different parameter settings and is not overfitted to either benchmark instances or the real-world data set. State of the art solutions, which outperform VNDM on benchmark PDPs, would need significant alteration and specialisation in order to produce solutions for the PMDP, due to the assumption of a single delivery being inherent at many levels of other heuristic designs. VNDM is not perfect, merely a good enough practical tool to conduct the case study analysis.

For many of the random, and some of the clustered, instances in the Li and Lim (2003) benchmarks, shorter solutions are possible if more routes are used (Figure 5.7, page 97). We have shown that the traditional PDP priority of minimising the number of routes, rather than minimising total distance, does not always produce the cheapest solutions in a real-world problem (Figure 6.14, page 116). The balance between vehicle maintenance costs and distance based running costs should be considered simultaneously in the objective of PMDP solvers.

Cooperating delivery companies can make significant savings when an efficient optimisation strategy such as VNDM is employed. Consignments are more often delivered by the carrier which can deliver them most cheaply. A coordinating body such as our industrial partner, Transfaction Ltd., has the potential to deliver increased carrier profits, reduced distributor costs and increased utilisation in the delivery chain. Section 7.2.1, page 121 shows that delivery costs can be reduced by up to 20%.

We conducted a thorough investigation into the impact of both planning and arrival time windows to quantify the time value of information in routing and scheduling problems. We found that as planning and arrival windows increase, it becomes easier to combine consignments, resulting in 25% increased utilisation of assets (Table 8.1, page 136) and a reduction in costs of between 15 and 30%. These benefits are realisable even if only a small portion of consignments adopt the increased windows proposed, as we observe similar trends when only 10% of the consignments in a scenario have flexible time windows (Figure 8.7, page 141). A portion of this saving could be offered as a discount to customers, as an incentive to provide more easily serviceable consignments.

The reduction in cost by increasing the planning window does not change greatly after the 12-hour mark (for our heuristic), given the CPU time available. Relaxing the CPU time limit sees only slight improvement suggesting there are no additional savings

to be made in our scenarios beyond a 12-hour planning window. Intuitively this makes sense, as, in the case of having no better solution, more time to plan is not going to make any difference.

Once the arrival window reaches 12 hours the delay in our solutions becomes negligible (Figure 8.1, page 137) and cost per request is reduced by approximately 10% compared to the manual strategy (Figure 8.2, page 137). Our results show that the current industry standard for pricing delay is not significant enough to encourage on-time delivery. A haulage company seeking to maximise profits would be best served producing shorter routes and arriving at destinations late (albeit within the one-hour grace period).

We have shown that cost savings of 15% to 18% are possible when hauliers cooperate (Figure 7.1, page 121). Cooperation also increases the capacity of a group of hauliers, by as much as 21% (Figure 7.4, page 125). The benefits of cooperation see diminishing returns above 10 separate carrier locations working together (Figure 7.5, page 126) assuming sufficient numbers of vehicles to meet demands. Larger cooperatives will always have lower operating costs than smaller ones as they are able to more efficiently schedule their consignments to the most optimal company locations.

In reality, delivery companies charge what they can get for consignments; if there is little competition – prices rise. Conversely, a competitive market leads to lower prices. We have carried out further investigation into how savings from cooperation could be turned into increased profit in resource constrained problems with a fixed number of vehicles. We propose that the revenue from a customer be modelled as a linear combination of distance and load and define company profit as the sum of revenues over all delivered consignments minus the costs associated with delivering these loads. We consider that each company aims to maximise its own profit by only reassigning customers when a cooperating company can pay off the original company's profit and still cover its delivery costs. The cooperating company makes the cost saving as its profit on such consignments. We have shown that this more realistic model of cooperation still leads to increased profits for all cooperating parties in a variety of different scenarios with differing company sizes (Figure 7.9, page 130). A particularly interesting result is that competing large companies stand to significantly benefit by

cooperating with a group of smaller companies. Benefits of cooperation scale with the number of companies in the cooperative but generally lie within 15-20%.

9.2 Contributions

- A new model for the PMDP.
- An approach that has been shown to solve a variety of realistic scenarios in acceptable time, without requirements for specialist knowledge of the problem (domain).
- The cost and distance savings possible through cooperation have been quantified for differently sized companies.
- Discounts to suppliers who give longer notice or are more flexible on delivery times have been explored in detail.
- Currently used delay penalties are shown to provide little incentive for timely delivery.

9.3 Future Work

This section presents a number of areas where future study is needed to answer questions raised by my research.

9.3.1 Economic Impact

So far, this research only investigates the marginal costs associated with deliveries. Profit margins, loss leading and other marketing techniques are widespread throughout the logistics industry. How would carriers operating these strategies effect the benefits of cooperation? Our cooperative models have assumed relatively simple revenue sharing approaches. A study into how cost savings could be distributed between customers and hauliers would require substantial additional research, potential strategies involve allowing carriers to auction jobs to cooperating parties, or having some central control involved in deciding which carriers take what consignments. Which approach produces

the lowest cost for the consumer? Which approach would be easiest to implement? How could these companies be persuaded to work together?

9.3.2 Breakdowns and Unpredictability

There are more real-world issues than are discussed in this thesis, we do not consider issues of vehicle reliability, for example, who pays the costs associated with missed delivery slots and what effect does this have on customer perceptions? We have shown that cooperation allows for an increase in usable vehicle capacity, allowing the same fixed cost assets to be more productive, assuming there is sufficient demand for service. We have not considered the fixed costs associated with carrier-owned vehicles in this research. If there is insufficient or inconsistent demand, implementing the strategies that this research suggests to be beneficial may result in reduced usage of carrier owned assets; how would this affect our suggestions?

9.3.3 Green Logistics

It is possible to consider the environmental impact of a schedule, alongside the time, distance and costs investigated in this thesis. Can we guarantee that the routes generated through optimisation produce lower levels of environmental emissions? The cheaper solutions that are discussed in this thesis are correspondingly shorter, as distance-related costs are dominant. While it is intuitive to assume that shorter and more highly utilised routes are better for the environment, this is not necessarily true, especially in cities where slow speeds and stop-start traffic cause fuel consumption to increase. In practice a more realistic model of traffic, travel time and pollution would be required to guarantee these savings. Another question this raises is how can cost and environmental concerns be balanced? Without incentive, why would any company choose a potentially more expensive schedule?

9.3.4 Hyper-heuristic Research

The hyper-heuristic methods I have presented offer a practical solution for the PMDP, but are shown to struggle on random instances of the PDP, compared to the state-of-the-art solutions, there is opportunity for optimalising the HH approach used. VNDM

performs best on clustered instances and data such as our real-world problem. Why is this? Is this true for other hyper-heuristic approaches? Another area of investigation lies in applying VNDM to other problem domains. Though some domain specific knowledge is used to guide VNDM, the majority of the method is transferable to other problems. We have investigated the effectiveness of this approach on the periodic vehicle routing problem with some success (Chen et al., 2016a). Can VNDM be used as a simple all-purpose hyper-heuristic for combinatorial problems?

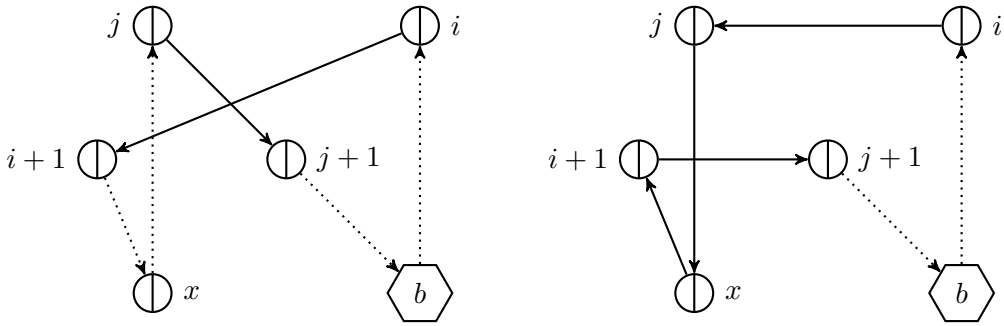
Appendix

A.1 Extended Glossary

A.1.1 Simple LSOs

A.1.1.1 k-opt

k-opt removes k edges from a single route and reconnects them in a different configuration. Figure A.1 demonstrates this for $k = 2$ (2-opt) on a very small example. Note that although the nodes j, x and $i + 1$ are still connected via the same route the *order* in which these locations is visited is swapped after the use of this operator. Cowling and Keuthen (2005) combine dynamic programming approaches within k-opt to reduce computation time.



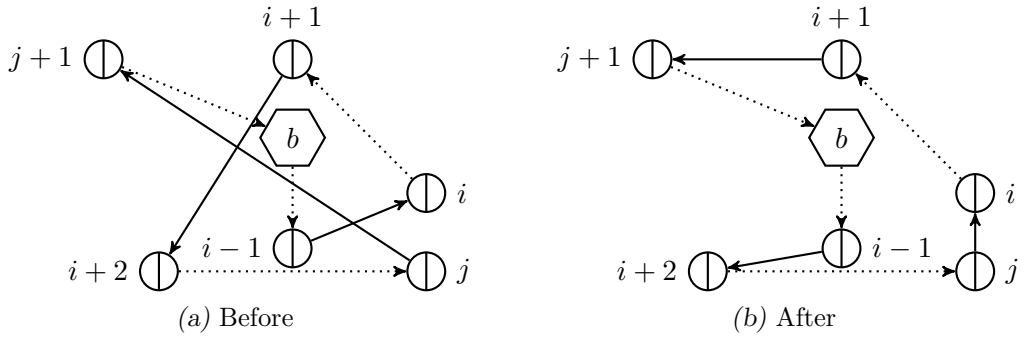
(a) Before *(b)* After

Fig. A.1: 2-opt operator.

A.1.1.2 Or-opt

Or-opt is a variant of k-opt where k = 3 and the direction of edges not replaced is preserved. In Figure A.2, the edges $i - 1 \to i$, $i + 1 \to i + 2$ and $j \to j + 1$ are replaced

by the edges $i-1 \rightarrow i+2$, $i+1 \rightarrow j+1$ and $j \rightarrow i$ respectively.



(a) Before *(b)* After

Fig. A.2: Or-opt operator.

A.1.1.3 I-opt

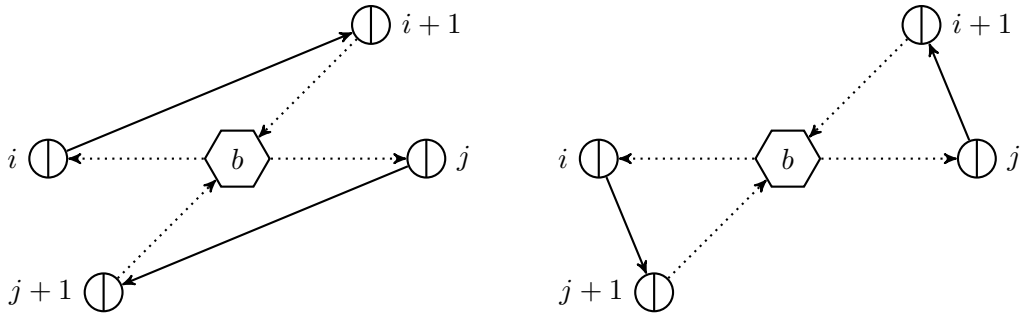
Bräysy (2003) presents I-opt as a modification of Or-opt such that instead of specifying edges to cut, a chain of nodes is selected to be preserved and a set of edges that satisfy Or-opt are cut. The resultant solution then reverses the preserved chains and applies Or-opt to reconnect the route. Bräysy (2003) tries to counteract the effects of insertion order on a tour by removing the customer furthest from the base location and $i-1$ nodes furthest from it, re-inserting these in every possible order using the cheapest insertion algorithm, and using Or-opt on the new routes after every k nodes are inserted.

A.1.1.4 2-opt*

2-opt* (Potvin and Rousseau, 1995) takes two complete routes in a solution and splits each in half. The resulting routes become the first half of one and the second half of the other and vice versa. Only 2 edges are changed, the paths before and after the cut remain unchanged and therefore are traversed in the same direction.

A.1.2 Tabu Search (TS)

Glover (1990) presents tabu search (TS) as a general purpose meta-heuristic framework for solving difficult optimization problems, applying TS to the TSP as an example application. TS attempts to avoid the problem of getting stuck in local optima associated with hill climbing algorithms. TS differs from traditional hill climbing algorithms with



(a) Before *(b)* After

Fig. A.3: 2-opt* operator.

acceleration (e.g. Tovey (1985)) in the introduction of forbidden "*tabu*" moves and aspiration criteria capable of overriding these.

At each iteration of TS, a neighbourhood of solutions similar to the current solution are generated using one or more of the LSOs introduced in Section 3.2.2. Two strategies, Best Admissible (BA) and First Improvement (FI) can be used to determine which of the new solutions is actually chosen. BA searches every solution in the neighbourhood only implementing the move with the highest pay-off, whereas FI accepts the first move to produce an improvement in the solution. In both cases, the selected move must either not be tabu or satisfy the aspiration criterion, usually that the move produces an improvement in the solution.

Once made, moves are tabu for a number of subsequent iterations so that the same customer cannot be moved back and forth between routes. To help eliminate the related problem of cycles of moves being repeated, the duration of the tabu for a given move is set to a random number of iterations, typically in the range [5,15]. TS stops after a fixed number of iterations, either overall or from when the best solution was found. Pseudo code for the TS procedure is given in Algorithm A.1.1. Malca and Semet (2003) apply TS to the PDP with fixed fleet size.

A.1.2.1 Taburoute

Taburoute (Gendreau et al., 1994) uses a variable fitness function as discussed in Section 2.3 and otherwise follows a structure similar to TS. Taburoute uses, as its neighbourhood, all GENI routes of each vertex to its k nearest neighbours (see Section 3.2.1.4).

Algorithm A.1.1 Tabu Search

Precondition: Initial Solution S

```
1: repeat
2:     best move ← null
3:     repeat
4:         generate a move in the current neighbourhood.
5:         evaluate the move
6:         if move > best move or best move = null
7:             if move is not tabu or move satisfies aspiration criterion
8:                 best move ← move
9:                 if using FI
10:                    break
11:    until a good move is generated (BA/FI)
12:    make the best move
13:    update the tabu list
14: until stopping criterion met          ▷ iterations, total or since last improvement
```

Cordeau et al. (2001) extend Taburoute to the VRPTW and two of its generalizations, the periodic and multi-depot instances. Though it does not always produce the best possible solutions, Cordeau et al. (2001) find TS to be simple, robust and efficient in use, converging to good solutions quickly. Cordeau et al. (2004a) extend this work to handle the route duration constraint in VRPTW.

A.1.2.2 Tabu search with adaptive memory

Rochat and Taillard (1995) introduce the concept of adaptive memory where sections of the most promising solutions seen so far are kept and re-introduced into the trial solution when a local optimum is reached.

A.1.2.3 Tabu search for the VRP with soft time windows

Taillard et al. (1997) propose a technique for applying the TS heuristic of Glover (1990) to the VRP with soft time windows. Initial solutions are generated randomly, first inserting a single random customer into each route then adding remaining customers randomly, using a greedy insertion heuristic. These initial solutions are stored in an adaptive memory, as described by Rochat and Taillard (1995). The adaptive memory

splits the initial solutions into sub-tours and sorts these by their objective fitness. Initial solutions are then generated by choosing sub-tours from the adaptive memory, biased towards high fitness sub-tours, to include in a new route. When a sub-tour is selected other tours involving those customers are removed from consideration. This process repeats until all customers have been visited or all sub-tours used. Remaining customers are added to the route using a greedy insertion heuristic. A neighbourhood of candidate solutions is created using the CROSS LSO (Section 3.2.2). CROSS is used as it preserves the ordering of chains of customers in a given solution, which is important when the time ordering of those customers is a constraint. The tabu list stores only the fitness of the overall solution and its tabu tenure; whilst this could rule out viable alternate routes with identical distances, the likelihood of this is low when working with double precision costs and thousands of customers. An interesting observation is that the tabu tenure used in this approach is half of the number of iterations used by the algorithm which appears very high; also high is the number of entries in the tabu list, set at 100,000.

Rancourt et al. (2012) applies a TS based meta-heuristic to a long haul vehicle routing and scheduling problem applying the rules on truck driver safety for long-haul trips in North America. Similar rules exist for the UK and Europe and must be considered as part of Transfaction Ltd.'s Problem, see Chapter 4.

A.1.2.4 Tabu search for the dynamic VRP

Gendreau et al. (2006) rely on an ejection chain operator (Section 3.2.2) to rapidly solve instances of the dynamic VRP. An evaluation function is used to estimate the route duration change caused by adding a customer's request into a route. An adapted Floyd-Warshall shortest path algorithm (Ahuja et al., 1993) is used to solve the resultant TSPs of each route, vertices are ordered based on first improvement. An adaptive memory (Rochat and Taillard, 1995; Taillard et al., 1997) is again used. Solutions are added to the adaptive memory if they have a fitness greater than the fitness of the worst individual solution currently in the memory, the worst candidate being dropped. An adaptive descent heuristic, for instance defined as a TS from a solution derived from the adaptive memory down to a local minima, can be used to check against the members

of the memory to potentially replace one. A new solution is then created from the memory and the process repeated.

Solutions are decomposed into sub-problems each containing approximately the same number of routes and each is solved independently. Each subsequent solution can be decomposed and recombined multiple times before being stored in the adaptive memory.

A.1.3 Genetic Algorithms (GA)

Potvin and Guertin (1996), Homberger and Gehring (1999), Baker and Ayechew (2003) and Mester and Bräysy (2005, 2007) present Genetic Algorithms (GAs) for variations of the VRP. The approaches differ in details such as the way initial solutions are generated but follow the same structure of initial solution, population generation, mutation and parent replacement. Here, the method of Baker and Ayechew (2003) is presented. Solutions are stored as a list of customers, each storing a vehicle id. If the solution is changed, a TSP problem must be solved for each vehicle with its assigned customers as the nodes. An initial assignment is made using a sweep algorithm whereby nodes are added to the first vehicle in numerical order once they are sorted by polar angle around the base location, once a vehicle's route is full due to capacity constraints nodes are assigned to the next vehicle. Further parents are generated using a generalized assignment algorithm. Binary tournament selection is used to pick two parents at each iteration where the better of two candidates is chosen for each parent. Child solutions are generated using 2-point crossover as shown in Figure A.4. The population is preserved between runs, each child potentially replacing a parent from the population. An additional metric, referred to as unfitness, is defined as the excess weight, in violation of limits plus the excess distance as proportions of their respective allowable totals. The aim is of course to find good feasible solutions but allowing strong infeasible solutions may lead to similar solutions which are feasible. The worst solution is defined as the solution with the worst (highest) unfitness score, in the case of a tie the solution with the worst fitness and joint worst unfitness is chosen. The child solution will replace the worst known solution if it has better fitness or unfitness (or both).

Node	1	**2**	**3**	4
Parent 1	R1	R1	R2	R2
Parent 2	R1	R2	R1	R1

(a) Before

Node	1	***2***	***3***	4
Child 1	R1	***R2***	***R1***	R2
Child 2	R1	***R1***	***R2***	R1

(b) After

Fig. A.4: GA crossover operator, here the x in Rx is the route that a node belongs to in a solution.

A.2 Generated Times

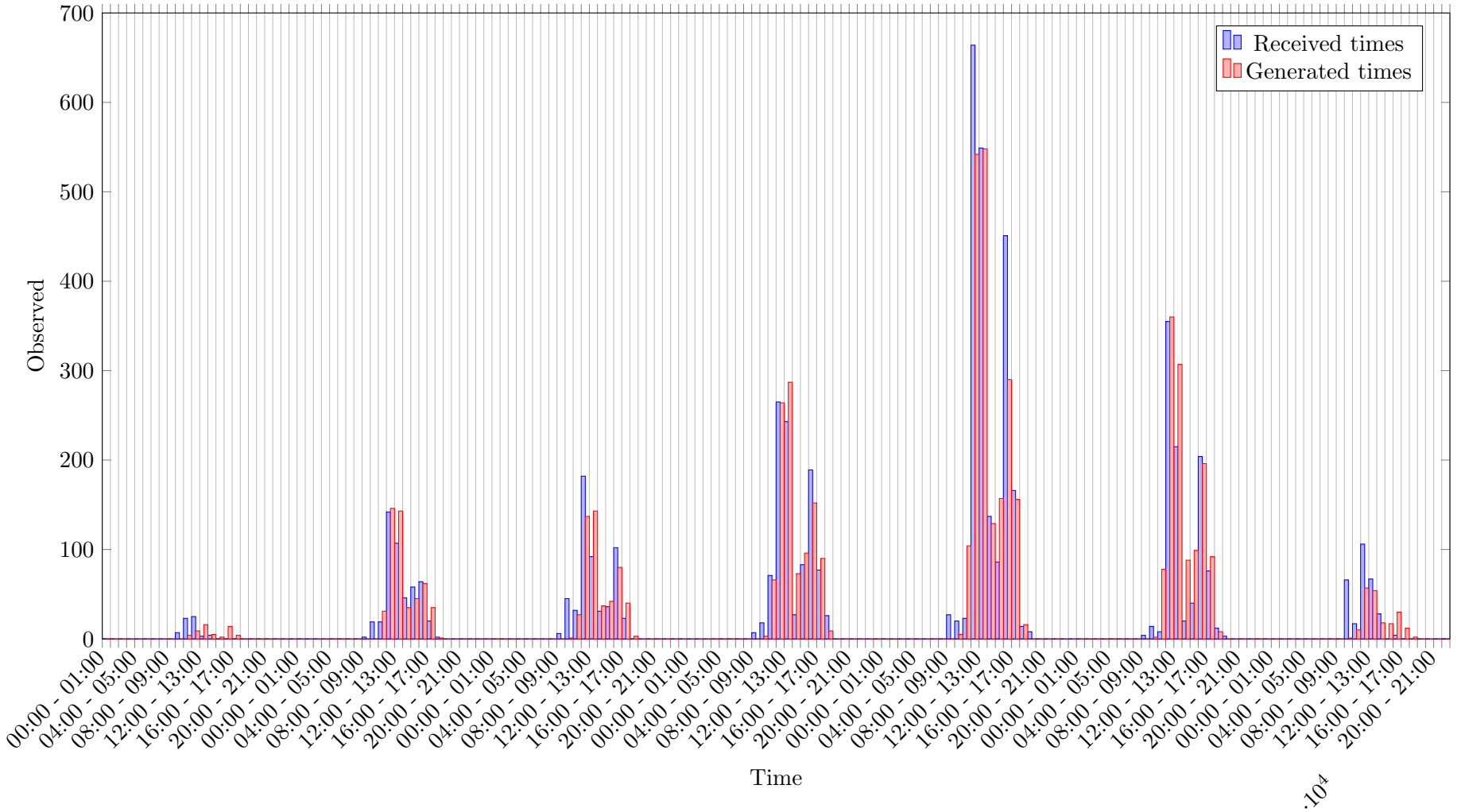


Fig. A.5: Consignment received times vs generated

A.3 Full Comparison Tables

The tables presented in this appendix represent the best result of 10 repeats for each instance/hyper-heuristic pair. Each run is given 5, 10 or 20 minutes depending on the size of the problem instance (100, 200 or 400 customers respectively). The results presented here represent over 1200 hours of CPU time.

Tab. A.1: HH Performance on 100 random customer benchmarks.

	Random		BEBO		VND		QL	
Name	r	d	r	d	r	d	r	d
LR1-1-1	**19**	**1650.80**	**19**	**1650.80**	**19**	**1650.80**	**19**	**1650.80**
LR1-1-2	**17**	**1487.57**	17	1520.82	**17**	**1487.57**	**17**	**1487.57**
LR1-1-3	**13**	**1292.67**	13	1314.63	**13**	**1292.67**	**13**	**1292.67**
LR1-1-4	9	1013.99	10	1046.74	**9**	**1013.39**	9	1013.99
LR1-1-5	**14**	**1377.11**	14	1384.37	**14**	**1377.11**	**14**	**1377.11**
LR1-1-6	**12**	**1252.62**	**12**	**1252.62**	**12**	**1252.62**	**12**	**1252.62**
LR1-1-7	**10**	**1111.31**	**10**	**1111.31**	**10**	**1111.31**	**10**	**1111.31**
LR1-1-8	**9**	**968.97**	**9**	**968.97**	**9**	**968.97**	**9**	**968.97**
LR1-1-9	12	1237.71	12	1241.13	**11**	**1208.96**	**11**	**1208.96**
LR1-1-10	**10**	**1159.35**	**10**	**1159.35**	**10**	**1159.35**	**10**	**1159.35**
LR1-1-11	**10**	**1108.90**	**10**	**1108.90**	**10**	**1108.90**	**10**	**1108.90**
LR1-1-12	9	1004.19	10	1068.61	**9**	**1003.77**	**9**	**1003.77**
LR2-1-1	4	1277.14	4	1377.16	**4**	**1257.37**	4	1279.90
LR2-1-2	4	1289.88	4	1347.52	**3**	**1197.67**	**3**	**1197.67**
LR2-1-3	3	1040.27	3	1245.81	**3**	**953.93**	3	991.77
LR2-1-4	3	1062.43	3	1110.92	**2**	**849.05**	3	1096.90
LR2-1-5	3	1066.66	3	1124.74	**3**	**1054.14**	3	1072.85
LR2-1-6	**3**	**944.65**	3	1008.77	3	1072.70	3	1127.18
LR2-1-7	3	1067.64	3	1073.86	**2**	**903.62**	3	1038.00
LR2-1-8	2	766.13	2	785.06	**2**	**741.30**	2	742.14
LR2-1-9	3	1110.63	3	1099.23	**3**	**930.59**	3	1087.78
LR2-1-10	3	1072.57	3	1075.63	**3**	**964.22**	3	1008.22
LR2-1-11	3	1045.21	3	1078.28	**3**	**907.86**	3	953.56

Tab. A.2: HH Performance on 100 random and clustered customer benchmarks.

	Random		BEBO		VND		QL	
Name	r	d	r	d	r	d	r	d
LRC1-1-1	**14**	**1708.80**	**14**	**1708.80**	**14**	**1708.80**	**14**	**1708.80**
LRC1-1-2	**12**	**1558.07**	**12**	**1558.07**	**12**	**1558.07**	**12**	**1558.07**
LRC1-1-3	**11**	**1258.74**	11	1271.08	**11**	**1258.74**	**11**	**1258.74**
LRC1-1-4	**10**	**1128.40**	10	1199.79	**10**	**1128.40**	10	1128.49
LRC1-1-5	**13**	**1637.62**	13	1640.30	**13**	**1637.62**	**13**	**1637.62**
LRC1-1-6	**11**	**1424.73**	12	1469.98	**11**	**1424.73**	**11**	**1424.73**
LRC1-1-7	**11**	**1230.14**	11	1320.78	**11**	**1230.14**	**11**	**1230.14**
LRC1-1-8	**10**	**1147.42**	11	1238.68	**10**	**1147.42**	**10**	**1147.42**
LRC2-1-1	4	1481.13	4	1795.36	**4**	**1455.54**	4	1535.12
LRC2-1-2	4	1505.85	4	1551.55	4	1424.71	**4**	**1402.95**
LRC2-1-3	3	1113.55	3	1117.09	3	1092.30	**3**	**1091.68**
LRC2-1-4	3	880.52	3	908.47	**3**	**825.85**	3	883.65
LRC2-1-5	4	1307.25	4	1375.83	**4**	**1306.41**	4	1458.93
LRC2-1-6	3	1229.70	3	1360.99	**3**	**1162.91**	**3**	**1162.91**
LRC2-1-7	3	1087.62	3	1064.40	**3**	**1062.05**	3	1079.65
LRC2-1-8	3	1021.27	3	1055.87	**3**	**862.94**	3	1043.37

Tab. A.3: HH Performance on 200 clustered customer benchmarks.

	Random		BEBO		VND		QL	
Name	r	d	r	d	r	d	r	d
LC1-2-1	**20**	**2704.57**	**20**	**2704.57**	**20**	**2704.57**	**20**	**2704.57**
LC1-2-2	**19**	**2764.55**	19	2774.57	**19**	**2764.55**	19	2764.79
LC1-2-3	18	2853.05	18	2946.48	**17**	**3465.66**	18	2956.02
LC1-2-4	17	3017.90	17	3206.38	**17**	**2777.46**	17	3032.49
LC1-2-5	**20**	**2702.05**	20	2782.52	**20**	**2702.05**	**20**	**2702.05**
LC1-2-6	**20**	**2701.03**	**20**	**2701.03**	**20**	**2701.03**	**20**	**2701.03**
LC1-2-7	**20**	**2701.03**	21	2987.74	**20**	**2701.03**	**20**	**2701.03**
LC1-2-8	20	2824.24	21	3586.17	**20**	**2767.79**	20	2806.49
LC1-2-9	18	2769.06	18	2794.84	**18**	**2724.24**	18	2725.45
LC1-2-10	18	3042.14	18	2919.08	**18**	**2820.23**	18	2971.62
LC2-2-1	**6**	**1931.44**	**6**	**1931.44**	**6**	**1931.44**	**6**	**1931.44**
LC2-2-2	6	2419.10	**6**	**2000.33**	6	2007.88	6	2070.36
LC2-2-3	6	2462.22	6	2287.21	**6**	**2151.67**	6	2287.92
LC2-2-4	6	2560.45	**6**	**2196.59**	6	2202.01	6	2416.52
LC2-2-5	7	2065.71	**6**	**1891.21**	**6**	**1891.21**	**6**	**1891.21**
LC2-2-6	7	2320.14	6	1869.70	**6**	**1857.78**	**6**	**1857.78**
LC2-2-7	7	2580.01	7	2406.91	**6**	**1875.01**	7	2119.85
LC2-2-8	7	2772.73	6	1983.09	**6**	**1932.41**	7	2531.51
LC2-2-9	7	2465.34	6	2064.67	**6**	**1861.09**	7	2289.21
LC2-2-10	7	2897.55	7	2316.77	**6**	**1961.18**	7	2592.19

Tab. A.4: HH Performance on 200 random customer benchmarks.

	Random		BEBO		VND		QL	
Name	r	d	r	d	r	d	r	d
LR1-2-1	**20**	**4819.12**	21	4967.33	**20**	**4819.12**	**20**	**4819.12**
LR1-2-2	18	4428.06	19	4376.67	**18**	**4394.91**	18	4416.82
LR1-2-3	15	3903.27	16	3885.18	15	3916.81	**15**	**3859.24**
LR1-2-4	11	3295.66	11	3372.85	**11**	**3035.14**	11	3366.11
LR1-2-5	17	4557.06	17	4876.26	**17**	**4439.55**	17	4505.99
LR1-2-6	15	4053.10	15	4495.60	15	4281.32	**14**	**4460.22**
LR1-2-7	13	3543.74	13	3818.84	**13**	**3476.60**	13	3630.14
LR1-2-8	10	3059.72	10	2956.80	**9**	**2966.93**	10	2936.38
LR1-2-9	15	4518.54	16	4583.60	**15**	**4416.90**	15	4429.27
LR1-2-10	12	3927.05	13	3845.36	**12**	**3742.21**	13	3790.15
LR2-2-1	5	4781.11	5	4637.96	**5**	**4337.34**	5	4637.38
LR2-2-2	5	5346.12	**4**	**4437.51**	4	4616.68	5	5084.45
LR2-2-3	4	5157.19	**4**	**4486.13**	4	4534.26	4	5101.29
LR2-2-4	3	3967.68	3	3464.04	**3**	**3159.53**	3	3793.70
LR2-2-5	4	3825.90	**4**	**3471.89**	4	3553.33	4	3629.85
LR2-2-6	4	4939.00	4	4688.85	**4**	**4200.50**	4	4957.57
LR2-2-7	4	4373.15	**3**	**3286.50**	3	4165.85	3	4055.87
LR2-2-8	3	3077.90	3	2830.92	**3**	**2724.29**	3	3299.12
LR2-2-9	4	4880.73	4	3665.64	**4**	**3206.65**	4	4820.68
LR2-2-10	4	4678.19	4	3824.13	**4**	**3663.49**	4	4293.19

Tab. A.5: HH Performance on 200 random and clustered customer benchmarks.

	Random		BEBO		VND		QL	
Name	r	d	r	d	r	d	r	d
LRC1-2-1	**19**	**3606.86**	20	3746.82	**19**	**3606.86**	19	3634.12
LRC1-2-2	17	3624.70	18	3519.59	**17**	**3359.18**	17	3536.45
LRC1-2-3	13	3531.11	14	3534.31	13	3355.96	**13**	**3293.08**
LRC1-2-4	10	2963.71	10	2983.57	**10**	**2914.47**	10	3050.71
LRC1-2-5	**17**	**3921.36**	17	4212.65	17	3941.56	17	3949.78
LRC1-2-6	17	3516.42	17	3617.96	**17**	**3396.61**	17	3426.43
LRC1-2-7	**16**	**3481.41**	16	3596.10	16	3498.94	16	3652.29
LRC1-2-8	**14**	**3274.37**	15	3395.99	14	3391.50	14	3391.68
LRC1-2-9	15	3197.68	15	3396.39	**14**	**3246.58**	15	3392.74
LRC1-2-10	13	3203.80	13	3167.27	**13**	**2907.99**	13	3042.46
LRC2-2-1	7	4140.95	7	3988.44	**7**	**3271.47**	7	4039.85
LRC2-2-2	6	4036.78	6	3482.21	**6**	**3452.40**	6	3538.54
LRC2-2-3	5	4069.68	5	3632.47	**5**	**3528.88**	5	3773.64
LRC2-2-4	4	3714.56	**4**	**2997.16**	4	2998.88	4	3753.64
LRC2-2-5	5	3814.98	**5**	**2954.58**	5	3439.36	5	3824.95
LRC2-2-6	5	3471.32	5	3121.70	**5**	**2947.10**	5	3139.43
LRC2-2-7	5	3610.49	5	2860.77	**5**	**2820.98**	5	3709.73
LRC2-2-8	5	3969.72	5	3295.88	**4**	**3175.80**	5	3583.75
LRC2-2-9	4	3822.54	4	2529.38	**4**	**2430.72**	4	3345.22
LRC2-2-10	4	3211.29	4	2144.63	**4**	**2138.04**	4	3122.07

Tab. A.6: HH Performance on 400 random customer benchmarks.

	Random		BEBO		VND		QL	
Name	r	d	r	d	r	d	r	d
LR1-4-1	**40**	**11008.85**	41	12051.26	40	11069.16	40	11464.51
LR1-4-2	34	10636.95	35	10639.12	34	10031.41	**33**	**10309.82**
LR1-4-3	26	9747.05	26	9677.49	**26**	**9206.40**	26	9685.00
LR1-4-4	**18**	**8771.16**	19	7514.84	19	7592.58	19	7861.49
LR1-4-5	33	10658.59	33	10740.28	**33**	**10354.37**	33	11237.52
LR1-4-6	29	10488.11	29	10603.00	**28**	**9873.44**	29	10367.68
LR1-4-7	23	9387.18	23	8929.60	**23**	**8590.37**	23	9102.70
LR1-4-8	17	7243.56	16	7091.69	**16**	**7064.28**	16	7376.67
LR1-4-9	27	10772.37	28	10716.57	**27**	**10195.71**	28	10777.67
LR1-4-10	24	9578.82	24	9371.65	**23**	**8943.90**	23	9169.52
LR2-4-1	10	14260.77	9	11605.04	**8**	**11201.38**	10	13622.01
LR2-4-2	9	14097.12	8	10515.79	**8**	**9520.44**	9	14126.43
LR2-4-3	7	12558.88	7	9257.95	**7**	**9177.06**	7	12572.50
LR2-4-4	5	9135.47	**5**	**6654.88**	5	7055.66	5	9831.22
LR2-4-5	8	12373.92	8	9914.27	**8**	**9701.25**	8	11617.47
LR2-4-6	7	12276.94	7	8940.58	**7**	**8823.15**	7	11976.53
LR2-4-7	6	11880.73	6	8586.93	**6**	**8524.23**	6	11779.42
LR2-4-8	5	9066.42	**5**	**7141.36**	5	7187.33	5	9768.87
LR2-4-9	8	12709.25	7	10287.89	**7**	**9962.42**	8	12730.21
LR2-4-10	7	11893.33	**6**	**7872.17**	6	9124.33	7	11690.58

Tab. A.7: HH Performance on 400 random and clustered customer benchmarks.

	Random		BEBO		VND		QL	
Name	r	d	r	d	r	d	r	d
LRC1-4-1	38	9624.02	39	9619.17	38	9211.06	**37**	**9484.95**
LRC1-4-2	35	8494.99	37	8549.04	**35**	**8194.06**	35	8395.42
LRC1-4-3	28	7953.54	28	7868.86	**27**	**7771.27**	27	7974.48
LRC1-4-4	20	6374.28	20	6551.66	**20**	**6344.24**	20	6368.56
LRC1-4-5	36	9347.46	38	9468.85	36	9178.22	**35**	**9183.38**
LRC1-4-6	**33**	**8575.21**	34	8635.36	33	8606.90	33	8604.10
LRC1-4-7	33	8813.13	34	8952.15	**33**	**8468.64**	33	8598.97
LRC1-4-8	31	8298.78	30	8617.80	**30**	**8203.63**	31	8333.45
LRC1-4-9	29	8580.96	30	8562.79	**29**	**8382.57**	30	8615.58
LRC1-4-10	27	7778.16	27	7773.59	**27**	**7545.90**	27	7901.50
LRC2-4-1	13	10041.06	**13**	**7233.26**	13	7307.14	13	9138.96
LRC2-4-2	13	10268.22	**12**	**7978.35**	12	8288.37	12	8854.54
LRC2-4-3	11	10963.19	**10**	**6485.29**	10	6511.94	11	9972.44
LRC2-4-4	7	8152.81	7	5295.52	7	5709.16	**6**	**8228.70**
LRC2-4-5	12	11458.99	11	8407.38	**11**	**7576.77**	12	10377.07
LRC2-4-6	11	9013.67	10	7215.13	**10**	**6979.27**	11	8904.62
LRC2-4-7	10	9919.95	9	7804.22	**9**	**7021.05**	10	8759.20
LRC2-4-8	9	10051.70	8	7078.69	**8**	**6874.11**	9	9845.77
LRC2-4-9	8	8507.99	**8**	**6463.79**	8	6567.16	8	8882.12
LRC2-4-10	8	9953.80	7	6827.80	**7**	**6354.33**	7	9631.84

Bibliography

Ahuja, R., Magnanti, T. and Orlin, J. (1993). *Network Flows: Theory, Algorithms, and Applications*. Book. Englewood Cliffs, New Jersey: Prentice Hall.

Albareda-Sambola, M., Fernández, E. and Laporte, G. (2014). The dynamic multi-period vehicle routing problem with probabilistic information. *Computers & Operations Research* 48(1): 31–39.

Augerat, P., Belenguer, J. M., Benavent, E., Corberán, Á., Naddef, D. and Rinaldi, G. (1998). Computational Results with a Branch and Cut Code for the Capacitated Vehicle Routing Problem. Technical report, Rapport de recherche- IMAG.

Baker, B. M. and Ayechew, M. (2003). A Genetic Algorithm for the Vehicle Routing Problem. *Computers & Operations Research* 30(5): 787–800.

Baldacci, R., Bartolini, E., Mingozzi, A. and Roberti, R. (2010). An exact solution framework for a broad class of vehicle routing problems. *Computational Management Science* 7(3): 229–268.

Beck, J., Prosser, P. and Selensky, E. (2003). Vehicle Routing and Job Shop Scheduling : What's the difference? In *Proceedings of the 13th International Conference on Artificial Intelligence Planning and Scheduling*: 267–276.

Belhaiza, S., Hansen, P. and Laporte, G. (2013). A hybrid variable neighborhood tabu search heuristic for the vehicle routing problem with multiple time windows. *Computers & Operations Research* 52 part B: 269–281.

Bellman, R. (1958). On a Routing Problem. *Quarterly of Applied Mathematics* 16: 87–90.

Benavent, E., Landete, M., Mota, E. and Tirado, G. (2015). The multiple vehicle pickup and delivery problem with LIFO constraints. *European Journal of Operational Research* 243(3): 752–762.

Bent, R. W. and Van Hentenryck, P. (2003). A Two-Stage Hybrid Algorithm for Pickup and Delivery Vehicle Routing Problems with Time Windows. In *Principles and Practice of Constraint Programming*. Springer Berlin Heidelberg: 123–137.

Bent, R. W. and Van Hentenryck, P. (2004). Scenario-Based Planning for Partially Dynamic Vehicle Routing with Stochastic Customers. *Operations Research* 52(6): 977–987.

Berbeglia, G., Cordeau, J.-F., Gribkovskaia, I. and Laporte, G. (2007). Static pickup and delivery problems: a classification scheme and survey. *Top* 15(1): 1–31.

Berbeglia, G., Cordeau, J.-F. and Laporte, G. (2010). Dynamic pickup and delivery problems. *European Journal of Operational Research* 202(1): 8–15.

Bianchi, L. (2000). Notes on dynamic vehicle routing-the state of the art. Technical report, Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA), Lugano, Switzerland.

Blocho, M. (2015). A Parallel Algorithm for Minimizing the Fleet Size in the Pickup and Delivery Problem with Time Windows. In *Proceedings of the 22nd European MPI Users' Group Meeting*. ACM: 20–21.

Blum, C., Puchinger, J., Raidl, G. R. and Roli, A. (2011). Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing* 11(6): 4135–4151.

Bodin, L., Golden, B. L., Assad, A. A. and Ball, M. (1983). The routing and scheduling of vehicles and crews The state of the art. *Computers & Operations Research* 10: 62–212.

Bortfeldt, A. and Homberger, J. (2013). Packing first, routing second: A heuristic for the vehicle routing and loading problem. *Computers & Operations Research* 40(3): 873–885.

Bouros, P., Sacharidis, D. and Dalamagas, T. (2011). Dynamic Pickup and Delivery with Transfers. In *Proceedings of the 12th International Symposium on Advances in Spacial and Temporal Databases*: 112–129.

Bräysy, O. (2002). Fast local searches for the vehicle routing problem with time windows. *INFOR* 40(4): 319–330.

Bräysy, O. (2003). A Reactive Variable Neighborhood Search for the Vehicle-Routing Problem with Time Windows. *INFORMS Journal On Computing* 15(4): 347–368.

Bräysy, O. and Gendreau, M. (2005a). Vehicle Routing Problem with Time Windows, Part I: Route Construction and Local Search Algorithms. *Transportation Science* 39(1): 104–118.

Bräysy, O. and Gendreau, M. (2005b). Vehicle Routing Problem with Time Windows, Part II: Metaheuristics. *Transportation Science* 39(1): 119–139.

Burke, E. K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E. and Qu, R. (2013). Hyper-heuristics: a survey of the state of the art. *Journal of the Operational Research Society* 64: 1695–1724.

Caric, T., Fosin, J., Galic, A., Gold, H. and Reinholz, A. (2007). Empirical Analysis of Two Different Metaheuristics for Real World Vehicle Routing Problems. In *International Workshop on Hybrid Metaheuristics*. Springer Berlin Heidelberg: 31–44.

Carrabs, F., Cordeau, J.-F. and Laporte, G. (2007). Variable Neighborhood Search for the Pickup and Delivery Traveling Salesman Problem with LIFO Loading. *INFORMS Journal on Computing* 19(4): 618–632.

Caseau, Y. (1999). Heuristics for Large Constrained Vehicle Routing Problems. *Journal of Heuristics* 5: 281–303.

Cheang, B., Gao, X., Lim, A., Qin, H. and Zhu, W. (2012). Multiple pickup and delivery traveling salesman problem with last-in-first-out loading and distance constraints. *European Journal of Operational Research* 223(1): 60–75.

Chen, Y., Cowling, P., Polack, F., Remde, S. and Mourdjis, P. (2017). Dynamic optimisation of preventative and corrective maintenance schedules for a large scale urban drainage system. *European Journal of Operational Research* 257(2): 494–510.

Chen, Y., Mourdjis, P., Polack, F. and Cowling, P. (2016a). Evaluating hyperheuristics and local search operators for period routing problem. In *Proceedings of the 16th European Conference on Evolutionary Computation in Combinatorial Optimisation*. Porto, PT: Springer International Publishing: 104–120.

Chen, Y., Polack, F., Cowling, P., Mourdjis, P. and Remde, S. (2016b). Risk Driven Analysis of Maintenance for a Large-scale Drainage System. In *Proceedings of the 5th International Conference on Operations Research and Enterprise Systems*.

Cherkesly, M., Desaulniers, G. and Laporte, G. (2015). Branch-Price-and-Cut Algorithns for the Pickup and Delivery Problem with Time Windows and LIFO Loading. *Computers & Operations Research* 62(1): 23–35.

Christofides, N. (1979). Combinatorial optimization. In *A Wiley-Interscience Publication, Based on a series of lectures, given at the Summer School in Combinatorial Optimization*. Sogesta, Italy.

Clarke, G. and Wright, J. W. (1964). Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. *Operations Research* 12(4): 568–581.

Cordeau, J.-F., Gendreau, M., Laporte, G., Potvin, J.-Y. and Semet, F. (2002). A guide to vehicle routing heuristics. *Journal of the Operational Research Society* 53(5): 512–522.

Cordeau, J.-F. and Laporte, G. (2003). A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B: Methodological* 37: 579–594.

Cordeau, J.-F., Laporte, G. and Mercier, A. (2001). A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational Research Society* 52(8): 928–936.

Cordeau, J.-F., Laporte, G. and Mercier, A. (2004a). Improved tabu search algorithm for the handling of route duration constraints in vehicle routing problems with time windows. *Journal of the Operational Research Society* 55(5): 542–546.

Cordeau, J.-F., Laporte, G., Potvin, J.-Y. and Savelsbergh, M. W. P. (2007). Transportation on Demand. *Handbooks in Operations Research and Management Science* 14(06): 429–466.

Cordeau, J.-F., Laporte, G. and Vigo, D. (2004b). *Short-Haul Routing*. Book. Montréal, Québec: Centre de recherche sur les transports (C.R.T.): 75.

Cordeau, J.-F. and Maischberger, M. (2012). A parallel iterated tabu search heuristic for vehicle routing problems. *Computers & Operations Research* 39(9): 2033–2050.

Cowling, P. I. and Johansson, M. (2002). Using real time information for effective dynamic scheduling. *European Journal of Operational Research* 139(2): 230–244.

Cowling, P. I., Kendall, G. and Soubeiga, E. (2001). A hyperheuristic approach to scheduling a sales summit. *PATAT III Springer LNCS* 2079: 176–190.

Cowling, P. I. and Keuthen, R. (2005). Embedded local search approaches for routing optimization. *Computers & Operations Research* 32(3): 465–490.

Cowling, P. I., Ouelhadj, D. and Petrovic, S. (2004). Dynamic scheduling of steel casting and milling using multi-agents. *Production Planning & Control* 15(2): 178–188.

Crainic, T. G., Nguyen, P. K. and Toulouse, M. (2015). Synchronized Multi-Trip Multi-Traffic Pickup & Delivery in City Logistics. *CIRRELT* 05(February): 1–24.

Dantzig, G. B. and Ramser, J. H. (1958). The Truck Dispatching Problem. *Management Science* 6(1): 80–92.

Demir, E., Bekta, T. and Laporte, G. (2012). An adaptive large neighborhood search heuristic for the Pollution-Routing Problem. *European Journal of Operational Research* 223(2): 346–359.

Demir, E., Bekta, T. and Laporte, G. (2013). A Review of Recent Research on Green Road Freight Transportation. *European Journal of Operational Research* 428(August).

Demir, E., Bekta, T. and Laporte, G. (2014). A review of recent research on green road freight transportation. *European Journal of Operational Research* 237(3): 775–793.

Department for Transport (2015). Load securing: vehicle operator guidance URL: `https://www.gov.uk/government/publications/load-securing-vehicle-operator-guidance` Date Accessed: 2015-06-15.

Department for Transport (2016). Drivers' Hours URL: `https://www.gov.uk/drivers-hours/overview` Date Accessed: 2015-06-15.

Desaulniers, G., Desrosiers, J., Solomon, M. M., Erdmann, A. and Soumis, F. (2002). VRP with pickup and delivery. In P. Toth and D. Vigo, eds., *The vehicle routing problem*. SIAM: 225–242.

Desrochers, M., Desrosiers, J. and Solomon, M. M. (1992). A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows. *Operations Research* 40: 342–354.

Desrosiers, J., Dumas, Y., Solomon, M. M. and Soumis, F. (1995). Time Constrained Routing and Scheduling. *Handbooks in Operations Research and Management Science* 8: 35–139.

Desrosiers, J., Dumas, Y. and Soumis, F. (1986). A Dynamic Programming Solution of the Large-Scale Single-Vehicle Dial-A-Ride Problem with Time Windows. *American Journal of Mathematical and Management Sciences* 6(3-4): 301–325.

Dff International Ltd. (2014). RHA Cost Tables URL: `http://www.rha.uk.net/docs/CostTables2014EDITION.pdf` Date Accessed: 2015-06-15.

Dff International Ltd. (2015). RHA National Directory of Hauliers URL: `https://www.findahaulier.co.uk/` Date Accessed: 2015-06-15.

Diaz, B. D. (2006). The VRP Web URL: `http://www.bernabe.dorronsoro.es/vrp/` Date Accessed: 2017-02-03.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik* 1: 269–271.

Dorer, K. and Calisti, M. (2005). An Adaptive Approach to Dynamic Transport Optimization. In F. Klugl, A. Bazzan and S. Ossowski, eds., *Applications of Agent Technology in Traffic and Transportation*. Birkhäuser Basel: 33–49.

Dumas, Y., Desrosiers, J. and Soumis, F. (1991). The pickup and delivery problem with time windows. *European Journal of Operational Research* 54(1): 7–22.

Erera, A., Karacik, B. and Savelsbergh, M. (2008). A dynamic driver management scheme for less-than-truckload carriers. *Computers and Operations Research* 35(11): 3397–3411.

Fahrion, R. and Wrede, M. (1990). On a Principle of Chain-exchange for Vehicle-routeing Problems (1-VRP). *The Journal of the Operational Research Society* 41(9): 821–827.

Faulin, J. and Juan, A. (2007). ALGACEA-2: An entropy-based heuristics for the Capacitated Vehicle Routing Problem. In *Seventh Metaheuristics International Conference*.

Figliozzi, M. A. (2010). The impacts of congestion on commercial vehicle tour characteristics and costs. *Transportation Research Part E: Logistics and Transportation Review* 46(4): 496–506.

Figliozzi, M. A., Mahmassani, H. S. and Jaillet, P. (2003). Framework for Study of Carrier Strategies in Auction-Based Transportation Marketplace. *Transportation Research Record* 1854(03): 162–170.

Figliozzi, M. A., Mahmassani, H. S. and Jaillet, P. (2007). Pricing in Dynamic Vehicle Routing Problems. *Transportation Science* 41(3): 302–318.

Fisher, M. (1995). Network Routing. *Handbooks in Operations Research and Management Science* 8: 1–33.

Ford Jr., L. R. (1956). Network Flow Theory. *RAND CORP, Santa Monica, California* (No. P-923).

Fox, B. L. (1993). Integrating and Accelerating Tabu Search, Simulated Annealing, and Genetic Algorithms. *Annals of Operations Research* 41(2): 47–67.

Gambardella, L. M., Taillard, É. D. and Agazzi, G. (1999). MACS-VRPTW: A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows. *IDSIA* : 1–17.

Gehring, H. and Homberger, J. (1999). A Parallel Hybrid Evolutionary Metaheuristic for the Vehicle Routing Problem with Time Windows. *Proceedings of EUROGEN99* 2: 57–64.

Gendreau, M., Guertin, F., Potvin, J.-Y. and Séguin, R. (2006). Neighborhood Search Heuristics for a Dynamic Vehicle Dispatching Problem with Pick-ups and Deliveries. *Transportation Research Part C: Emerging Technologies* 14(3): 157–174.

Gendreau, M., Hertz, A. and Laporte, G. (1992). New Insertion and Post Optimization Procedures for the Traveling Salesman Problem. *Operations Research* 40(6): 1086–1095.

Gendreau, M., Hertz, A. and Laporte, G. (1994). A Tabu Search Heuristic for the Vehicle Routing Problem. *Management Science* 40(10): 1276–1290.

Gendreau, M., Laporte, G. and Potvin, J.-Y. (2002). Metaheuristics for the capacitated VRP. In P. Toth and D. Vigo, eds., *The vehicle routing problem*. Philadelphia, PA: SIAM Monographs on Discrete Mathematics and Applications. SIAM Publishing: 129–154.

Glover, F. (1990). Artificial Intelligence, Heuristic Frameworks and Tabu Search. *Managerial and Decision Economics* 11(5): 365–375.

Glover, F. (1996). Ejection chains, reference structures and alternating path methods for traveling salesman problems. *Discrete Applied Mathematics* 65(94): 223–253.

Golden, B. L. and Assad, A. A. (1988). *Vehicle Routing: Methods and Studies*. Book. Amsterdam: North-Holland.

Gschwind, T. and Irnich, S. (2012). Effective Handling of Dynamic Time Windows and Synchronization with Precedences for Exact Vehicle Routing. Technical report, Johannes Gutenberg University Mainz: 30, Mainz, Germany.

Haghani, A. and Jung, S. (2005). A dynamic vehicle routing problem with time-dependent travel times. *Computers & Operations Research* 32(11): 2959–2986.

Hansen, P. and Mladenović, N. (2001). Variable neighborhood search : Principles and applications. *European Journal of Operational Research* 130: 449–467.

Hansen, P., Mladenović, N. and Moreno Pérez, J. a. (2009). Variable Neighbourhood Search: Methods and Applications. *Annals of Operations Research* 175(1): 367–407.

Hasle, G., Lie, K.-A. and Quak, E., eds. (2007). *Geometric Modelling, Numerical Simuation, and Optimization. Applied Mathematics at SINTEF*. Berlin Heidelberg: Springer-Verlag: 258.

Hentenryck, P., Bent, R. W. and Upfal, E. (2009). Online stochastic optimization under time constraints. *Annals of Operations Research* 177(1): 151–183.

Homberger, J. and Gehring, H. (1999). Two Evolutionary Meta-Heuristics for the Vehicle Routing Problem with Time Windows. *INFOR* 37(3): 297–318.

Homberger, J. and Gehring, H. (2005). A two-phase hybrid metaheuristic for the vehicle routing problem with time windows. *European Journal of Operational Research* 162(1): 220–238.

Horn, M. E. T. (2002). Fleet scheduling and dispatching for demand-responsive passenger services. *Transportation Research Part C: Emerging Technologies* 10(1): 35–63.

Hosny, M. I. (2010). *Investigating Heuristic and Meta-Heuristic Algorithms for Solving Pickup and Delivery Problems*. Ph.D. thesis, Cardiff University.

Jarboui, B., Derbel, H., Hanafi, S. and Mladenović, N. (2013). Variable neighborhood search for location routing. *Computers & Operations Research* 40(1): 47–57.

Karp, R. M. (1972). Reducibility Among Combinatorial Problems. In *Complexity of Computer Computations*. Springer US: 85–103.

Koç, Ç., Bekta, T., Jabali, O. and Laporte, G. (2014). A Hybrid Evolutionary Algorithm for Heterogeneous Fleet Vehicle Routing Problems with Time Windows. *Computers & Operations Research* 64(March): 11–27.

Koning, D. (2011). *Using Column Generation for the Pickup and Delivery Problem with Disturbances*. Masters thesis, Universiteit Utrecht.

Kontoravdis, G. and Bard, J. F. (1995). A GRASP for the vehicle routing problem with time windows. *ORSA journal on Computing* 7(1): 10–23.

Krumke, S. O., Rambau, J. and Torres, L. M. (2002). Real-Time Dispatching of Guided and Unguided Automobile Service Units with Soft Time Windows. In R. Möhring and R. Raman, eds., *Proceedings of the 10th Annual European Symposium on Algorithms*. Springer-Verlag: 637–648.

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

Laporte, G. (2009). Fifty Years of Vehicle Routing. *Transportation Science* 43(4): 408–416.

Laporte, G. and Osman, I. H. (1995). Routing Problems: A Bibliography. *Annals of Operations Research* 61(1): 227–262.

Larsen, A. (2001). *The Dynamic Vehicle Routing Problem*. Ph.D. thesis, Technical University of Denmark.

Lee, W. L. (2013). Real-Life Vehicle Routing with Non-Standard Constraints. *Proceedings of the World Congress on Engineering* I: 3–8.

Lenstra, J. K. and Kan, a. H. G. R. (1981). Complexity of Vehicle Routing and Scheduling Problems. *Networks* 11(2): 221–227.

Li, F., Golden, B. L. and Wasil, E. (2005). Very large-scale vehicle routing: new test problems, algorithms, and results. *Computers & Operations Research* 32(5): 1165–1179.

Li, H. and Lim, A. (2003). A Metaheuristic for the Pickup and Delivery Problem with Time Windows. *International Journal of Artificial Intelligence Tools* 12(02): 173–186.

Li, J.-Q., Mirchandani, P. B. and Borenstein, D. (2009). Real-time vehicle rerouting problems with time windows. *European Journal of Operational Research* 194(3): 711–727.

Li, Y., Lim, A., Oon, W. C., Qin, H. and Tu, D. (2011). The tree representation for the pickup and delivery traveling salesman problem with LIFO loading. *European Journal of Operational Research* 212(3): 482–496.

Lim, A., Zhang, Z. and Qin, H. (2016). Pickup and Delivery Service with Manpower Planning in Hong Kong Public Hospitals. *Transportation Science* .

Lim, A. and Zhu, W. (2006). A Fast and Effective Insertion Algorithm for Multi-depot Vehicle Routing Problem with Fixed Distribution of Vehicles and a New Simulated Anealing Approach. *Advances in Applied Artificial Inteligence* 4031: 282–291.

Lin, S. (1965). Computer Solutions of the Traveling Salesman Problem. *Bell System Tech. J.* : 44:22452269.

Lorini, S., Potvin, J.-Y. and Zufferey, N. (2011). Online vehicle routing and scheduling with dynamic travel times. *Computers & Operations Research* 38(7): 1086–1090.

Malandraki, C. and Daskin, M. S. (1992). Time Dependent Vehicle Routing Problems: Formulations, Properties and Heuristic Algorithms. *Transportation Science* 26: 185–200.

Malca, F. and Semet, F. (2003). A tabu search heuristic for the pickup and delivery problem with time windows and a fixed size fleet. *Unpublished manuscript* : 1–5.

McLeod, F., Cherrett, T., Shingleton, D., Bekta, T., Speed, C., Davies, N., Dickinson, J. and Norgate, S. (2012). 'Sixth Sense Logistics: Challenges in supporting more flexible, human-centric' scheduling in the service sector. In *Annual Logistics Research Network (LRN) Conference*. Cranfield, UK.

Mester, D. and Bräysy, O. (2005). Active guided evolution strategies for large-scale vehicle routing problems with time windows. *Computers & Operations Research* 32(6): 1593–1614.

Mester, D. and Bräysy, O. (2007). Active-guided evolution strategies for large-scale capacitated vehicle routing problems. *Computers & Operations Research* 34(10): 2964–2975.

Mitrović-Minić, S., Krishnamurti, R. and Laporte, G. (2004). Double-horizon based heuristics for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B: Methodological* 38(8): 669–685.

Mladenović, N. and Hansen, P. (1997). Variable Neighbourhood Search. *Computers & Operations Research* 24(1): 1097–1100.

Moon, I., Lee, J.-H. H. and Seong, J. (2012). Vehicle routing problem with time windows considering overtime and outsourcing vehicles. *Expert Systems with Applications* 39(18): 13202–13213.

Mourdjis, P. J., Chen, Y., Polack, F., Cowling, P. I. and Robinson, M. (2016a). Variable Neighbourhood Descent with Memory: A Hybrid Metaheuristic for Supermarket Resupply. In M. J. Blesa, C. Blum, A. Cangelosi, V. Cutello, A. Di Nuovo, M. Pavone and E.-G. Talbi, eds., *Proceedings of the 10th International Workshop on Hybrid Metaheuristics*. Switzerland: Springer International Publishing: 32–46.

Mourdjis, P. J., Cowling, P. I. and Robinson, M. (2014). Metaheuristics for the pick-up and delivery problem with contracted orders. In C. Blum and G. Ochoa, eds., *Proceedings of the 14th European Conference on Evolutionary Computation in Combinatorial Optimisation*. Granada, ES: Springer-Verlag: 170–181.

Mourdjis, P. J., Polack, F., Cowling, P. I., Chen, Y. and Robinson, M. (2016b). The effect of cooperation in pickup and multiple delivery problems. In *Proceedings of the 5th International Conference on Operations Research and Enterprise Systems*: 287–295.

Nagata, Y. (2007). Edge Assembly Crossover for the Capacitated Vehicle Routing Problem. In *Proceedings of the 7th European Conference on Evolutionary Computation in Combinatorial Optimization*. Springer Berlin Heidelberg: 142–153.

Nagata, Y. and Bräysy, O. (2008). Efficient local search limitation strategies for vehicle routing problems. In J. Hemert and C. Cotta, eds., *Lecture Notes in Computer Science*. Berlin Heidelberg: Springer-Verlag: 48–60.

Nagata, Y. and Bräysy, O. (2009). Edge Assembly-Based Memetic Algorithm for the Capacitated Vehicle Routing Problem. *Networks* 54: 20–22.

Nagata, Y. and Kobayashi, S. (1997). Edge Assembly Crossover: A High-power Genetic Al- gorithm for the Traveling Salesman Problem. In *Proceedings of the 7th International Conference on Genetic Algorithms*: 450–457.

Nahum, O. E. (2013). *The Real-Time Multi-Objective Vehicle Routing Problem*. Ph.D. thesis, Bar-Ilan University.

Osman, I. H. (1993). Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operations Research* 41(4): 421–451.

Ostertag, A., Doerner, K. F. and Hartl, R. F. (2008). A variable neighborhood search integrated in the POPMUSIC framework for solving large scale vehicle routing problems. In *International Workshop on Hybrid Metaheuristics*. Springer Berlin Heidelberg: 29–42.

Ouelhadj, D. and Petrovic, S. (2009). A survey of dynamic scheduling in manufacturing systems. *Journal of Scheduling* 12(4): 417–431.

Paraskevopoulos, D. C., Repoussis, P. P., Tarantilis, C. D., Ioannou, G. and Prastacos, G. P. (2008). A reactive variable neighborhood tabu search for the heterogeneous fleet vehicle routing problem with time windows. *Journal of Heuristics* 14(5): 425–455.

Parragh, S. N., Doerner, K. F. and Hartl, R. F. (2009). Variable Neighborhood Search for the Dial-a-Ride Problem. *Computers & Operations Research* 37(6): 1129–1138.

Pidd, M. (1998). *Computer Simulation in Management Science*. Book. John Wiley & Sons, Inc.

Pillac, V., Gendreau, M., Guéret, C. and Medaglia, A. L. (2013). A review of dynamic vehicle routing problems. *European Journal of Operational Research* 225(1): 1–11.

Pillac, V., Guéret, C. and Medaglia, A. L. (2012). A fast re-optimization approach for dynamic vehicle routing. *Technical Report* hal-007397: 1–22.

Pirkwieser, S. and Raidl, G. R. (2009). Multiple variable neighborhood search enriched with ilp techniques for the periodic vehicle routing problem with time windows. In *International Workshop on Hybrid Metaheuristics*. Springer Berlin Heidelberg: 45–59.

Pisinger, D. and Ropke, S. (2007). A general heuristic for vehicle routing problems. *Computers & Operations Research* 34(8): 1–44.

Polacek, M., Hartl, R. F. and Doerner, K. F. (2004). A Variable Neighborhood Search for the Multi Depot Vehicle Routing Problem with Time Windows. *Journal of Heuristics* 10: 613–627.

Potvin, J.-y. and Guertin, F. (1996). A Genetic Algorithm for Vehicle Routing with Backhauling. *Applied intelligence* 6: 345 – 355.

Potvin, J.-Y. and Rousseau, J.-M. (1995). An Exchange Heuristic for Routeing Problems with Time Windows. *Journal of the Operational Research Society1* 46(12): 1433–1446.

Quintiq (2015). PDPTW World Records URL: `http://www.quintiq.com/optimization/pdptw-world-records.html` Date Accessed: 2016-01-01.

Rancourt, M.-E., Cordeau, J.-F. and Laporte, G. (2012). Long-Haul Vehicle Routing and Scheduling with Working Hour Rules. *Transportation Science* 47(1): 0–37.

Remde, S., Cowling, P. I., Dahal, K., Colledge, N. and Selensky, E. (2011). An empirical study of hyperheuristics for managing very large sets of low level heuristics. *Journal of the Operational Research Society* 63(3): 392–405.

Repoussis, P. P., Paraskevopoulos, D. C., Tarantilis, C. D. and Ioannou, G. (2006). A reactive greedy randomized variable neighborhood tabu search for the vehicle routing problem with time windows. In *International Workshop on Hybrid Metaheuristics*. Springer Berlin Heidelberg: 124–138.

Robu, V., Noot, H., La Poutré, H. and van Schijndel, W.-J. J. (2011). A multi-agent platform for auction-based allocation of loads in transportation logistics. *Expert Systems with Applications* 38(4): 3483–3491.

Rochat, Y. and Taillard, É. D. (1995). Probabilistic Diversification And Intensification In Local Search For Vehicle Routing. *Journal of Heuristics* 1(1): 147–167.

Ropke, S. (2005). *Heuristic and exact algorithms for vehicle routing problems*. Ph.D. thesis, University of Copenhagen.

Ropke, S. and Cordeau, J.-F. (2009). Branch and Cut and Price for the Pickup and Delivery Problem with Time Windows. *Transportation Science* 43(3): 267–286.

Ropke, S. and Pisinger, D. (2005). An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows. *Transportation Science* 40(4): 455–472.

Savelsbergh, M. W. P. (1992). The Vehicle Routing Problem with Time Windows: Minimizing Route Duration. *INFORMS Journal on Computing* 4(2): 146–154.

Savelsbergh, M. W. P. and Sol, M. (1998). Drive: Dynamic Routing of Independent Vehicles. *Operations Research* 46(4): 474–490.

Sbihi, A. and Eglese, R. W. (2010). Combinatorial optimization and Green Logistics. *Annals of Operations Research* 175(1): 159–175.

Sexton, T. R. and Choi, Y.-M. (1986). Pick-up and delivery of partial loads with time windows. *American Journal of Mathematical and Management Sciences* 6(3-4): 369–398.

Shaw, P. (1998). Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems. In *Principles and Practice of Constraint Programming*: 417–431.

Sintef (2008). Transportation optimization portal - PDPTW Benchmarks URL: `http://www.sintef.no/pdptw` Date Accessed: 2017-01-01.

Solomon, M. M. (1987). Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. *Operations Research* 35(2): 254–265.

Sontrop, H., van der Horn, P., Uetz, M., Horn, P. V. D., Uetz, M., van der Horn, P. and Uetz, M. (2005). Fast Ejection Chain Algorithms for Vehicle Routing with Time Windows. In *Proceedings of the 2nd International Workshop on Hybrid Metaheuristics*: 78–89.

Stobart, E. (2013). LGV C+E (Class 1) - Flexi - Various Locations URL: `http://goo.gl/ogOJA0` Date Accessed: 2015-06-15.

Taillard, É. D. (1993). Parallel Iterative Search Methods for Vehicle Routing Problems. *Networks* 23(8): 661–673.

Taillard, É. D., 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): 170–186.

Takes, F. W. and Kosters, W. A. (2010). Applying Monte Carlo Techniques to the Capacitated Vehicle Routing Problem. In *Proc. BeNeLux Conf. Artif. Intell.*: 25–26.

Talbi, E.-G. (2009). *Metaheuristics: From Design to Implementation*. Hoboken, NJ, USA: John Wiley & Sons, Inc, vol. 74 .

Tarantilis, C. D. (2005). Solving the vehicle routing problem with adaptive memory programming methodology. *Computers & Operations Research* 32(9): 2309–2327.

Tepić, J., Tanackov, I. and Stojić, G. (2011). Ancient Logistics - Historical Timeline and Etymology. *Technical Gazette* 18(3): 379–384.

TetraSoft, A. (2003). MapBooking Algoritm for Pickup and Delivery Solutions with Time Windows and Capacity restraints. URL: `http://www.tetrasoft.dk/english-info/` Date Accessed: 2015-10-01.

Thomas, B. W. and White III, C. C. (2004). Anticipatory Route Selection. *Transportation Science* 38(4): 473–487.

Thompson, P. M. and Psaraftis, H. N. (1993). Cyclic transfer algorithm for multivehicle routing and scheduling problems. *Operations Research* 41(5): 935–946.

Toth, P. and Vigo, D. (1997). Heuristic Algorithms for the Handicapped Persons Transportation Problem. *Transportation Science* 31(1): 60–71.

Toth, P. and Vigo, D., eds. (2002). *The Vehicle Routing Problem*. SIAM Monographs on Discrete Mathematics and Applications. SIAM Publishing.

Tovey, C. A. (1985). Hill climbing with multiple local optima. *SIAM Journal on Algebraic Discrete Methods* 6(3): 384–393.

Waisanen, H. A., Shah, D. and Dahleh, M. A. (2007). Fundamental Performance Limits for Multi-Stage Vehicle Routing Problems. *Unpublished manuscript* (OPRE-2007-08-388): 1–24.

Watkins, C. J. C. H. and Dayan, P. (1992). Q-learning. *Machine Learning* 8(3-4): 279–292.

Xu, H., Chen, Z.-L., Rajagopal, S. and Arunapuram, S. (2003). Solving a Practical Pickup and Delivery Problem. *Transportation Science* 37(3): 347–364.

Yang, X., Strauss, A. K., Currie, C. and Eglese, R. W. (2014). Choice-Based Demand Management and Vehicle Routing in E-fulfilment. *Transportation Science* 50(2): 473–488.

Zhu, K. (2004). Information Transparency of Business-to-Business Electronic Markets: A Game-Theoretic Analysis. *Management Science* 50(5): 670–685.

Zhu, K. Q. and Ong, K.-L. (2000). A reactive method for real time dynamic vehicle routing problem. *IEEE International Conference on Tools with Artificial Intelligence* 12: 176–180.