Martin Damyanov Aleksandrov

Master Thesis

Heuristics and Policies for Online Pickup and Delivery Problems

Orientador : Pedro Barahona, Professor, CENTRIA, FCT, UNL, Lisbon, Portugal

Co-orientadores : Philip Kilby, Researcher, NICTA, Canberra, Australia
Toby Walsh, Group Leader, NICTA, Sydney, Australia

President : José Júlio Alves Alferes
Main referee : Paula Alexandra da Costa Amaral
Referee : Pedro Manuel Corrêa Calvente Barahona

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

15 October, 2012

Heuristics and Policies for Online Pickup and Delivery Problems

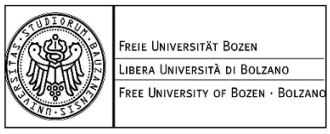
Martin Damyanov Aleksandrov

Department of Informatics, Faculty of Sciences and Technology,
Univesidade Nova de Lisboa

Advisor : Prof. Pedro Barahona, CENTRIA, UNL, Lisbon, Portugal

co-Advisor 1 : Dr. Philip Kilby, NICTA, Canberra, Australia
co-Advisor 2 : Prof. Toby Walsh, NICTA, Sydney, Australia

Submission date : 15 **October 2012**



Declarations

I, Martin Damyanov Aleksandrov, certify that this master thesis has been written entirely by me, that is a record of work carried out by me, it has not been submitted in any previous higher degree application and it will not be submitted in any future higher degree application.

I, Martin Damyanov Aleksandrov below referred to as **the Student**, certify that this piece of work has been conducted in both the Universidade Nova de Lisboa, Portugal, below referred to as **the Institution 1**, between February 2012 and October 2012, and the National Information and Communications Technology Research Center of Excellence, NICTA, Australia, below referred to as **the Institution 2**, between October 2011 and January 2012. Attention has been taken with respect to both the study and graduate regulations in **the Institution 1** and the working regulations imposed by **the Institution 2** as follows:

1. According to the common reguirements for obtaining a higher-education degree, posted by **the Institution 1**, this work is submitted to and public available at the Department of Informatics, Faculty of Sciences and Technology, which is part of **the Institution 1**.

2. According to the conditions of the Visiting Researcher Agreement contract between **the Institution 2** and **the Student**, started at 1 October 2011 and finished at 18 January 2012, the Intellectual Property created during the course of this Agreement is property of **the Institution 2**.

Date : Signature of the Student :

I, Pedro Barahona, hereby certify that **the Student** has fulfilled the conditions of the regulations appropriate for the degree of European Master in Computational Logic in **the Institution 1** and that **the Student** is qualified to submit the thesis in application for that degree.

Date : Signature of the advisor :

I, Philip Kilby, hereby certify that **the Student** has fulfilled the conditions of the regulations mentioned in the Visting Researcher Agreement contract and that **the Student** is qualified to submit the thesis in application for that degree.

Date : Signature of the co-advisor :

In submitting the thesis to the Universidade Nova de Lisboa we understand that we are giving permission for it to be mabe available for use in accordance with the regulations of the University Library for the time being in force, subject to any copyright vested in the work not being affected thereby. We also understand that the title and the abstract will be published, and that a copy of this work will be submitted to the Academic Archive, to the Advisor and to the co-Advisor.

Access to Printed copy of the thesis through the University Nova de Lisboa.

Date : Signature of the Student :

Date : Signature of the advisor :

Preface

This Master thesis entitled *"Heuristics and Policies for Online Pickup and Delivery Problems"* has been prepared by Martin Damyanov Aleksandrov during the period October 2011 to October 2012 at the National ICT Australia and at the Universidade Nova de Lisboa. Martin Damyanov Aleksandrov visited both places under the common study and mobility regulations of the European Master Program in Computational Logic.

The thesis work was partially supported by the National ICT Australia according to the Visitor Research Agreement contract between NICTA and Martin Damyanov Aleksandrov. The subject of the thesis is to develop new online dynamic algorithms for dispatching the fleet of vehicles for a well-known subclass of Vehicle Routing Problems, viz. Pickup and Delivery Problems, and to extend the current research in this area by possibly contributing to some basic research problems in intelligent transport systems with potential application to areas like traffic control, vehicle routing and logistics. It is motivated by a problem arised in an Australian courier firm and it aims to achieve and to satisfy their necessities as much as possible.

Next, I would like to thank my thesis advisor, Pedro Barahona, who is a professor at Universidade Nova de Lisboa for his support and interest throughout this project. I have now served as Pedro's master student during the last year and I have enjoyed our collaboration together.

I would like to thank professor Toby Walsh and doctor Philip Kilby, who are research leaders at the NICTA's Neville Roach and Canberra laboratories, respectively. They were supporting me during the whole time by giving usefull pieces of advice, opinions and comments on the theoretical and practical aspects of this thesis.

In addition, I wish to specially thank doctor Philip Kilby for providing me technical information about the state-of-the-art Vehicle Routing Solver, Indigo 2.0, which is implemented by him and it is a property of NICTA. I really appreciate his efforts and usefull pieces of advice during the last one year.

Many special thanks to Hanna Grzybowska for providing the NICTA internal reports, collected on June 30, September 1 and September 6, 2011. They were very usefull in modelling and understanding the addressed problem in terms of human, fleet and customer resources as well as in providing specific details about the working conditions in the company.

I would like also to express thanks to Menkes van den Briel, who is an operational research professional in the optimization group at NICTA. Our discussions significantly contributed for better understanding the addressed problem.

Lastly, as a master student in such an European program I spent most of my time travelling around the world. During all this experience my family, Damyan, Svetla and Iliyan, and my friends were my moral support and I wish to thank them for encouraging me.

Martin Damyanov Aleksandrov

Abstract

In the last few decades, increased attention has been dedicated to a specific subclass of Vehicle Routing Problems due to its significant importance in several transportation areas such as taxi companies, courier companies, transportation of people, organ transportation, etc. These problems are characterized by their dynamicity as the demands are, in general, unknown in advance and the corresponding locations are paired. This thesis addresses a version of such Dynamic Pickup and Delivery Problems, motivated by a problem arisen in an Australian courier company, which operates in Sydney, Melbourne and Brisbane, where almost every day more than a thousand transportation orders arrive and need to be accommodated. The firm has a fleet of almost two hundred vehicles of various types, mostly operating within the city areas. Thus, whenever new orders arrive at the system the dispatchers face a complex decision regarding the allocation of the new customers within the distribution routes (already existing or new) taking into account a complex multi-level objective function.

The thesis thus focuses on the process of learning simple dispatch heuristics, and lays the foundations of a recommendation system able to rank such heuristics. We implemented eight of these, observing different characteristics of the current fleet and orders. It incorporates an artificial neural network that is trained on two hundred days of past data, and is supervised by schedules produced by an oracle, Indigo, which is a system able to produce suboptimal solutions to problem instances. The system opens the possibility for many dispatch policies to be implemented that are based on this rule ranking, and helps dispatchers to manage the vehicles of the fleet. It also provides results for the human resources required each single day and within the different periods of the day. We complement the quite promising results obtained with a discussion on future additions and improvements such as channel fleet management, traffic consideration, and learning hyper-heuristics to control simple rule sequences.

Resumo

Nas últimas décadas, tem sido dedicada uma crescente atenção a uma subclasse específica de problemas de roteamento de veículos, devido à sua importância significativa em diversas áreas de transporte, tais como empresas de táxis, empresas de courier, transporte de pessoas, transporte de órgãos, etc. Estes problemas são caracterizados por sua dinâmica, já que os pedidos envolvendo pares de localizações são, em geral, desconhecidos com antecedência. Esta tese trata de uma versão de Problema de Recolha e Entrega Dinâmico, motivado por um problema surgido numa empresa de entregas australiana, que atua em Sydney, Melbourne e Brisbane, e em que todos os dias são feitos mais de mil pedidos de transporte de entregas. A empresa tem uma frota de quase 200 veículos de vários tipos, e opera principalmente dentro das áreas urbanas. Sempre que um novo pedido de entrega chega ao sistema, os despachantes enfrentam uma decisão complexa quanto à alocação dos novos pedidos dentro das rotas de distribuição (já existentes ou novas), tendo em conta uma função complexa função multi-objetivo.

A tese centra-se no processo de aprendizagem de regras de despacho simples, e lança as bases de um sistema de recomendação capaz de classificar as heurísticas existentes. Oito dessas foram implementadas, tendo em conta as características das encomendas e do estado corrente de alocação da frota e encomendas. O sistema incorpora uma rede neural artificial, treinada por dados referentes a 200 dias de actividade, e supervisionada por um oráculo, Indigo, que é um sistema capaz de produzir soluções sub-ótimas para instâncias do problema. O sistema abre a possibilidade de implementação de variadas políticas de despacho baseadas numa ordenação destas regra heurísticas, e fornece indicações para os recursos humanos necessários em cada dia, e em diferentes períodos do dia. Os resultados obtidos são bastante promissores e são complementados com uma discussão sobre futuras adições e melhorias, tais como canais de gestão de frotas, análise de tráfego, e aprendizagem de hiper-heurísticas para controlar seqüências de regras simples.

List of Abbreviations

1. BAL - Balanced Heuristic.
2. CDVRP - The Capacitated Dynamic Vehicle Routing Problem.
3. CVRP - The Capacitated Vehicle Routing Problem.
4. CUR - Current Orders Heuristic.
5. DARP - The Dial-a-Ride Problem.
6. DPDP - The Dynamic Pickup and Delivery Problem.
7. DPDPTW - The Dynamic Pickup and Delivery Problem with Time Windows.
8. dDVRP - The Delivery Dynamic Vehicle Routing Problem.
9. DVRP - The Dynamic Vehicle Routing Problem.
10. DVRPTW - The Dynamic Vehicle Routing Problem with Time Windows.
11. GEO - Geographical Closeness Heuristic.
12. GIS - Graphical Information Systems.
13. IMM - Immediate Cost Heuristic.
14. MAV - Minimize Vehicles Heuristic.
15. MIN - Minimum Cost Heuristic.
16. ML - Machine Learning.
17. NN - Neural Network.
18. ODPDP - The Online Dynamic Pickup and Delivery Problem.
19. ODPDPTW - The Online Dynamic Pickup and Delivery Problem with Time Windows.
20. OSDPDPTW - The Online Stochastic and Dynamic Pickup and Delivery Problem with Time Windows.

21. PDP - The Pickup and Delivery Problem.
22. PDPTW - The Pickup and Delivery Problem with Time Windows.
23. PDTSP - The Pickup and Delivery Travelling Salesman Problem.
24. pDVRP - The Pickup Dynamic Vehicle Routing Problem.
25. RAND - Random Heuristic.
26. RL - Reinforcement Learning.
27. SHIFT - Shift Profitability Heuristic.
28. STDPDP - The Stochastic and Dynamic Pickup and Delivery Problem.
29. STDVRP - The Stochastic and Dynamic Vehicle Routing Problem.
30. STVRP - The Stochastic Vehicle Routing Problem.
31. SVRP - The Static Vehicle Routing Problem.
32. TSP - The Traveling Salesman Problem.
33. UDVRP - The Uncapacitated Dynamic Vehicle Routing Problem.
34. VRP - The Vehicle Routing Problem.

List of Tables

1.1 Sydney fleet characteristics. 5
1.2 Bonds standart services. 5
1.3 Bonds extra services. 6

2.1 VRP constraints. 21
2.2 Paired and time window constraints. 22
2.3 Time slices scenarios. 24

3.1 Route constraints : regarding the arrival moment. 32
3.2 Route constraints : regarding the new order capacities. 32
3.3 Minimum Cost constraints. 35
3.4 Balanced constraints. 40
3.5 Current Orders constraints. 42
3.6 Shift Profitability constraints. 43
3.7 Geographical Closeness constraints. 45
3.8 Minimize Vehicles constraints. 46
3.9 Immediate Cost constraints. 47
3.10 Weaken constraints. 48
3.11 Worst-case complexity assesments. 50

4.1 Real and seeming order statistics. 57
4.2 Once-a-day pooling strategy : cost statistics. 67
4.3 Once-a-day pooling strategy : time performance. 68
4.4 Once-a-day pooling strategy : global valuation statistics. 69
4.5 Time-zones pooling strategy : global valuation statistics. 69
4.6 Fixed-time-span pooling strategy : global valuation statistics. 69
4.7 Time-zones pooling strategy : global vs. local statistics. 71
4.8 Fixed-time-span pooling strategy : global vs. local statistics. 71

5.1 Demand features. 76
5.2 Schedule current features. 77

List of Figures

1.1 The distribution of the arriving orders within a single weekday. 3
1.2 A specification of Vehicle Routing Problems. 11

2.1 Non-homogenous and periodic counting process. 25

3.1 Rasterized area of Sydney. 44

4.1 Datasets. 53
4.2 Order arrival distributions. 53
4.3 Regression matrix. 55
4.4 Arrivals and weight distributions. 56
4.5 Comparison of the original and the generated datasets. 57
4.6 Evaluation procedure. 59

5.1 Neural network general structure. 75
5.2 Recommendation system scheme. 77
5.3 Global learning features. 78
5.4 Global and local learning features. 79
5.5 Policy comparison in *once* pooling strategy. 81
5.6 Fleet management under π_{PG} policy. 82
5.7 Policy comparison in *time-zones* pooling strategy. 84

Contents

1 Introduction **1**
- 1.1 Motivation . . . 2
  - 1.1.1 Fleet and service description . . . 4
  - 1.1.2 Objectives . . . 6
  - 1.1.3 Summary of our problem as ODPDP . . . 7
- 1.2 History Notes . . . 8
- 1.3 Specification of VRPs . . . 10
- 1.4 The Vehicle Routing Solver Indigo . . . 11
- 1.5 Thesis contributions . . . 13
- 1.6 Thesis Structure . . . 14

2 A Framework for DPDP **15**
- 2.1 The Problem Formulation . . . 16
- 2.2 The Basic VRP . . . 20
- 2.3 Paired Customers and Time Windows . . . 22
- 2.4 Switching to a Dynamic Setting . . . 23
  - 2.4.1 Time slices . . . 24
  - 2.4.2 Prediction model . . . 24
- 2.5 Objective Function . . . 25
- 2.6 NP-hardness . . . 28

3 Simple Dispatch Heuristics **29**
- 3.1 General Assumptions . . . 30
  - 3.1.1 Possible routes . . . 32
  - 3.1.2 Capacity constraints . . . 32
  - 3.1.3 Additional travel costs . . . 33
  - 3.1.4 Time window urgency . . . 33
  - 3.1.5 Distance measures . . . 34
- 3.2 Heuristics . . . 35
  - 3.2.1 Minimum cost heuristic . . . 35
  - 3.2.2 Balanced heuristic . . . 40
  - 3.2.3 Current orders heuristic . . . 41
  - 3.2.4 Shift profitability heuristic . . . 42
  - 3.2.5 Geographical closeness heuristic . . . 44
  - 3.2.6 Minimize vehicles heuristic . . . 45

3.2.7 Immediate cost heuristic . . . 47
3.2.8 Random heuristic . . . 48
3.2.9 Weaken heuristic constraints . . . 48
3.3 Summary . . . 49

4 Heuristic Experiments **51**
4.1 General Assumptions . . . 52
4.2 Benchmark Datasets . . . 52
4.2.1 Original Dataset . . . 52
4.2.2 Arrival distributions . . . 53
4.2.3 Regression analyses . . . 54
4.2.4 Predictor distributions . . . 55
4.2.5 Generated datasets . . . 56
4.3 An Experimental Framework . . . 58
4.3.1 Initial schedule . . . 58
4.3.2 Suboptimal schedule . . . 58
4.3.3 Evaluation procedure . . . 59
4.3.4 Request sequences . . . 60
4.3.5 Solutions and partial solutions . . . 60
4.3.6 Heuristic correctness . . . 62
4.4 Experimental Results . . . 63
4.4.1 Once-a-day strategy . . . 64
4.4.2 Time-zones strategy . . . 64
4.4.3 Fixed-time-span strategy . . . 65
4.4.4 Cost . . . 66
4.4.5 Time performance . . . 67
4.4.6 Correctness . . . 68
4.4.7 Global vs. local correctness . . . 70
4.5 Summary . . . 72

5 Learning Experiments **73**
5.1 Machine Learning . . . 73
5.2 Neural Network . . . 74
5.3 Features, Datasets and System Scheme . . . 76
5.4 Dispatch Policies . . . 79
5.4.1 Policy *MaxSumPG* . . . 79
5.4.2 Policy *MinSumPG* . . . 80
5.4.3 Policy *MaxSumPLG* . . . 80
5.4.4 Policy *MinSumPLG* . . . 80
5.5 Experimental Results . . . 80
5.6 Summary . . . 85

6 Future Perspective **86**
6.1 Cross-utilization 87
6.2 Traffic Congestion 88
6.3 Hyper-heuristics 88
6.4 Channel fleet management 88

A Appendix References **90**

Bibliography **95**

Chapter 1

Introduction

Introduction is a new beginning.
by Martin Aleksandrov

The chapter starts with the presentation of the motivational background behind our work by describing the problem source and its characteristics. We map the task in our attention into a problem of the well-known subclass of Vehicle Routing Problems (VRPs), namely Pickup and Delivery Problems (PDPs). As we deal with a courier company, where the new demands arrive dynamically, it would be beneficial to take into consideration their dynamicity as well as to consider a prediction model of these demands, which tells us when and where a new errand would occur with some degree of certainty.

We bring the reader closer to the class of VRPs by surveying some of the relevant literature sources and by discussing some of the efforts put in such problems. The history notes do not give a total overview of such problems but we attemp to track the research line in this area along the years and give an additional motivation to our project. In parallel, we highlight the features of our work in order to support it with a more clear distinction from the previous work.

The next section presents a specification of the VRPs. In the past many people have specified these problems and we do not argue that our criterion is a total or an unique one, rather we present it to narrow the attention of the reader to the task we deal with. We capture our characterization of the VRPs with a figure, depicting the relations between the different subclasses and underlining the assumptions made within each one of them.

We continue this introductory chapter with the description of the oracle used along the thesis, namely VRP Solver Indigo 2.0. It has been implemented in NICTA and it is a property of NICTA. We present the general architecture phases of Indigo and we shortly discuss the methodologies used in each one of them. Finally, we give a brief summary of the thesis contributions and thesis structure.

1.1 Motivation

A local courier firm has approached NICTA about providing a decision support tool to schedule their pick-up and deliveries. Bonds Express is a part of Bonds Transport Group, which has been established in 1966 and has recently grown to become a specialist multidiscipline, Australian transport and 3PL provider. It is privately owned and it is dedicated to providing a total quality transport solution to its customers. It operates fast, secure and reliable express courier and taxi-truck services mostly within the areas of Sydney, Melbourne and Brisbane. Offering a full range of services, Bonds has the ability to meet standart and urgent delivery needs. They also offer highly competative rates for interstate and international deliveries[1].

Next, we describe the scope of the problem arised in Bonds Express Couriers. The dispatchers in the company control six channels of information flow between the customers, requesting services, and the drivers executing these services. Each dispatcher manages one channel, which does not correspond to a region, rather to a range of vehicle types. Currently, the dispatchers estimate themselves the most probable service time per job. The role of the dispatchers is of essential importance as we can view them as the distribution points in such information flows. At the same time they obey certain constraints related to their working conditions, namely they have a limited scheduling horizon and they cannot exchange information between each other. The drivers can refuse to do an assigned job and they are very selective and picky about the tasks to be performed. Currently, the order of the requests to be serviced is decided by the drivers, which means they participate in constructing their own route. The dispatchers wish to have a higher control on this order and to navigate the drivers towards their next destinations more precisely and routinely.

The information about vehicle locations arrive at the center dispatching unit via GPS every 3 minutes. The other direction of communication is realized via mobile text messages, which can be ambigious. Therefore, the dispatchers wish to have an improved and frequent communication with the drivers. The company has a policy of accepting all incoming job requests, even though they know they might not be able to satisfy the customer on-time requirements. In this case, the penalty for arriving too late is reflected in the profit for making the particular job. They pay the driver from the moment he picks up the first order till the moment they drop off the last one. They must pay the minimum hourly rate to the driver and keep the right to send him back home at any time. Once a driver is sent home he is no more available to the dispatchers on that day.

The fleet obeys specific limitations not only with respect to the physical characteristics of its vehicles, but also with respect to the driver contracts. According to the current working conditions, drivers are allowed to work a maximum of 8 hours per day. If this time is exceeded some extra regulations are in force, which are not within our scope. Thus, each vehicle has a fixed time working horizon. In the next section, we present additional information about the fleet characteristics, i.e. type, commodities, average velocity per kilometre and an estimate of the average running cost.

[1] `http://www.bondscouriers.com.au/`

The demands themselves must obey several restrictions. They should be serviced within limited response time, which depends on the customer as well as on the type of the chosen service. Recall, that our orders are paired and composed of a pick-up and delivery requests. Thus, a natural constraint is having the pick-up location to precede the delivery one. Besides the geographical and the time preference data attached to all the incoming requests, they also include information about their commodity characteristics. In practice, this would allow us to build loading and unloading process specifications for each vehicle. The latter is important as it could significantly reduce the service time at a customer location and further optimize our objectives.

During a working day many dynamic orders arrive at the company. As expected, in the different parts of the day the order arrival frequency is different and, consequently, the total workload of the fleet would differ in these periods. Based on the distribution of the new order arrivals along the day we define four time zones. Each of them has specific characteristics such as distribution, average job commodity characteristics, most used type of vehicle within a particular zone, average travel time between clients and others, which we discuss later on. The proposed definition of time zones, which is based on the provided data is depicted in Figure 1.1 [1], where time t_i is the moment the zone i starts. The division is as follows : *Night* (time zone 1), *Morning pick hours* (time zone 2), *Day* (time zone 3) and *Evening pick hours* (time zone 4). Figure 1.1 also shows the demand arrival rate during a single working day and it also includes the orders requested for the following days.

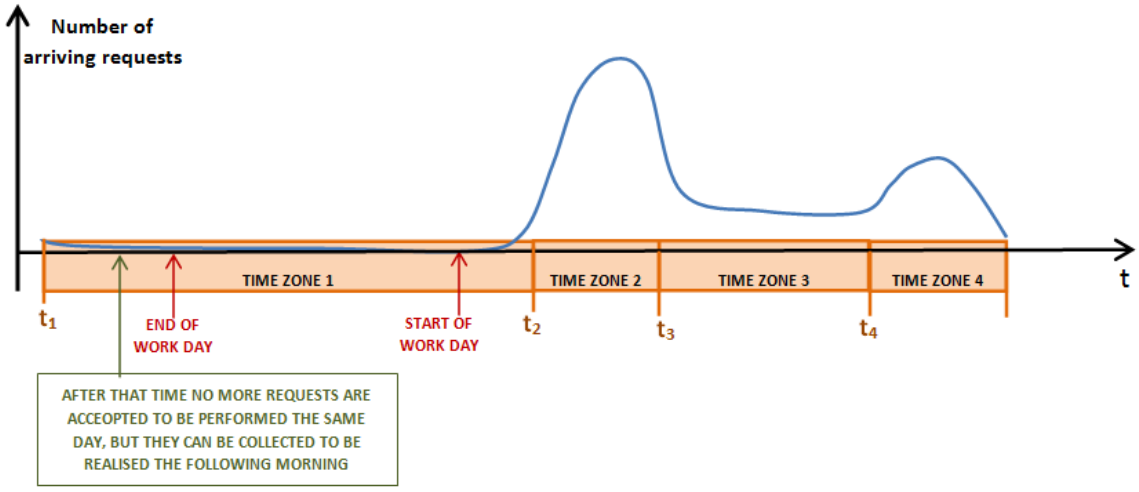


Figure 1.1: The distribution of the arriving orders within a single weekday.

Using this distribution we can prepare a routing and scheduling plan for the entire day, for each time zone, for a fixed time span or whenever a predefined event happens. This recalculation of the plan can be made on the basis of additional information, either revealed to the

[1] According to the NICTA internal report from June 30, 2011.

decision maker or based on historical basis. Once specified the frequency of rerouting and rescheduling of the current plan, the rules regarding the diversion of freight vehicles need to be defined. For modelling purposes, it has been assumed that the drivers can communicate with the dispatchers via communication devices installed only at client locations. As a consequence, while they are serving a client they can be informed about the changes in the routing and scheduling plan. However, each vehicle must perform the lastly assigned service.

Once a shipping parcel has been picked up, its delivery location has to be visited by the same vehicle in the same day. Hence before modifying the current sequence of clients to visit a list of compulsory delivery customers needs to be created, which cannot be shifted to any other route, but can be visited in an order different from the originally established. Although, we do not assume a possible cross-utilization amongst vehicles, in practice it might be an important issue.

1.1.1 Fleet and service description

In our work we concentrate on the area around Sydney, which is lying between -33.4 and -34.4 latitude, and between 150.67 and 151.67 longitude coordinates. It covers the main metropolitan area as well as its surrounding regional centers. The core Sydney fleet is comprised of approximately 200 vehicles that range from CBD bicycles to vans and flat-tops to 14 tonne trucks and semi-trailers. We summarize the types of vehicles operating within the Sydney urban network in the Table 1.1 [1]. There are 17 types of vehicles, each one described with the following features :

- vehicle **type**
- vehicle **code**
- maximum number of **pallets** a type is able to accommodate
- maximum **load** a type is able to accommodate
- **number** of available vehicles of the given type
- estimate of the **average cost** per kilometer the company pays to have a vehicle of that type on the road
- **average velocity** of a vehicle

The cost was estimated taking into account the type of vehicle, its average gas and/or oil consumption per a hundred kilometres, the additional expenses when a vehicle is full and the average gas price within the Sydney area for the months between April and June, 2012 [2]. The velocity considered is according to the urban city speed regulations with respect to the vehicle type, assumed an area without traffic congestion.

[1] According to the NICTA internal report from June 30, 2011.

[2] `http://www.carbonblack.com.au/dealer/50/cheapest-petrol-prices.aspx`

#	Type of vehicle	Vehicle code	Max. weight [kgs]	Max. pallets	Number	Cost [$/km]	Vehicle speed [km/h]
1	Pushbike	PB	2	0	6	0.02	15
2	Motorbike	MB	25	0	6	0.07	35
3	Car	CAR	75	0	4	0.12	50
4	Station Wagon	SW	250	0	8	0.17	50
5	Small Van	SV	500	0	5	0.26	50
6	Large Van	1V	1000	0	68	0.4	50
7	1 Tonne Tray	1T	1000	2	34	0.44	50
8	2 Tonne Tray	2T	2000	4	18	0.87	50
9	2 Tonne Van	2V	2000	2	4	0.73	50
10	2 Tonne Pan	2P	2000	2	2	0.73	50
11	4 Tonne Tray	4T	4000	6	10	1.5	50
12	4 Tonne Taut	4TA	4000	6	1	1.5	50
13	6 Tonne Tray	6T	6000	8	4	2.3	50
14	8 Tonne Tray	8T	8000	10	17	2.95	50
15	12 Tonne Tray	12T	12000	12	3	4.76	50
16	14 Tonne Tray	14T	14000	14	1	5.49	50
17	14 Tonne Pan	14P	14000	14	1	5.49	50

Table 1.1: Sydney fleet characteristics.

Standard Service	Maximum Delivery time
Standard Courier	3 hours
Priority Courier	1 hour and 55 minutes
Express Guaranteed	arrangement on the phone

Table 1.2: Bonds standart services.

The fleet tries to satisfy each customer necessity through a variety of standard and extra services. The standard services are summarized in Table 1.2 together with the maximum delivery times within which they should be performed, and the extra services are described in Table 1.3[1]. The actual average delivery times are significantly less than the maximum quoted, although these are subject to traffic and weather conditions. Extra time of approximately 25% should be allowed for meeting deadlines of deliveries of more than 30 kilometres driving distance and to destinations outside the metropolitan area[2]. Recall, that the conditions of the type of service are discussed with the customer when the demand is being requested and they are incorporated in his time preferences.

[1] According to the NICTA internal report from June 30, 2011.

[2] `http://www.bondscouriers.com.au/`

Extra Service	Description
Bicycle Courier	fast, reliable and economic service; consignment size and weight limitations
Motorbike Courier	faster than a car or a van at normal courier charges; covers limited area and includes consignment size and weight limitations
Taxi Truck Service	slower than most of the fleet member and at higher charges in both travel and service aspects; significant consignment size and weight capacity
Taxi Truck Priority	faster than Taxi Truck Service, but on a higher hourly and kilometer rate
Intrastate, Interstate and Overseas	same day, overnight air and road freight

Table 1.3: Bonds extra services.

1.1.2 Objectives

Here we list the objectives the company is willing to achieve. As this master thesis is a part of a bigger project, whose final goal is to deliver an end-user product to the company, i.e. a decision supporting tool, we concentrate on those aims more relevant to our problem.

1. The minimum number of requests assigned to a vehicle type in order to ensure profitability for the shift.
2. The time of active performance of a vehicle type, i.e. without waiting times.
3. The average number of requests per driver per vehicle type.
4. The average schedule duration per driver per vehicle type.
5. The degree of dynamism of the addressed problem.
6. The maximization of simultaneous utilisation of vehicles for multiple deliveries.
7. The improvement of the overall fleet utilization.
8. The capacity of any vehicle that should not be exceeded at any time.
9. The satisfaction of the customers, which is improved by ensuring more on-time services.
10. The quality of the scheduling plan, i.e. the number of late jobs performed a given day.

11. The number of contracted drivers in relation with the jobs performed for a single day. The objective is to minimize the former and to maximize the latter.

12. The number of drivers to be contracted at the beginning of the working day and the number of those sent home along the day.

13. The order, in which a driver visits the clients.

14. The final cost to be minimized.

1.1.3 Summary of our problem as ODPDP

In this subsection we extract all relevant information from the informal specification given above in order to concentrate on the problem in hands. We restrict the problem to the Sydney fleet of the company and services only within the metropolitan area and its surronding regional centers. We consider a heterogenous fleet of 192 vehicles, each with a type, capacities, running cost and running velocity described in Table 1.1. Each vehicle has a home location which is where the day starts and ends. Once a vehicle leaves its depot, the firm pays a cost per hour until the vehicle returns at the end of its shift. We consider a shift length of minimum 1 and maximum 8 hours per day. Also each vehicle type has a given capacity in terms of weight and number of pallets. There are hard constraints on the total weight and the total number of pallets that can be carried at any one time.

The problem is dynamic as the orders can arrive at any time. Each order is described by an order id, order arrival time, order weight, order pallets, pick-up location, earliest pick-up time, latest pick-up time, delivery location, and latest delivery time. The earliest pick-up time is a hard constraint as the package is assumed not to be ready before this time. We may require vehicles to wait at a location till the earliest pick-up time constraint is satisfied. The latest pick-up and delivery times are soft constraints and there are piecewise linear penalty terms in the objective for violating these constraints. There are service times for each location to account the time needed to collect or deliver the requested parcel. Due to the lack of more precise information about the fleet and order commodity dimensions we could not build loading and unloading specifications for the vehicle types, and assume 3 minutes of service time for goods less that 250 kilograms and 10 minutes [1], otherwise. The pick-up and delivery times are defined by the customer. They express his preferences and can be arranged during the booking time. The type of service needed is also taken into account when these preferences are discussed. In addition, we suppose orders are scheduled within a single day, assuming that some of them have been requested some days before. In reality, we may hold requests at the end of the day for the next morning, afternoon, evening or even for several subsequent days.

Importantly, the routing is supposed to be online. Whilst we may have a tentative schedule for all current orders, we only commit to a pick-up or delivery when the previous demand has been executed. A vehicle can be diverted at a client location, but not along its way between any two consecutive visits. The driver may consult with a dispatcher about his

[1] According to the NICTA internal report from June 30, 2011.

next destination via PDA device. Finally, we do not allow any load to be stored at any of the depot, customer or intermediate locations. In practice, a vehicle could deliver a package to some place, called *store*, where another or even the same vehicle will arrive later and collect the particular good. The last would be an interesting extension to our work, but since our focus is centered on learning dispatch decisions we do not consider this issue.

1.2 History Notes

A Vehicle Routing Problem (VRP) can be defined as a problem of finding the optimal routes of delivery or collection from one or several depots to a number of cities or customers, while satisfying some constraints. Collection of household waste, gasoline delivery trucks, goods distribution, snowplough and mail delivery are the most used applications of the VRP. The VRP plays a vital role in distribution and logistics. Huge research efforts have been devoted to studying the VRP since 1954 when Dantzig and Ramser [17] have described the problem as a generalised problem of Travelling Salesman Problem (TPS). Since this point onwards a tremendous amount of research work has been concentrated on the comparison between practically expensive exact methods and heuristic approaches. Some of the work on this subject after the eighties are Christofides et al., 1981 [12], who used spanning tree and shortest path relaxations to solve a number of instances derived from the literature. Desrochers, Lenstra and Savelsbergh in 1990 [18] (Laporte, 1992 [42]) surveyed main exact and approximate algorithms developed for a VRP, at a level appropriate for a first graduate course in combinatorial optimization. Later on Baldacci et al. in 2008 [5] introduced an exact algorithm for the capacitated version of a VRP (CVRP) based on the set partitioning formulation and additional cuts that correspond to capacity and clique inequalities in the VRP graph. Baldacci also discussed some recent advances the same year (2008) in [4]. Apart from the classical formulation and its variants also many efforts have been made in modelling specific optimization problems by means of VRP. For instance, Dong et al. 2011 described a variant of VRP in Flight Ticket Sales Companies for the service of free pickup and delivery of airline passengers to the airport (see [20]).

An important characteristic of a VRPs is whether the information of the demands is known in advance. From this perspective, the class of VRPs can be split into Static VRPs (Berbeglia et al., 2007 [7]) and Dynamic VRPs (Psaraftis, 1988 [51], Kilby et al., 1998 [38] and Larsen et al., 2002 [44]). The latter type captures some specificities appearing in the real-case studies, which usually a static approach disregards. The main difference is that in the static problem all the information about the demands is assumed to be available in advance. On the contrary, a dynamic instance tries to capture the dynamic behaviour of these client requests, as they typically arrive as the day progresses. Hence, its problem instances must be solved a large number of times and, moreover, in a reasonable time. Especially, in the areas such as taxi-companies, urgent transportation services (people (DARP, Cordeau and Laporte, 2003 [14]), orgar freights (Awasthi and Sandholm, 2009 [3]), etc.), etc. many Static VRP must be solved and further analyzed. The last could be a difficult task due to the NP-hard nature of these problems. Thus, many researchers and practitioners divert their attention towards heuristic and look-ahead approaches (Mes et al., 2010 [45]) for the DVRP. Many papers have been written on this VRP variant (Mitrović-Minić et al.

2004, [41]). Also the limited knowledge of the incoming demands motivated people to take a close look over the heuristics for Stochastic VRPs (STVRP). Swihart and Papastavrou [55], 1999, considered demands arriving according to a Poisson process. In 2006, Ichoua ([32]) exploited a strategy based on a probabilistic knowledge about future requests in order to predict where such a stochastic event would occur. Such approaches usually improve the fleet management. Later on Hvattum et al. [31], 2007, implemented a Branch-and-Regret heuristic for Stochastic and Dynamic VRP (STDVRPs). Several other practically important variants such as DVRP with Time Windows (DVRPTW), DVRP with Pick-Ups (pDVRP), DVRP with Deliveries (dDVRP), DVRP with Pick-Ups and Deliveries (DPDP, Parragh et al. 2008 [50]) and Capacitated DVRP (CDVRP, Kopmanz et al. 2001 [40], Ganapathy et al. 2009 [23], Chandran and Raghavan, 2008 [11]) as well as Uncapacitated DVRP (Angelelli et al. 2007 [1]) have also been investigated thoroughly.

The class closer to this thesis is a variant of DPDP. Its instances are characterized with additional considerations regarding the demands, e.g. the deliveries are paired and they must be executed in a particular order (i.e. the pickup location to be visited before the delivery one). In other words, objects or people have to be transported between an origin and a destination. Moreover, issues related to the order arrival dynamicity and to anticipating future demands should be taken into account, and therefore we consider Stochastic and Dynamic PDP (STDPDP). Furthermore, the class of PDPs can be classified into three different groups. The first group consists of *many-to-many* problems, in which any vertex can serve as a source or as a destination for any commodity. An example of a many-to-many problem is the Swapping Problem (Anily and Hassin, 1992 [2]). In this problem, every vertex may initially contain an object of a known type of commodity as well as a desired type of commodity. The problem consists of constructing a route performing the pickups and deliveries of the objects in such a way that at the end of the route, every vertex possesses an object of the desired type of commodity. Problems in the second group are called *one-to-many-to-one* problems. In these problems commodities are initially available at the depot and are destined to the customer vertices; in addition, commodities available at the customers are destined to the depot. Finally, in *one-to-one* problems, each commodity (which can be seen as a request) has a given origin and a given destination. Problems of this type arise, for example, in courier operations and door-to-door transportation services. Thus, our problem can be classified as *one-to-one* PDP. In addition, we assume customer preferences on time windows (PDPTW, Mitrović-Minić, 1998 [47] and DPDPTW, Mitrović-Minić et al., 2004 [43]) and stochasticity (STDPDP, Chun-Mei, 2011, [13]).

Moreover as the problem instances are revealed incrementally our focus is on Online DPDPTW (Jaillet and Wagner, 2008 [35]). To the best of our knowledge no research work addresses the exact problem we focused on, i.e. Online Stochastic and Dynamic Pickup and Delivery Problem with Time Windows (OSDPDPTW). More detailed specification can be given regarding the fleet dimensions such as the number of the fleet members, i.e. single-vehicle DPDPs (Swihart and Papastavrou, 1999 [55], and Gribkovskaia, and Laporte in 2008 [26]) and multi-vehicle DPDPs, investigated by Jaillet et al. in [34], 2004 and Dessouky, and Lu the same year [19]. Besides the number of vehicles, their type as well as their home locations, so-called depots, also have attracted significant research interest. In [33], 2000

Irnich studied the multi-depot PDP with a single hub and heterogenous fleet. He focused on problems where all possible routes can easily be enumerated, i.e. the problem primarily considers the assignment of transportation requests to routes. The hub serves as a consolidation point which often assumes short routes between it and the locations in the transportation network, i.e. involve only one or very few customers. The rationale for this one is in the narrow time windows as well as in the high quantities, which make it possible to fully load a vehicle at one customer. Recall, the Bonds Express Couriers has at its disposal a heterogenous fleet, whose members have their own depot. Consequently, the variant we consider is a multi-depot OSDPDPTW. In addition to the quantitave and type description of the fleet a significant attention is devoted to the commodity dimensions of the fleet members. For example, Hernández-Pérez and Salazar-González in 2005 ([28]) studied the multi-commodity version of PDTSP, while in 2011 ([52]) Psaraftis explored exactly dynamic programming solutions for the multi-commodity PDP when one or two vehicles are available. In this thesis we assume a fleet, which has several commodity dimensions. In our case these are the number of pallets and the weight a particular order is composed of.

We are interested specifically in learning dispatch policies to control the fleet of vehicles online, rather than using an offline approach. We are not aware of much research in this precise setting. Inspired by this and the real case-study arisen in the courier company we designed a recommendation system, which applies dispatch rules whenever a new errand arrives taking into account the current parameters of the overall fleet. The decisions made by the system also take into account the possibility of unreserved demands to occur. There are several attempts to dispatch the fleet of vehicles, which we report next. For instance, Cortés et al. (2008, [15]) applied a hybrid-predictive control for fixed-fleet size DPDPs including traffic congestion, incorporating future information regarding unknown demands and expected traffic conditions. Also Gendreau et al. (2006, [24]) proposed neighborhood search heuristics to optimize the planned routes of vehicles in a context where new requests, with a pick-up and a delivery location, occur in real-time. Their study is based on ejection chains technique and, furthermore, they investigate the impact of a master-slave parallelization scheme on the optimization process. Two years later, in 2009, Beham et al. ([6]) considered agent-based simulation of dispatching rules in DPDP. This work treats the topic of solving dial-a-ride problems. A simulation model is introduced that describes how an agent is able to satisfy the transportation requests using a complex dispatching rule, which is optimized by metaheuristic approaches. The authors are using fitness function in order to evaluate the quality of the agent state.

1.3 Specification of VRPs

In this section we present a specification of the VRP subclass of optimization problems we are dealing with in this thesis. The graph of Figure 1.2 with top node "Vehicle Routing Problems" follows the historic introduction of the previous section and provides a clear view on how a given general VRP instance could be specified. We do not argue that our presentation is unique as there are many other possible classificators that take into account other problem features. For instance, a vehicle diversion is one of them: a particular vehicle can decide to visit a new, previously-unknown location on its way towards some previously-

known customer. Another issue we omitted is related with the traffic congestion. On could argue that the traffic is important feature in areas such as courier, cap companies, etc. and a problem instance could be classified with respect to the intensity of the traffic conditions. To emphasize the instance we deal with, we draw a blue line starting from the root, passing through the nodes matching our assumptions, and it ends into several nodes describing some of the general fleet, customer and dynamic features taken into account during modelling the real case-study.

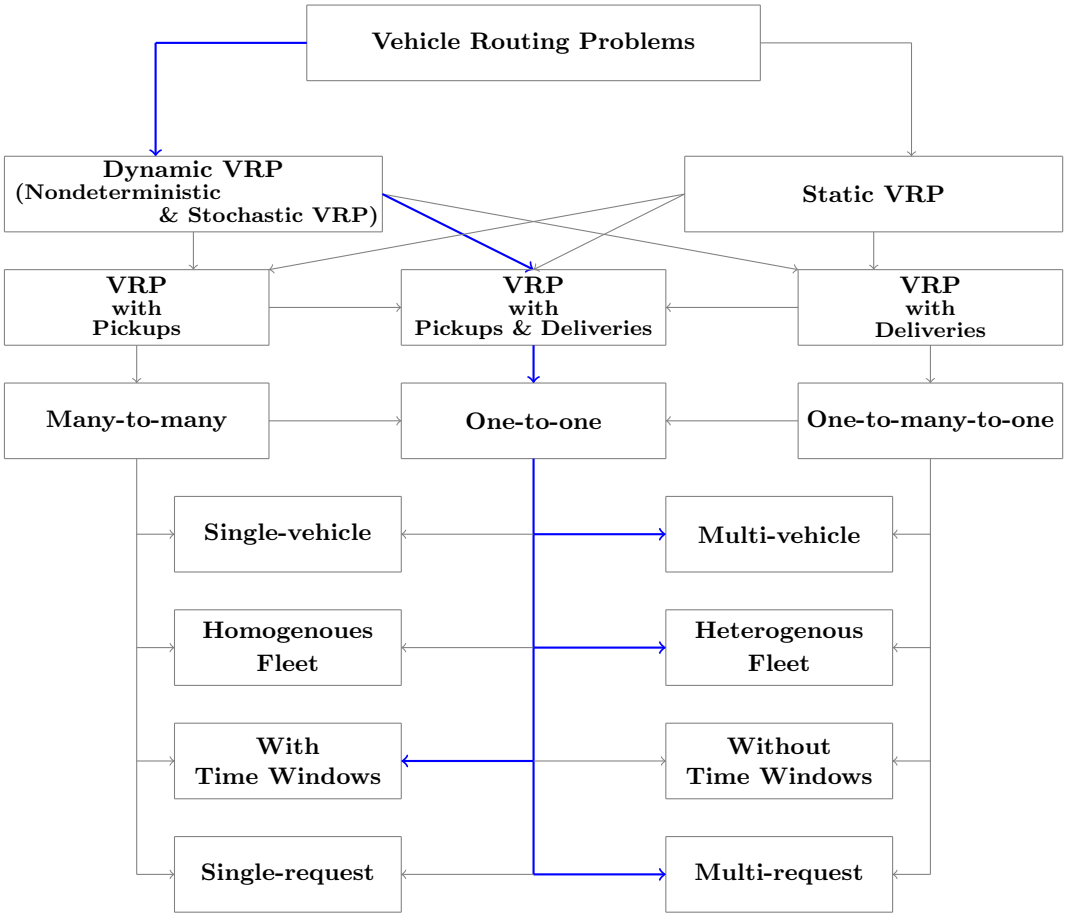


Figure 1.2: A specification of Vehicle Routing Problems.

1.4 The Vehicle Routing Solver Indigo

Here, we give a brief description of the VRP solver Indigo version 2.0 used along this work. In order to obtain meaningful results allowing us to rate the implemented dispatch rules a large number of suboptimal solutions are needed. Each solution is composed of vehicle assignments as well as the precise timing of their executions. For this purpose, we used Indigo as an oracle, able to produce such timetables when given a particular specifiction of

a VRP instance. The process of creating such a schedule, which helps in managing the fleet of vehicles, is divided into two main phases. During the first, the system constructs visiting assignments to each vehicle subject to the following requirements :

1. It returns ordered routes.
2. The objective could be specified in terms of distance, time or cost measures.
3. It allows arbitrary customer requests, i.e. pickups and deliveries.
4. A single time window for each customer location.
5. Vehicle limits, i.e. maximal capacities.
6. Vehicle availability window, i.e. the time when a particular vehicle is available. The start and end locations are arbitrary.
7. Compatibility constraints.
8. Metrics (i.e. distance, time, cost) represented as a matrix of values between any two locations.
9. Only one route per vehicle is allowed.

The usual way to construct a solution to routing problems is via insertion. That is, the representation of the emerging routes is kept internally. The system first chooses a visit to insert, then looks at all possible insertion points and when the best such point is selected, it updates the routes, and continues by considering the next visit. The insertion methods implemented are basically weighted combinations of visit characteristics such as :

1. The number of routes where the visit can be feasibly inserted into.
2. The width of the time window(s).
3. The size of the load(s).
4. The minimum insertion cost.
5. The insertion regret cost (difference between best and second-best cost).
6. The amount the insertion reduces the slack (spare time) in the best route.

Once the initial routes are built, they are subsequently improved by means of local search during the second phase of constructing the final schedules. The VRP Solver searches for a neighbour solution (defined according to one of a number of local search neighbourhoods) that is both feasible according to the basic constraints, and cost-reducing. When one is found, Indigo calculates the true cost and feasibility of a possible implementation using invariants. This is the cost which would then be used to determine whether the change is accepted. The following local search neighbourhoods are realized in Indigo :

1. 2-opt assumes moving a pair of consecutive visits to another route or to the same one, but at a different part, in both forward and reverse orientations.

2. Or-opt considers moving a sequence of k consecutive visits to another route or another part of the same route, in both forward and reverse orientations. The number k usually is of an order between 5 and 10.

3. Large Neighbourhood Search partially deconstructs (removes visits from the solution) and then re-constructs the solution. To do the last it uses the construction methods listed above. Thus, in effect a call to the VRP Solver with a partially-constructed solution is made, but acceptance of the resulting solution depends on the meta-heuristic being used. The designed meta-heuristics for that purpose are Hill-climbing - best first, Hill-climbing - first-found, Adaptive tabu search, Limited Discrepancy Search and Simulated Annealing.

The quality of the final schedules in terms of the specified objective depends partly on the number of the improvement iterations performed during the second implementation phase. In our work, all the solutions produced are improved under the same parameter settings, which allow us to use them as an uniform baseline during the experiments. The parameters we used to build solutions are Large Neightbourhood Search ([16]) combined with the Simulated Annealing ([21]) meta-heuristic, for a total number of 10000 improvement iterations.

1.5 Thesis contributions

Here we summary the contributions we made in the field of Vehicle Routing Problems. An Australian courier company has approached NICTA to provide a decision-support tool for managing their deliveries. Motivated by this problem we concentrated on the development of a recommendation module for this tool. We first introduced a theoretical framework within which we conducted our research. Although, there are many such frameworks in the literature, we adopted one fitting best to our needs. We modelled the real case-study as an Online Stochastic and Dynamic Pickup and Delivery Problem. The latter VRP variant has been investigated thoroughly within the last years, however, not much research attention considers the exact assumptions we made.

In order to support the company workforce in dispatching the fleet we implemented eight online dispatch heuristics, which take into account the current status of the fleet together with the new demands. This set is composed of the rules **Minimum Cost**, **Balanced**, **Current Orders**, **Shift Profitability**, **Geographical Closeness**, **Immediate Cost**, **Minimize Vehicles** and **Random**. For each rule we presented its algorithm as well as discussed its correctness and complexity.

Next, we generated 330 days of benchmark data, which were used during the experiments. The datasets were generated with a particular attention on the problem in hand (a version of Pickup and Delivery Problem). We continued with the evaluation of the dispatch rules over a hundred of these artificially generated benchmarks. For this purpose we introduced several valuation measures, i.e. *precise* and *type* global and local measures. The entire procedure was performed under three pooling scenarios, i.e. (*once*, *time-zones* and *fixed-time-span*) for both cases of presence and absence of seeming client demands. We compared

the results and reached the conclusion that the best profit gained with respect to the offline solution was achieved when *time-zones* strategy was applied. We also reported various descriptive statistics for each of the scenarios.

In the last part of our work we presented the recommendation module. Under its scheme we implemented four ad-hoc policies and we discussed their performance. These are **MinSumPG**, **MaxSumPG**, **MinSumPLG** and **MaxSumPLG** policies some of which are based only on global features and others take into account also local properties. We compared the results produced with the offline schedules for thirty days of client requests. The best performance in terms of final cost in *once* pooling strategy was achieved by **MinSumPLG**, however, the best management in terms of used vehicles and cost was realized by **MaxSumPG** policy. In *time-zones* and *fixed-time-span* all the policies performed good. However, using more vehicles is crucial in the final cost minimization (**MinSumPG** and **MinSumPLG**). On the other hand, if one is interested in using less human resources, then pursuing **MaxSumPG** or **MaxSumPLG** can be beneficial as they actuate relatively smaller number of vehicles and at the same time achieve good final cost values. Our online system achieved costs $34 - 43\%$ above than those obtained using Indigo in the *once*-setting and in the *time-zones* it even outperformed the solver for some of the time slices and policies.

1.6 Thesis Structure

The thesis continues with a description of the framework we work within in Chapter 2. Chapter 3 presents the dispatch rules we implemented. Then we describe the first phase of our experiments in Chapter 4. During this phase we build the learning datasets used during the second experimental phase in Chapter 5. Finally, Chapter 6 concludes the thesis with a summary of our contributions to the field of routing problems and a discussion on some promising future perspectives.

Chapter 2

A Framework for DPDP

Being predictive optimizes the costs.
by Martin Aleksandrov

This chapter begins with the description of the framework we used to model the addressed problem. It introduces several appropriate notions and notations (i.e. *time windows*, *pickup and delivery route*, etc.), which are relevant to our work and highlights several features of the problem in hand.

Subsequently we formulate the basic model of a VRP in terms of a linear integer program. We present the collection of constraints used and explain their semantics. Next, we introduce the additional requirements imposed by the fact that each call in the company contains information about two connected demands, i.e. a pick-up and a delivery requests. Precedence limitations related to these services are also considered and presented together with client preferences in terms of time windows for both subdemands.

The chapter continues with a discussion on two important questions arising in the dynamic PDPs. The first one is related to the possibility of weakening the dynamic burder implied by the multiple daily arrivals and the second one affects the subject of predicting those demands. We discuss relevant strategies such as splitting the working horizon into *time slices* and using a stochastic model in order to anticipate future demands.

The objective function is presented in detail in the next section together with its components and a description of the terms within its body. We conclude with a small section devoted to the computational complexity of the arised multi-vehicle routing problem.

2.1 The Problem Formulation

Fleet In our model, we have $m < \infty$ vehicles with different capacity characteristics. Let we denote the set of them with $V = \{v_1, \ldots, v_m\}$. For each vehicle v_j we denote the maximum number of pallets and weight capacity with Q_j and U_j, respectively. These values should never be exceeded. Each day vehicle v_j starts and ends its shift at so-called depot d_j with no cargo loaded. We denote as $D \equiv \cup_{j=1}^{m} d_j$ the set of all depot locations. Lastly, each vehicle v_j performs with different average velocity on the road.

Orders *An order* is a demand requested to the central dispatcher unit either through a phone or an internet inquiry. Let the set of all transportation orders be $O = \{o_1, \ldots, o_n\}$. Each order $o_i \in O$ is composed of quantities q_i and u_i (number of pallets and weight, respectively). These are to be transported from an origin l_i^1 to a destination l_i^2, satisfying the customers time preferences at these locations.

Requests *A request* is a subdemand, which contains information only about the pick-up or the delivery customer. Thus, each order $o_i \in O$ corresponds to two requests. By $R_O = \{r_1^1, r_1^2 \ldots, r_n^1, r_n^2\}$ we denote the set of all transportation requests, where $o_i = (r_i^1, r_i^2)$, r_i^1 and r_i^2 being the i-th order with its corresponding pick-up and delivery requests. The commodity values are positive for a pick-up and negative for a delivery request.

Urban network Let $L^1 \equiv \cup_{i=1}^{n} l_i^1$ and $L^2 \equiv \cup_{i=1}^{n} l_i^2$ be the sets of all pick-up and delivery locations, where n is the number of orders. Furthermore, let $L \equiv L^1 \cup L^2$ be the set of all the customer locations and $\mid L \mid \leq 2 * n$ be its upper bound. Thus, we have at most $2 * n + m$ (customer plus depots) distinct locations and for every pair $l_i, l_j \in L \cup D$, let $d_{i,j}$ denote the distance between l_i and l_j, and $t_{i,j}^k$ the travel time between them by a vehicle $v_k \in V$.

Time windows Each request $r_i \in R_O$ has an associated *time window*, i.e. the time interval, in which service at the particular location must take place. For $o_i = (r_i^1, r_i^2) \in O$, the time window for r_i^1 is denoted by $[t_e^{i,1}, t_l^{i,1}]$ and for r_i^2 by $[t_e^{i,2}, t_l^{i,2}]$. The *release* time is the earliest time a request and *deadline* its latest time.

Service time Each visit to a particular location requires time for executing the services related with it, such as loading or unloading. We call this period *the service time* associated with the considered request. This time usually depends on the vehicle performing the service and it may differ for the pick-up and the delivery services and customers, but in this thesis we assume them to be equal at both the order locations and vehicle-independent. Therefore, if $o_i = (r_i^1, r_i^2)$ is an order and each request location requires s_i time units, then the service time for the whole order is $2 * s_i$. In addition, the depots are the only locations, to which no service times are associated.

A *customer request* $r_i^k \in R_O$ is thus represented by the following tuple :

$$r_i^k = < t_i, (-1)^{k+1} * q_i, (-1)^{k+1} * u_i, [t_e^{i,k}, t_l^{i,k}], l_i^k >,$$

where (1) t_i is the arrival time of r_i^k, (2) the second and the third components are the commodities q_i and u_i, however, they are multiplied by an addition term, which is 1 if the request is a pick-up and -1, otherwise, (3) the request time window and (4) the location where the request has been required.

The first definition below relates the vehicles in the fleet V with a set of requests, while the second generalizes that relationship for the entire fleet. Both definitions impose constraints on the vehicle management.

Definition 1 : (**Vehicle Route**) Let $V = \{v_1, \ldots, v_m\}$ be a vehicle fleet. Then for each $v_j \in V$ a *pick-up and delivery route* $R_j = \{r_{j_1}, \ldots, r_{j_{n_j}}\}$ (PDR) is an ordered set of visits through a subset of R_O such that :

- $r_{j_1}, r_{j_{n_j}}$ are associated with $d_j \in D$ and no r_{j_m} with $m \in \{2, n_j - 1\}$ does.
- For a given order $o_i \in O$ both or neither r_i^1 and r_i^2 belong to R_j. If both r_i^1 and r_i^2 belong to R_j, then r_i^1 is serviced before r_i^2.
- The vehicle v_j services each request in R_j exactly once.
- The total load of all pickups in R_j does not exceed the maximal commodity values Q_j and U_j, at any location.
- For each request $r_{j_m} \in R_j$, the time window is feasible, i.e. $t_e^{j_m} \leq t_l^{j_m}$.

Note that the first and the last visit in a PDR (the depot) are virtual requests and can be represented, for each $d_j \in D$, with the tuple $< 0:00, 0, 0, [c_j\ , f_j], d_j >$, where (1) $[c_j, f_j]$ is the depot shift-time window, which opens at c_j and closes at f_j, and (2) d_j is the $j-$th depot location. We denote the set of these requests as R_D and we call each element from it *a home request*.

Definition 2 : (**Routing Plan**) Let $V = \{v_1, \ldots, v_m\}$ be the considered fleet. A *pickup and delivery routing plan* (PDRP) for managing the fleet V is a set of routes $\mathcal{R} = \{R_j \mid v_j \in V\}$ such that :

- The route R_j is a pickup and delivery route for each vehicle $v_j \in V$.
- The set $\{R_j \mid v_j \in V\}$ is a partition of R_O.

We thus defined the vehicle routes as disjoint sets of requests, whose union results in the entire set R_O. Each of them contains information about the total load a vehicle has to carry, but they do not address timing at the locations, possible waiting times for early arrivals or delays for late arrivals. These are captured by the following definitions.

Definition 3 : (**Scheduling Plan**) Let $V = \{v_1, \ldots, v_m\}$ be the fleet of vehicles. For a set of PDRs $\mathcal{R} = \{R_j \mid v_j \in V\}$, $\mathcal{I} = \{I_j \mid R_j \in \mathcal{R}\}$ is a *pickup and delivery scheduling plan* (PDSP) where:

- For each $R_j = \{r_{j_1}, \ldots, r_{j_{n_j}}\}$, there is an associated itinerary $I_j = \{i_{j_1}, \ldots, i_{j_{n_j}}\}$. Each element i_{j_k} is called *the timetable* associated with the request r_{j_k} or *request itinerary* is defined as,

$$i_{j_k} =< a^j_{j_k}, w^j_{j_k}, b^j_{j_k}, s_{j_k}, e^j_{j_k} >,$$

  where $a^j_{j_k}$, $w^j_{j_k}$, $b^j_{j_k}$, s_{j_k} and $e^j_{j_k}$ are the arrival, waiting, starting, service and departure times for the given request.

Similarly, to the home requests we define a timetable at each depot location $d_j \in D$. That is, a tuple of the form : $< c_j,\ 0,\ c_j,\ 0,\ p_j >$, where c_j is the time a vehicle is available at d_j and p_j is the time when it leaves that location. We call such a tuple *home itinerary* and the set of those we denote as I_D.

Definition 4 : (**Routing and Scheduling Plan**) A *pickup and delivery routing and scheduling plan* (PDRSP) is a pair $P = (\mathcal{R}, \mathcal{I})$, where $\mathcal{R}$ is a routing plan and $\mathcal{I}$ is a scheduling plan for it.

From now on we assume that whenever we discuss a PDRSP we know its underlying set of orders. When we work with multiple sets of demands we will explicitly refer to one if needed. Our final goal is to construct such a routing and scheduling plan achieving convenient total costs and managing the fleet in a reasonable way during that day. The next measures address the quality of a fleet schedule. We do not argue that these are all the measures for evaluting a given plan. Rather than, they were reasonably chosen representing the company interests. From now on in order to avoid a possible confusion with the notations every time we need a component of a particular request r^k_j or an itinerary i^k_j we will refer to it directly.

Definition 5 : (**Vehicle performance**) Let $P = (\mathcal{R}, \mathcal{I})$ be a PDRSP and $R_j = \{r_{j_1}, \ldots, r_{j_{n_j}}\}$ be a PDR executed by a vehicle $v_j \in V$ following timetable $I_j = \{i_{j_1}, \ldots, i_{j_{n_j}}\}$.

- $Duration(j, P) = e^j_{j_{n_j-1}} - a^j_{j_2}$ is the duration of the route R_j
- $FreeP(j, l, P) = (Q_j - \sum_{k=1}^{l} q_{j_k})$ for each $l \in \{1, \ldots, n_j\}$ is the free capacity in v_j with respect to the number of pallets at visit r_{j_l}
- $FreeW(j, l, P) = (U_j - \sum_{k=1}^{l} u_{j_k})$ for each $l \in \{1, \ldots, n_j\}$ is the free capacity in v_j with respect to the weight at visit r_{j_l}
- $ServiceTime(j, P) = \sum_{k=1}^{n_j} s_{j_k}$ is the *serving time* of route R_j

- $WaitingTime(j,P) = \sum_{k=1}^{n_j} w_{j_k}^j$ is the *waiting time* of route R_j
- $Lateness(j,P) = \sum_{k=1}^{n_j} l_{j_k}^j$ is the *total lateness* of route R_j, where $l_{j_k}^j = b_{j_k}^j - t_l^k$, if $b_{j_k}^j > t_l^k$ or 0, otherwise, is the lateness at visit r_{j_k} by vehicle v_j
- $TravelTime(j,P) = \sum_{k=1}^{n_j-1} t_{k,k+1}^j$ is the *total travel time* of route R_j
- $ExecutionTime(j,P) = TravelTime(j,P) + WaitingTime(j,P) + ServiceTime(j,P)$ is the *total execution time* of route R_j
- $Unsat(j,P) = \sum_{k=1}^{n_j} penalty(r_{j_k})$ is the number of *unsatisfied* customers along R_j, where $penalty(r_{j_k}) = 1$, if $l_{j_k}^j > 0$, and 0, otherwise

The best profit of R_j in terms of serviced clients and execution time could be achieved if we maximize n_j and minimize $ExecutionTime(j,P)$, and $Lateness(j,P)$ possibly satisfying all hard and soft constraints. As for each request $r_{j_k} \in R_j$, s_{j_k} is fixed at the location l_k, then $ServiceTime(j,P)$ will be fixed for n_j requests. Moreover, under the assumption that the fleet of vehicles are moving with constant velocity on the road, we have that $TravelTime(j,P)$ is also fixed for a given route. Hence, for a fixed number of customers along R_j, we can minimize $ExecutionTime(j,P)$ only if we minimize $WaitingTime(j,P)$. In addition, minimizing $Lateness(j,P)$ will maximize the customer satisfaction, i.e. maximize the number of on-time deliveries. Despite the vehicle measures above, one is more interested in overall fleet performance or in the quality of work a particular vehicle type produces throughout the day. The former gives a possibility to evaluate the entire fleet schedule, while the latter observes more the road behaviour of a particular subset of the fleet.

Definition 6 : (**Fleet performance**) Let $V = \{v_1, \ldots, v_m\}$ be the fleet of vehicles, $O = \{o_1, \ldots, o_n\}$ be a set of transportantion orders, and $P = (\mathcal{R}, \mathcal{I})$ a PDRSP for managing O by V. Then we define the following features related with this plan :

- $ServiceTime(P) = \sum_{j=1}^{m} Service(j,P)$ is the *service time* of plan P
- $WaitingTime(P) = \sum_{j=1}^{m} Waiting(j,P)$ is the *waiting time* of plan P
- $Lateness(P) = \sum_{j=1}^{m} Lateness(j,P)$ is the *lateness* of plan P

- $TravelTime(P) = \sum_{j=1}^{m} Travel(j, P)$ is the *travel time* of plan P, i.e. the total travel time of the fleet
- $ExecutionTime(P) = TravelTime(P) + WaitingTime(P) + ServiceTime(P)$ is the *total execution time* of plan P
- $Unsat(P) = \sum_{j=1}^{m} Unsat(j, P)$ is the total number of *unsatisfied* customers in P
- $Active(P) = \{R_j \mid Execution(j, P) \neq 0\}$ is the set of active vehicles in P

Definition 7 : (**Type performance**) Let $O = \{o_1, \ldots, o_n\}$ be a set of transportantion orders and $V = \{v_1, \ldots, v_m\}$ the fleet of vehicles. If T is a vehicle type, then by $V^T = \{v_j \mid v_j \in V \text{ and it is of type } T\}$ we denote the set of vehicles of type T. Furthermore, let $P = (\mathcal{R}, \mathcal{I})$ be a PDRSP. Then the pair $P^T = (\mathcal{R}^T, \mathcal{I}^T)$ represents a PDRSP for V^T, where the sets $\mathcal{R}^T = \{R_j \mid R_j \in \mathcal{R} \text{ and } v_j \in V \text{ is of type } T\}$ and $\mathcal{I}^T = \{I_j \mid v_j \text{ is of type } T\}$ are, respectively, the routes performed only by vehicles of that type and their itineraries.

- $n^T = \sum_{j=1}^{|\mathcal{R}^T|} (n_j - 2)$ is the total number of serviced locations by the vehicles of that type where n_j is the number of visits in $R_j \in \mathcal{R}^T$
- $Active_{\mathcal{R}}^T = \sum_{j=1}^{|\mathcal{R}^T|} (Service(j, P^T) + Travel(j, P^T))$ is the time of active performance of the vehicles from V^T
- $n_j^T = \frac{n^T}{2 * |Active(P^T)|}$ is the average number of serviced orders per driver of a vehicle from V^T
- $AverageDur(P^T) = \frac{\sum_{j=1}^{|\mathcal{R}^T|} Duration(j, P^T)}{|\mathcal{R}^T|}$ is the average schedule duration per vehicle from V^T.

2.2 The Basic VRP

In this section we formulate the core of our problem as a linear integer program. First we define three integer variables, two of which serve as order and vehicle indicators and the third represents the current vehicle freight. In the next, we assume that the sets of indices for the depot and customer locations and disjoint. For each $v_j \in V$ and $r_i \in R_O$, let $x_{ij} = 1$ if and only if the request r_i is assigned to vehicle v_j. Secondly, the variable $y_{i_1 i_2 j} = 1$ if and only if the vehicle v_j travels between the customers requested r_{i_1} and r_{i_2}, both belonging

to R_O. Lastly, the variable z_{ij}^c stores the cummulative load for the commodity $c \in \{q, u\}$ carried by vehicle v_j, visiting location l_i of request $r_i \in R_O$. It accepts only natural values as we consider a natural number assigned to each commodity. In each time instant this value should be less or equal than the maximum allowed for the particular commodity in vehicle v_j. As the only locations where a vehicle could change its capacities are the customer ones, which belong to its route, the value of z_{ij}^c will increase after visiting a pick-up location and it will decrease when service at a delivery location is performed. Next, we define the VRP problem in terms of integer constraints using the variables we defined as follows:

1. $\forall r_i \in R_O. \sum_{j=1}^{m} x_{ij} = 1$
2. $\forall r_i \in R_O. \sum_{j=1}^{m} \sum_{r_k \in R_O} y_{ikj} = 1$
3. $\forall v_j \in V$ with home request $r_j \in R_D. \sum_{r_i \in R_O} y_{jij} = 1$
4. $\forall v_j \in V$ with home request $r_j \in R_D. \sum_{r_i \in R_O} y_{ijj} = 1$
5. $\forall r_i \in R_O. \forall v_j \in V. \sum_{r_{k_1} \in R_O} y_{k_1ij} - \sum_{r_{k_2} \in R_O} y_{ik_2j} = 0$
6. $\forall r_i \in R_O. \forall v_j \in V. \sum_{r_k \in R_O} y_{ikj} * Q_j \geq z_{ij}^q + q_i$ and $\sum_{r_k \in R_O} y_{ikj} * U_j \geq z_{ij}^u + u_i$
7. $\forall r_{i_1}, r_{i_2} \in R_O \cup R_D. \forall v_j \in V. y_{i_1i_2j} = 1 \Rightarrow z_{i_1j}^q + q_{i_2} = z_{i_2j}^q$ and $z_{i_1j}^u + u_{i_2} = z_{i_2j}^u$
8. $\forall r_i \in R_O. \forall v_j \in V$ with home request $r_j \in R_D. y_{jij} = 1 \Rightarrow z_{ij}^q = 0$ and $z_{ij}^u = 0$
9. $\forall r_i \in R_O. \forall v_j \in V$ with home request $r_j \in R_D. y_{ijj} = 1 \Rightarrow z_{jj}^q = 0$ and $z_{jj}^u = 0$
10. $\forall r_i \in R_O. \forall v_j \in V. z_{ij}^q \geq 0$ and $z_{ij}^u \geq 0$

Table 2.1: VRP constraints.

The first constraint imposes that each transportation request is assigned to exactly one vehicle. Recall, that we do not consider a possibility of transferring packages between different vehicles at any location. Hence, an order which has been picked up by one driver has to be delivered by the same vehicle. The second constraint expresses that each request in R_O is serviced exactly once by exactly one vehicle. The next limitation imposes that each vehicle departs from its home location (services its home request) towards exactly one customer location. Similarly, the vehicle arrives at its home from exactly one customer location. The following requirement, sometimes called *equilibrium* condition, imposes that whenever a vehicle arrives at a customer location to perform services, it also must depart from it. Constraint 6 assures that when visiting a new customer, the corresponding vehicle has enough capacity to accommodate her needs. The next constraint tells us how to calculate the current vehicle load when travelling from one location to another. Also it says that on its way between any two consecutive visits, a vehicle does not change its capacities except at the customer locations associated with these visits. The reason for that is because we do not allow a cross-utilization amongst fleet members and we do not have store locations anywhere on the map. Recall, that the amount of load has positive and negative values for

the pick-up and the delivery locations, respectively. Next, constraints 8 and 9 impose that each vehicle depart from and arrive at its depot empty and the last constraint keeps the values of the current cummulative commodities non-negative.

2.3 Paired Customers and Time Windows

Following the basic constraints in the previous section, we introduce constraints to impose that each order is paired (comprised of pick-up and delivery requests). Hence, our locations are connected and under our assumptions such places should be visited by the same vehicle. A natural precedence constraint is presented and discussed. In addition, each request has its own time preferences, i.e. time window within which the service at a particular location should take place. The earliest such time is a hard constraint as we assume that the package is not ready before that time. Consequently, if a vehicle arrives earlier than this time, it should wait in order to start its services. The latest time for a given location is a soft constraint and it could be violated. We pay a penalty for such a delay, which is taken into account when we consider the objective function.

Next, we proceed more formally. Let $o_i = (r_i^1, r_i^2) \in O$ be an order. Let r_i^1 be at location l_i^1 and r_i^2 at location l_i^2. In addition, let $[t_e^{i,1}, t_l^{i,1}]$ and $[t_e^{i,2}, t_l^{i,2}]$ be the time windows related to these locations. Let the demand o_i to be assigned to vehicle $v_j \in V$ and let $I_j \in \mathcal{I}$ be its itinerary in plan $P = (\mathcal{R}, \mathcal{I})$. Futhermore, let p_j be the departure time for that vehicle. Thus, in addition to constraints 1. - 10., the following constraints should also be satisfied.

11. $\forall o_i \in O \forall v_j \in V.\ x_{i^1 j} = 1 \Leftrightarrow x_{i^2 j} = 1$
12. $\forall o_i \in O \forall v_j \in V.\ x_{i^1 j} + x_{i^2 j} = 2 \Rightarrow d^j_{j_{i^1}} \leq d^j_{j_{i^2}}$
13. $\forall o_i \in O.\ t_e^{i,1} \leq t_l^{i,1}$ and $t_e^{i,2} \leq t_l^{i,2}$
14. $\forall r_i, r_k \in R_O.\ \forall v_j \in V.\ y_{ikj} = 1 \Rightarrow d^j_{j_i} + t^j_{i,k} \leq d^j_{j_k}$
15. $\forall r_i, r_k \in R_O.\ \forall v_j \in V.\ y_{ikj} = 1 \Rightarrow d^j_{j_i} + t^j_{i,k} = a^j_{j_k}$
16. $\forall r_i, r_k \in R_O.\ \forall v_j \in V.\ y_{ikj} = 1 \Rightarrow a^j_{j_i} + w^j_{j_i} + s^j_{j_i} = d^j_{j_i}$
17. $\forall r_j \in R_D.\ d^j_1 \geq p_j$

Table 2.2: Paired and time window constraints.

The initial constraint expresses the fact that r_i^1 is assigned to vehicle v_j if and only if r_i^2 is also assigned to v_j. The second constraint relates both the pick-up and the delivery locations of order o_i. It simply imposes that l_i^1 must be visited before l_i^2. That is, if o_i is assigned to v_j, then the departure time at l_i^1 must be less than the one at the corresponding delivery location l_i^2. The next limitation imposes the feasibility of the pick-up and the delivery time windows. In general the release times at two paired locations could differ, but we assume that the earliest delivery time is equal to the earliest pick-up time. The next condition relates any two consecutive departure times and enforces that a vehicle departs from its next location after it has departed from its current location plus the additional travel time between these locations. Constraint 15 expresses that once a vehicle departs

from a location, it arrives at the next one after the required time units. We assume that the vehicle driver does not wait somewhere on his way between the two locations. The only place he is allowed to wait is at a customer location whenever he arrives earlier than the earliest time for that location. The length of the waiting time window associated with a request $r_i \in R_O$ is $w_i = t_e^i - a_i$ if $a_i < t_e^i$ and 0, otherwise. We assume that in the latter case, the driver starts serving the client immediately after he arrives, as expressed by constraint 16. In other words, the only time a driver spends at a customer location is equal to the service time needed for the corresponding demand. Within this time the driver could consult a dispatcher about his next destination and then to head towards it. The final condition simply says that each vehicle departs from its home location at the departure time associated with that depot.

Costraints 1-17 represent a linear integer program corresponding to the static version of our problem. This program models our problem as a PDPTW. However, the addressed problem is dynamic as the client orders arrive unpredictable. Thus, hereafter, we discuss the issues related with the dynamic PDPTW version.

2.4 Switching to a Dynamic Setting

In this section we discuss the issues related to the problem dynamics. First, we deal with the so called *one-to-one* PDPTW. In other words we have a fleet of vehicles that leave empty their home locations, serve a particular number of demands and travel empty back to their depots at the end of their job. Each order has a given commodity, which is to be serviced between an origin and a destination. As the time goes by along a working day, the amount of known information revealed to the decision maker is getting larger. This knowledge is updated each time a set of new orders arrives, at which moment each vehicle is either moving towards a customer, serving a customer or waiting at a location in order to start service. In a real-time setting the system should make a decision for each vehicle in the plan, to wait or to go, and for the new order, to accept or to reject it. Following the company policy, the system must accept each order, even though it cannot be immediately assigned to a vehicle. A decision to assign the order to a given vehicle is made, whenever the system evaluates where it can be included. Such an update of the current plan should be based on features of the current pickup and delivery routing and scheduling plan as well as on the characteristics of the new obligations. In the literature, many approaches have investigated different short-term and long-term features such as the current makespan, i.e. the time the last client is serviced, total latency, i.e. the sum of the completion times for all routes in the current plan, total distance traveled by the vehicles, degree of dynamism, number of the late orders, number of after shift vehicles and many others.

On the other hand, in the process of decision making, predictions can be made on the basis of historical data. A predicted model of the environment can be useful in several ways. Based on the arrival and commodity distributions we can try to predict with a given certainty where and when a new order will occur and also what will be its commodities. Using this information would help us to define georgraphical zone characteristics such as average total workload within a zone for a given time frame, types of vehicles which should

operate within a zone and an interval, waiting and buffering strategies related with the vehicle routes, etc. Thus, simulating artificial obligations could be beneficial for vehicle control and could contribute to achieve better objective values at the end of the day.

2.4.1 Time slices

During the daily time horizon the system could make a great number of small non-optimal decisions. Therefore, the implemeted final schedule may benefit if we reoptimize the current pickup an delivery routing and scheduling plan from time to time by taking into account the revealed new information. In this way the fleet could advance whenever a new decision needs to be made. We split the working time horizon based on the notion of *time slices*, initially proposed by Kilby et al. ([38], 1998). Although they consider only slices of equal length, we also investigate the case of different duration. For each time slice we solve a static PDPTW and use its suboptimal schedule as a reference in the evaluation phase of the conducted experiments. In total, we consider three horizon divisions. Firstly, we reroute and reschedule the current plan once at the beginning of the working day, i.e. we have only one time slice. In this case we know in advance only the orders stated on the previous days. Note that this strategy coincide with the one of not having slices at all and it has received significant research attention in the last years. The second division is based on the definition of time zones. The distribution of the new arrivals can be used for setting the boundaries of such time periods along the day. Then, the plan will be reoptimized before each time zone. This division is motivated by the past data statistics we studied. The third splitting is based on fixed-time-span intervals, i.e. intervals of equal duration. Lastly, we mention a possibility of rescheduling every time a new event happens. However, to reoptimize all vehicle routes whenever a client has been serviced or a new order has arrived is not realistic in problems with a high degree of dynamism, i.e. if the orders arrive within a time interval, significantly smaller than the time needed for doing any adequate global update (as in the case with companies such as Bonds Express). Table 2.3 contains the number n_{ts} of slices in each setting and the motivation behind them.

#	n_{ts}	**Motivation**	**Slice length**
1.	1	paper case	fixed
2.	5	arrival distribution	variable
3.	7	uniform working time	fixed

Table 2.3: Time slices scenarios.

2.4.2 Prediction model

In what follows, we discuss benefits of considering a prediction model. In the literature there are some attemps to exploit past data in order to anticipate the future demands ([32, 55]). Such a model aims helping to predict where and when possible demands will occur. For this purpose we split the time horizon in one-hour intervals. From one interval to the next the value of the process increases by the number of the orders arrived within this particular interval. Therefore, we can view this process as a stochastic counting process. In general,

there are two types of counting processes with respect to the arrival rate. One with a constant increments or *homogenous* and one that experiences different arrival numbers, i.e. *non-homogenous*. Moreover, a counting process can be classified with respect to the time horizon domain. If the domain is discrete, we call the process *discrete* and *countinuous*, otherwise. Each discrete counting process can be divided into several subgroups, according to the distribution of the interarrival time, i.e. the distribution of the time between each pair of consecutive process values. From this perspective, a process can be clasified as *periodic* or *aperiodic*. The latter type is specialized even further following the specific distribution mass function. For instance, a Poisson counting process has an exponential interarrival distribution (see [49]).

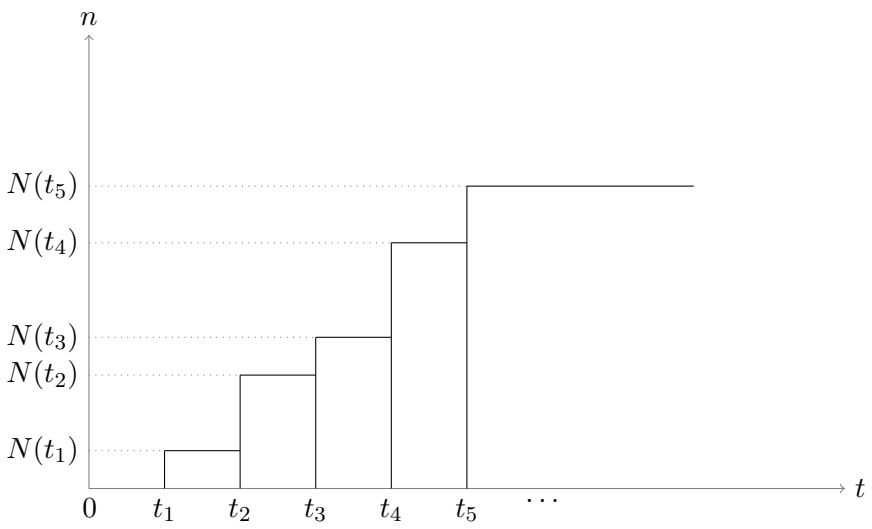


Figure 2.1: Non-homogenous and periodic counting process.

The distribution of the number of arrivals at the system can be approximated by a counting process. Figure 2.1 depicts a general graph of a non-homogenous, periodic counting process. We used an instance of it to collect various statistics for the different time intervals (see Section 4.2.5). For each of them we studied the distributions of the order features (see Section 2.1) based on past data and then used the statistics to generate one hundred datasets of orders. Their goal is to simulate fleet activity and in this manner to allocate the vehicles in positions, which are better for handling the occurrence of possible unreserved demands.

2.5 Objective Function

In this section we present the objective function of our problem, aiming at total minimizing the final total costs spent throughout a day for running the fleet. The costs are made up from the total vehicle running costs, driver wages, penalties for late services as well as extra costs for after-shift work. Since we assume heterogenous fleet of vehicles, the cost the company pays to have different vehicle types on the road is described in Table 1.1. In addition,

the driver wage has a constant value [1], which is based on hourly basis. We proceed more formally by presenting the request graph. A structure on top of which the desired objective is defined.

Definition 8 : (**Request graph**) Let $R_O \cup R_D$ be the request source of our problem. Then, a *request graph* G_O is a pair (N_O, E_O), where (1) $r_i \in N_O$ if and only if $r_i \in R_O \cup R_D$ and (2) $(r_i, r_j) \in E$ if and only if $r_i, r_j \in R_O \cup R_D$.

The request graph is just a network of all the order and home requests. Its nodes contain the information about the individual requests, while its edges relate them. Note that it is a fully connected, directed multi-graph and it imposes no restrictions either on the fleet or on their routes. Therefore, we relate the PDR of a given vehicle with a route in the request graph.

Definition 9 : (**Vehicle Graph Route**) Let $V = \{v_1, \ldots, v_m\}$ be the considered fleet, $G_O = (N_O, E_O)$ the request graph, and $R_j = \{r_{j_1}, \ldots, r_{j_{n_j}}\}$ the PDR for vehicle $v_j \in V$ according to the PDRSP P. A *vehicle graph route* $R_j^{G_O}$ corresponding to R_j is the ordered set $\{(r_{j_m}, r_{j_{m+1}}) \mid m \in [1, n_j - 1]\}$.

Note that for each vehicle $v_j \in V$ the vehicle graph route $R_j^{G_O}$ is a specific subgraph of the request graph G_O. Also as the vehicles service disjoint request subsets of R_O, it follows that the vehicle graph routes represent disjoint subgraphs in G_O. The latter does not prevent to have a common request location in more than one vehicle graph route and which is visited more than one time. Furthermore, note that for each vehicle graph route $R_j^{G_O}$ there is an itinerary $I_j \in \mathcal{I}$ associated with it.

In what follows we define the costs the company pays when a vehicle traversing its graph route. Let $V = \{v_1, \ldots, v_m\}$ be the fleet and for each vehicle v_j, let c'_j be the price to have v_j on the road and c''_j the driver's wage. Also recall that for any two locations $l_s, l_k \in L \cup D$ we denote by $d_{s,k}$ the distance and by $t^j_{s,k}$ the travel time between them.

Definition 10 : (**Edge cost**) Let $R_j^{G_O}$ be the vehicle graph route for the vehicle $v_j \in V$ and $I_j = \{i_{j_1}, \ldots, i_{j_{n_j}}\}$ be its itinerary. Then for each edge $(r_{j_s}, r_{j_{s+1}}) \in R_j^{G_O}$ the *cost* for traversing it by v_j is given with the expression :

$$c_{j_s j_{s+1}} = c'_j * d_{j_s, j_{s+1}} + c''_j * (t^j_{j_s, j_{s+1}} + w^j_{j_{s+1}} + s_{j_{s+1}}) + p^j_{j_{s+1}},$$

where (1) $w^j_{j_{s+1}}, s_{j_{s+1}} \in i_{j_{s+1}}$ are the waiting and service times at the location of $r_{j_{s+1}}$ and (2) $p^j_{j_{s+1}}$ is the penalty cost for the lateness at this location, if incurred. Note that a penalty at the last location means for some after-shift work. However, we do not differentiate it from any other penalty and we calculate all of them the same way. The latter can be done taking into account how urgent is the particular request. Nevertheless, we describe this concept in the next chapter when we discuss the cost-forming factors observed by a dispatching rule

[1] According to the NICTA internal report from September 6, 2011.

when evaluating where the new orders are to be inserted.

Definition 11 : (**Route cost**) Let $R_j^{G_O}$ be the vehicle graph route for the vehicle $v_j \in V$ and $I_j = \{i_{j_1}, \ldots, i_{j_{n_j}}\}$ be its itinerary. Then the cost v_j needs to execute the entire graph route $R_j^{G_O}$ is the following :

$$c_j = \sum_{(r_{j_s}, r_{j_{s+1}}) \in R_j^{G_O}} c_{j_s j_{s+1}}$$

Definition 12 : (**Schedule cost**) Given a plan $P = (\mathcal{R}, \mathcal{I}) \in \mathcal{P}$ from a set of plans for managing the fleet $V = \{v_1, \ldots, v_m\}$, the objective function $f : \mathcal{P} \to \mathbf{R}$, which takes a plan and returns a real number is given by the next expression where $e_j = (r_{j_s}, r_{j_{s+1}})$:

$$f(P) = \sum_{j=1}^{m} \sum_{e_j \in R_j^{G_O}} c_{j_s j_{s+1}} =$$

$$\sum_{j=1}^{m} \sum_{e_j \in R_j^{G_O}} (c'_j * d_{j_s j_{s+1}} + c''_j * (t^j_{j_s j_{s+1}} + w^j_{j_{s+1}} + s_{j_{s+1}}) + p^j_{j_{s+1}}) =$$

$$\sum_{j=1}^{m} \sum_{e_j \in R_j^{G_O}} c'_j * d_{j_s j_{s+1}} + \sum_{j=1}^{m} \sum_{e_j \in R_j^{G_O}} c''_j * (t^j_{j_s j_{s+1}} + w^j_{j_{s+1}} + s_{j_{s+1}}) + \sum_{j=1}^{m} \sum_{e_j \in R_j^{G_O}} p^j_{j_{s+1}}$$

The first term in the objective function $f(P)$ expresses the cost the company pays to run the entire fleet V according to P, the second represents the accummulated cost for the drivers' wages and the third accounts for the overall penalties for the late services. We may also express $f(P)$ by :

$$f(P) = \sum_{j=1}^{m} \sum_{e_j \in R_j^{G_O}} c'_j * d_{j_s j_{s+1}} + \sum_{j=1}^{m} \sum_{e_j \in R_j^{G_O}} c''_j * t^j_{j_s j_{s+1}} + \sum_{j=1}^{m} \sum_{e_j \in R_j^{G_O}} c''_j * w^j_{j_{s+1}} +$$
$$\sum_{j=1}^{m} \sum_{e_j \in R_j^{G_O}} c''_j * s_{j_{s+1}} + \sum_{j=1}^{m} \sum_{e_j \in R_j^{G_O}} p^j_{j_{s+1}},$$

where there are distinct terms for the total travel, total waiting and total service costs related with the plan P. All the cost components may serve as meaningfull indicators of the quality of a given plan. In general, one would be interested in minimizing the penalty term in the objective function and possibly to have control over the waiting term (it might be beneficial to wait some time at a customer location, but not too long). Finally, we present the objective function in terms of the integer variables we introduced previously. Recall, $x_{ij} = 1$ if and only if the i-th request from R_O is assigned to vehicle $v_j \in V$, and $y_{i_1 i_2 j} = 1$ if and only if vehicle $v_j \in V$ elapses the distance between locations l_{i_1} and l_{i_2} from $L \cup D$.

$$f(P) = \sum_{i=1}^{2*n} \sum_{j=1}^{m} x_{ij} \sum_{k=1}^{2*n} (y_{ikj} * c_{ik})$$

2.6 NP-hardness

Here, we briefly discuss the theoretical complexity of the problem in hand. The Pickup and Delivery Problem (PDP) is a kind of Vehicle Routing Problem (VRP), a variant of the Travelling Salesman Problem (TSP), which is well-known to be an NP-hard problem. These problems are particularly expensive as the exact algorithms solving them require, in general, exponential time. We do not show a detailed proof of NP-hardness for TSP (see [29, 36]), but we underline the major issues when is to be reduced to our problem.

In the classical TSP there is a set of cities, travel cost for each pair of them and a salesman. The goal is to construct such a visiting assingment of the salesman through all the cities such that the overall travel expense does not exceed a predetermined value. This problem is reducible to the task of finding a Hamiltonian cycle in the graph of locations. Therefore, as a single-vehicle VRP is reducible to the classical TSP, it means that a single-vehicle PDP does so as well. However, when the fleet includes many vehicles another problem emerges related to the vehicle assignments, i.e. how to partition the set of cities (clients) with respect to the number of "salesmen", so that for each vehicle and its respective set of cities a TSP algorithm is applied. Hence, multi-vehicle routing problem is harder that the single-vehicle one as two phases can be distinguished : (1) building vehicle assignments or partition the set of all customers and assign them to the vehicles and (2) for each vehicle apply TSP algorithm to optimize its route. We already discuss that problem 2 is NP-hard, however, this time we have multiple such problems. The problem 1 is also NP-hard because there is an exponential number of partitions for the set of all locations. Thus, we have an exponential number of possible vehicle assignments. Although, we do not give precise proof of the complexity, we believe that the problem in hand is at least as hard as a problem from NP^{TSP}, i.e. problems which are solvable by a non-deterministic Turing machine in a time proportional to the one needed for solving a TSP instance. Recall, the VRP solver Indigo 2.0. has a two-phased architecture. The first one is related to finding a reasonable vehicle assignment (problem 1 above) and during the second one the oracle tries to optimize all the routes for a certain number of local search iterations (problems of type 2).

Chapter 3

Simple Dispatch Heuristics

Making a decision can be vital.
by Martin Aleksandrov

This chapter addresses the heuristics we implemented and tested within our framework. It starts with a short introduction of the general circumstances under which the dispatchers have to make a decision about acceptance or rejection of a new arrival. It might be difficult for them to be aware of all the current parameters of the fleet and the environment in order to perform the best decisions. The online heuristic algorithms presented here, intend to help them in their daily work.

In the first section we introduce several notions related with a current pickup and delivery routing and scheduling plan. Then we discuss cost-forming issues that must be taken into account by each online heuristic decision.

Next sections cover the decision-making heuristics and explain the schedule and order features they take into account. We formalize and discuss the constraints they impose. Amongst these are **Minimum Cost**, **Balanced**, **Current Orders**, **Shift Profitability**, **Geographical Closeness**, **Minimize Vehicles**, **Immediate Cost** and **Random** heuristics. We partly talk about their correctness and worst-case complexity in the main body and leave the rest in the Appendix. At the end of this chapter we summarize the main computational results.

3.1 General Assumptions

An inquiry could be requested either via phone conversation or via company webportal. In addition to these dynamic demands there is also a significant number of orders stated on previous days and the information about them is available in the morning of the working day. The dynamism of the problem is a consequence of the arrival frequency of previously unknown errands. When such a stochastic event occurs, a dispatcher needs to decide on which route of which vehicle the new order should be included so that a certain set of constraints is satisfied in some optimal way. As they follow a policy aiming to maximize the number of serviced and satisfied customers, the dispatchers expect that the system will advice them in a reasonable time. This policy is adopted by the company aiming at loosing no customers. After a dispatcher accepts a given order, he could decide to inform a driver of the recommended vehicle immediately when it is not in a motion or to buffer the new demand for a while according to a predetermined strategy. The last decision partly depends on the current system response in terms of the immediate versus delayed profit gained by assigning the new demand to that vehicle.

In the considered setting the smallest time interval within which new orders can be distinguished is one minute (we do not distinguish between orders, which arrive in different seconds within a particular minute), so the time can be considered as discretized. This issue also has other benefits, which we are going to discuss in chapter 5 where we describe the learning mechanism. We proceed more formally by introducing several notations, which we are going to use in the following sections.

Demands Let t be the moment (minute), in which new demands arrive at the system. The set L^t contains all the customer locations known by this time instant. Also let the set of all previously scheduled transportation orders be $O^t = \{o_1, \ldots, o_n\}$ and the set of new arrivals $S^t = \{s_1, \ldots, s_k\}$. Each order s_k contains information about two requests, denoted as s_k^1 and s_k^2. In addition, q_k and u_k are the commodity values associated with s_k for the number of pallets and its weight, respectively. They are positive for s_k^1 and negative for s_k^2. In other words, the positive commodities are to be transported from l_k^1 to l_k^2, which are the respective locations.

Time windows Also there are time windows at each location, which express customer convenience. Let $[t_e^{j,1}, t_l^{j,1}]$ be the one at l_j^1 and $[t_e^{j,2}, t_l^{j,2}]$ at l_j^2. Recall, we assume that $t_e^{j,2} = t_e^{j,1}$.

Fleet The new transportation orders are assigned to vehicles of the company fleet $V = \{v_1, \ldots, v_m\}$. Each fleet member v_j has given maximal capacities, Q_j and U_j, and its own depot, located at $d_j \in D$ (the set of depot locations). Vehicle availability is based on the historical data and it takes into account the driver maximum shift length of eight hours stated in their contracts. We assume three vehicle shifts, i.e. 5 a.m. - 13 p.m., 9 a.m. - 15 p.m. and 13 p.m. - 19 p.m..

Plans Let $P^t = (\mathcal{R}^t, \mathcal{I}^t)$ be the PDRSP in the time instant t. That is, each vehicle $v_j \in V$ is assigned to exactly one route R_j^t according to $\mathcal{R}^t$ following the itinerary $I_j^t \in \mathcal{I}^t$. Recall, $R_j^t = \{r_{j_1}, \ldots, r_{j_{n_j}}\}$ is an ordered set of requests, starting and ending with the respective home request ($r_{j_1} \equiv r_{j_{n_j}} \equiv r_j \in R_D$), and each intermediate demand $r_{j_k} \in R_j^t$ corresponds to an order request from R_{O^t} (the set of the assigned requests so far), either a pick-up or a delivery, which is located at $l_{j_k} \in L$.

Based on the notations above we will define some notions related with them. A vehicle route R_j^t is called *empty* if $n_j = 2$, i.e. there are no other requests assigned to it except the home ones. On the other hand, a *non-empty* route contains an even number of customer demands as we consider a PDP.

With respect to the dynamic moment t and the plan P^t we can distinguish several vehicle states. If a vehicle has finished its job, it is not available anymore that day narrowing default availability time window. Here, we assume a vehicle has finished its work if it has left the last customer on its route according to P^t. Other vehicle state is when the driver is heading towards his next destination, in which case the vehicle cannot be diverted. In time t a vehicle can also be serving a client or waiting at a customer location in order to start its duties. In both cases the driver can be consulted by a dispatcher about his next jobs. Also a vehicle may have no assigned requests at time t and be available waiting at home. Such a vehicle has a currently empty route and if it is profitable enough a driver can be contracted for the rest of the respective availability window. Such vehicle is considered as a candidate for a new order if this is to be serviced after the beginning of the driver shift time.

Regarding the vehicle states we can distingusih several types of assigned visiting sequences. If v_j has executed its route by the time t, R_j^t is *infeasible with respect to* t. Also R_j can be infeasible with respect to t if v_j starts its shift after t and there is no new order $s_k \in S^t$ to be serviced after that time. Otherwise, R_j^t is a *possible route* regarding t. Next, if v_j cannot meet the capacity constraints with respect an order $s_k \in S^t$ the route R_j^t is *infeasible with respect to* s_k. Hence, R_j^t is a *possible route*, whenever it is feasible with respect to t and to an order from S^t. When the system makes a decision where to include a new demand it has to take into consideration both the non-empty and empty feasible routes. Also, as we have a paired variant of VRP, for each $s_k \in S^t$ a decision should be made for both s_k^1 and s_k^2. The paired constraints relating these requests must be satisfied, i.e. both s_k^1 and s_k^2 must be serviced by the same vehicle v_j and l_k^1 must be visited before l_k^2. Thus, the inclusion of s_k has to be done in a sequence. We call *a gap* each pair of consecutive visits along a vehicle route. Thus, each heuristic must evaluate all gaps within the possible routes in the current plan. A gap is a *possible* place for accommodating a new demand if it satisfies a collection of constraints imposed by a heuristic. When given an order $s_k \in S^t$ each heuristic computes all possible gaps along all possible routes together with their additional costs. It also takes into account the vehicle states according to P^t. Then as candidates for accommodating the new order are considered only those pairs of spots, which have as first component a "pick-up" gap and as the second one a "delivery" gap. Finally, the pair of gaps with minimal total cost is selected. In this sense, apart from only considering order and schedule features, the heuristics try to minimize the local extra expenses.

3.1.1 Possible routes

As discussed in the previous paragraph whenever new orders arrive we need to determine which routes possibly can accommodate them. The last can be done if we first determine the possible routes with respect to the arriving moment.

Let t be the considered instant, in which order $s_k \in S^t$ arrives. Also let $P^t = (\mathcal{R}^t, \mathcal{I}^t)$ be the current plan. Then a route $R_j^t = \{r_{j_1}, \ldots, r_{j_{n_j}}\}$ from $\mathcal{R}^t$ is a *possible* candidate *with respect to* t if it satisfies the following constraints:

$$\begin{array}{ll} \text{F 1.} & d^j_{j_{n_j-1}} \geq \max(t, t_e^{k,1}) \\ \text{F 2.} & d^j_{j_1} \leq \max(t, t_l^{k,1}) \end{array}$$

Table 3.1: Route constraints : regarding the arrival moment.

Constraint F 1. requires that vehicle v_j is still executing its route, i.e. its departure time of the last customer is at least the maximum of t and $t_e^{k,1}$. Otherwise, the vehicle either will travel on its way to its depot or it will be already at home. The second constraint imposes that v_j should have started executing its duties by the maximum of t and $t_l^{k,1}$. In general, we have $t \leq t_l^{k,1}$. However, to keep generality, we assume that it is possible that $t_l^{k,1} \leq t$ when a package has been delayed for processing, but it has to be delivered the same day. Then we consider only the moving vehicles as possible candidates for that package. Note that both constraints restrict the set of all routes P^t with respect to $[t_e^{k,1}, t_l^{k,1}]$, but there is a possibility to violate $[t_e^{k,2}, t_l^{k,2}]$, albeit at the cost of additional penalties, l_k^2 can be visited either in an after-shift time or it can imply possible late services at some of the successive locations.

3.1.2 Capacity constraints

To consider a route as a possible candidate to accommodate the new demand, it must meet a set of constraints regarding the capacity dimensions of the vehicle and the new order. As we consider only two commodities, i.e. number of pallets and weight, we have only two constraints.

Let t be the moment, in which order $s_k \in S^t$ arrives. Let its commodity dimensions be q_k and u_k for the number of pallets and its weight, respectively. Also let $P^t = (\mathcal{R}^t, \mathcal{I}^t)$ be the current PDRSP in. Then a route $R_j^t = \{rj_1, \ldots, r_{j_{n_j}}\}$ from $\mathcal{R}^t$ is a *possible* candidate *with respect to* s_j if it satisfies the following constraints:

$$\begin{array}{ll} \text{C 1.} & q_k \leq Q_j \\ \text{C 2.} & u_k \leq U_j \end{array}$$

Table 3.2: Route constraints : regarding the new order capacities.

3.1.3 Additional travel costs

There might be extra expenses when a vehicle is diverted from its current state in order to service a new order. The heuristics should compute such costs and we discuss the assumptions made when calculate them.

Let us consider a route $R_j^t = \{rj_1, \ldots, r_{j_{n_j}}\}$ executed by vehicle $v_j \in V$ running with average velocity of m_j. Let further $s_k = (r_k^1, r_k^2) \in S^t$ be a new order and let r_{j_s} and $r_{j_{s+1}}$ be the visits between which we want to compute the extra costs for serving a request r_k^v being either r_k^1 or r_k^2. In addition, let $[t_e^{k,v}, t_l^{k,v}]$ be its time window. Recall, c_j' is the cost the company pays per kilometer for having v_j on the road and with c_j'' is denoted the wage of that vehicle driver. Also, recall, that s_k^j is the service time needed for r_k^p. Then the additional travel time needed by v_j is $t_k^j = t_{j_s,k}^j + t_{k,j_{s+1}}^j - t_{j_s,j_{s+1}}^j$. Furthermore let w_k^j be the waiting time at the new location l_k^p defined as follows : $w_k^j = t_e^{k,p} - (d_{j_s}^j + t_{j_s,k})$ if $d_{j_s}^j + t_{j_s,k} < t_e^{k,p}$ and 0, otherwise. Thus, the extra travel plus driver costs are given by the expression :

$$c_i^j = c_j' * (t_k^j * m_j) + c_j'' * (t_k^j + s_k^j + w_k^j) + p_k^j.$$

The first term is the cost the company pays for the additional travel distance and the second represents the extra driver wage, including waiting and service times at the new location. The last term is the penalty incurred in case of delay.

3.1.4 Time window urgency

The travel costs are not the only costs related with the new service. Visiting the new locations can imply late services at some of the successive visits, which lead to additional undesired costs. We calculate these costs under several assumptions, which we describe next on.

Let, again, t be the moment when $s_k \in S^t$ arrives and let r_k^p be one of its subdemands. Also, let $[t_e^{k,p}, t_l^{k,p}]$ be the time window at l_k^p. We assume that the length of that time window reflects the customer urgency about receiving his packages with respect to the moment they are demanded. Let $v_j \in V$ is assigned to that request and it arrives at l_k^p at time $T > t_l^{k,p}$. Thus, we define a measure of lateness as $L_p^j = \frac{T - t_e^{k,p}}{t_l^{k,p} - t_e^{k,p}}$ to express the delay at l_k^p by v_j. It has a value greater than 1 and it aims to capture the *urgent* need of the customers to receive their deliveries. For instance, a delay of 10 minutes in a time window of 30 minutes has $L = \frac{40}{30} \approx 1.33$, whereas the same delay in a time window of 5 hours leads to a value of $L = \frac{310}{300} \approx 1.03$

In addition to the lateness measure we also need to determine the costs related to a particular delay. The last is simply the serving cost according to the current scheduling plan. More formally, let v_j is late at l_h and let according to the current plan P^t the cost for serving the parcels at l_h be c_h^j. Hence, the penalty we assign to l_h is $p_h^j = L_h^j * c_h^j$.

An interesting case is when a vehicle needs an extra time after its shift. In that scenario, each driver is paid according to special regulations, which are not in the scope of this thesis. Thus, we assume he is awarded the same hourly rate stated in his contract when working under standart conditions. Hence, the after-work delay is captured as a regular delay at any of the locations. However, in this case the lateness is regarding the depot time window, which is exactly the shift time of the driver.

3.1.5 Distance measures

Both the regular travel costs and the penalties discussed above are time-dependent. However, the real continuious time might be sometimes inappropriate measure of the vehicle performance due to the possible waiting times at some locations or rare and imperfect communication between the drivers and the dispatchers, for instance. Thus, a discretized and qualified measure need to be used when we evaluate the vehicle performance between clients. Such measures are usually the distance measures related to the concrete VRP. Ideally, it would be a situation, in which we use a Geographical Information System (GIS) to retrieve the real-time road distances between any two geographical points in the considered area. In this work we do not consider such a scenario due to the absence of a proper GIS module. However, we use the Vincenty distance as a good approximation measure of the straight line between any two locations on the Earth. Notice that each road-based distance between two points can be represented as a finite linear combination of multiple Vincenty-based distances.

The depot and customer pickup and delivery locations are generated uniformly and independently of each other, and they are specified by their latitude and longitude coordinates. Let $l_j, l_k \in L \cup D$ be two locations. Then the Vincenty distance is realized via the function VincentyDistance(l_j, l_k), which is used hereafter by the implemented heuristic algorithms. As not being a major topic of our scope we leave the opportunity to the inquisitive reader to read more about the formulae in [56].

3.2 Heuristics

In the following sections we describe the heuristics we realized. Each one begins with an informal description of the considered heuristic and then it introduces the constraints the rule imposes during the evaluation. Each rule observes the current PDRSP P^t and a new order $s_k \in S^t$, and it returns a recommendation on where s_k should be included. We call each such decision a *suggestion* because it just allocates the new loads to a vehicle route. This type of decision allows us to rearrange some of the vehicle loads later on. However, a *commitment* to an order is a decision after which we cannot change the carrier of that demand.

3.2.1 Minimum cost heuristic

Minimum cost heuristic calculates the cost of including the new order between any two consecutive visits along a route of a possible candidate. In addition, all hard constraints have to be met, i.e. vehicle capacities and the earliest pickup times of both the new order and the succesive customer locations. The increase on the objective function is made up of the additional travel costs for visiting the new locations, the extra expense of a possible waiting and a certain service times at those locations, any additional penalties for late pickups and deliveries as a result of the incurred vehicle diversion, and possibly a supplementary cost for the extra time needed to make deliveries at the end of the vehicle shift. Finally, the heuristic selects the feasible gap between the visits with minimum extra expenditures.

Let the time instant, in which the new order $s_k \in S^t$ arrives be t. Let also $P^t = (\mathcal{R}^t, \mathcal{I}^t)$ be the current plan and let $R_j^t = \{r_{j_1}, \ldots, r_{j_{n_j}}\}$ be a possible route in it run by the vehicle $v_j \in V$. Furthermore, let r^t be the current position of v_j (either at some location l_{j_s} associated to a visit $r_{j_s} \in R_j^t$ or at some position between two visits). In addition, let r_k^p be one of the two requests constituting the order s_k. Then a pair of consecutive visits $(r_{j_s}, r_{j_{s+1}})$ along R_j^t with $s \in \{1, n_j - 1\}$ is a *possible* place where r_k^p can be inserted if the following constraints are obeyed:

M 1. $\forall l \geq s+1 . q_k \leq Q_j - \sum_{h=1}^{l} q_{j_h}$

M 2. $\forall l \geq s+1 . u_k \leq U_j - \sum_{h=1}^{l} u_{j_h}$

M 3. $d_{j_s}^j + t_{j_s,p}^j \geq t_e^{k,p}$

M 4. $d_{j_s}^j + t_{j_s,p}^j \leq t_l^{k,p}$

M 5. $d_p^j + t_{p,j_{s+1}}^j \geq t_e^{j_{s+1}}$

M 6. $t_{e'}^{k,p} \leq t_{l'}^{k,p}$,
where $t_{e'}^{k,p} = \max(t_e^{k,p}, t_e^{j_s} + t_{j_s p}^j)$ and $t_{l'}^{k,p} = \min(t_l^{k,p}, t_l^{j_{s+1}} - t_{p j_{s+1}}^j)$

Table 3.3: Minimum Cost constraints.

The first two constraints guarantee that the decision to include r_k^p between r_{j_s} and $r_{j_{s+1}}$ will not result in a violation of the vehicle maximal capacities, namely the number of pallets and the weight. The next pair of conditions imposes that v_j should arrive at the location l_i^k of r_i^k no earlier than the time the package is ready and no later than the latest preference of that customer. The latter condition is required only for the new demand. All the remaining visits can be assigned a positive delay at their locations. In addition, after we leave this new location we must arrive at the next scheduled visit, i.e. $r_{j_{s+1}}$, later than its earliest pick-up time in order to avoid waiting there. The last constraint, M 6., checks whether the time window at the location of r_i^k is feasible with respect to the gap $(r_{j_s}, r_{j_{s+1}})$.

Implementation

The new order s_k is composed of a pickup r_k^1 and a delivery r_k^2 requests. Thus, a fesible gap needs to be selected for both subdemands. In addition, both the pickup and the delivery locations of the new order needs to be placed somewhere after r^t and the pickup location should precede the delivery one. Let vehicle v_j be on its way to r_{j_s} or already at l_{j_s}. Moreover, let $F_j^1 = \{(p^1, q^1) \mid \exists l \geq s.(p^1, q^1) = (r_{j_l}, r_{j_{l+1}})$ is possible place for $r_k^1\}$ and $F_j^2 = \{(p^2, q^2) \mid \exists l \geq s.(p^2, q^2) = (r_{j_l}, r_{j_{l+1}})$ is possible place for $r_k^2\}$ be the sets of possible gaps along R_j^t for the pick-up and the delivery requests, respectively. We are interested in those (p^1, q^1) and (p^2, q^2) such that p^2 occurs after p^1 or $p^2 = p^1$. Among these, the system selects the one with minimum combined cost increase.

The algorithm starts with the initialization of several variables, where it will store the feasible spots and their costs, respectively. Then it determines the set of possible routes $\mathcal{R}_{pos.}^t$. In this step we check for each route $R_j^t \in \mathcal{R}^t$ whether it satisfies the constraints F 1., F 2., C 1. and C 2. The next cycle collects all the possible gaps for each possible route R_j^t in the sets F_j^1 and F_j^2 according to the constraints M 1. - M 6. and at the end of each iteration it follows an update of the global variables F^1 and F^2. After this, set F^1 contains the sets of all possible gaps for the pickup request, and similarly, for set F^2, regarding the delivery. Next, for each route in $\mathcal{R}_{pos.}^t$ we select a possible pickup gap and we summed up its cost together with the cost for each delivery gap from F^2. If this calculated cost is less than the current minimal cost, the algorithms updates its value and the currently best locations, where the new subdemands could be included.

The supplementary costs for inserting the request r_k^p between the visits r_{j_s} and $r_{j_{s+1}}$ are calculated by the function IncreaseCost$(r_{j_s}, r_{j_{s+1}}, r_k^p, R_j^t)$, which takes into account the cost for the additional travel time to the location l_k^p, the costs related with a possible waiting w_p^j and a certain service s_p^j at that location and the penalties for the possibly implied future lateness at previously planned locations, and in particular the extra expenditures for after-shift work. The IncreaseCost() function calls the distance method VincentyDistance(r_1, r_2), which returns the geodesic distance between the locations of the requests r_1 and r_2. The cost is then computed as described in Section 3.1.3. In the algorithm the value of $b_{j_s}^j$ is the start time at the location associated with the request r_{j_s} by the vehicle v_j according to its current schedule I_j^t, $d_{j_s}^j$ is the departure time at the same

location of the same vehicle and $t_l^{j_s}$ is the latest time for the request r_{j_s}. Also $c_{j_s}^j$ is the cost for serving the client at l_{j_s} by v_j according to P^t and the fraction $\frac{d}{m_j}$ is the average time, within which v_j elapses the distance d.

Algorithm 3.2.1: MINIMUMCOST(P^t, s_k)

---

comment: *The algorithm MinimumCost returns the feasible gaps for s_k*

comment: *in a route from $\mathcal{R}^t$ with minimal extra cost.*

1. $F^1 = \emptyset, F^2 = \emptyset, C^1 = \emptyset, C^2 = \emptyset$
2. $min\_cost = \max$
3. min_pickup, min_del

4. $P^t_{pos.} =$ POSSIBLEROUTES$(\mathcal{R}^t, t, s_k)$

5. **for** $R_j^t \in \mathcal{R}^t_{pos.}$
6. $\quad F_j^1 = \emptyset, F_j^2 = \emptyset, C_j^1 = \emptyset, C_j^2 = \emptyset$
7. $\quad$ **for** $e = s$ **to** n_j
8. $\quad\quad$ **if** POSSIBLEGAP$(r_{j_e}, r_{j_{e+1}}, r_k^1)$
9. $\quad\quad\quad c^1_{ee+1} =$ INCREASECOST$(r_{j_e}, r_{j_{e+1}}, r_k^1, R_j^t)$
10. $\quad\quad\quad C_j^1 = C_j^1 \cup \{c^1_{ee+1}\}$
11. $\quad\quad\quad F_j^1 = F_j^1 \cup \{(r_{j_e}, r_{j_{e+1}})\}$
12. $\quad\quad$ **if** POSSIBLEGAP$(r_{j_e}, r_{j_{e+1}}, r_k^2)$
13. $\quad\quad\quad c^2_{ee+1} =$ INCREASECOST$(r_{j_e}, r_{j_{e+1}}, r_k^2, R_j^t)$
14. $\quad\quad\quad C_j^2 = C_j^2 \cup \{c^2_{ee+1}\}$
15. $\quad\quad\quad F_j^2 = F_j^2 \cup \{(r_{j_e}, r_{j_{e+1}})\}$
16. $\quad F^1 = F^1 \cup F_j^1, F^2 = F^2 \cup F_j^2$
17. $\quad C^1 = C^1 \cup C_j^1, C^2 = C^2 \cup C_j^2$

18. **for** $R_j^t \in \mathcal{R}^t_{pos.}$
19. $\quad$ **for** $(p^1, q^1) \in F_j^1$
20. $\quad\quad$ **for** $(p^2, q^2) \in F_j^2$
21. $\quad\quad\quad$ **if** $p^1 \geq p^2$
22. $\quad\quad\quad\quad cost = c^1 + c^2$
23. $\quad\quad\quad\quad$ **if** $cost \leq min\_cost$
24. $\quad\quad\quad\quad\quad min\_cost = cost$
25. $\quad\quad\quad\quad\quad min\_pickup = (p^1, q^1)$
26. $\quad\quad\quad\quad\quad min\_del = (p^2, q^2)$

27. **return** $(\{min\_pickup, min\_del\})$

Complexity and correctness

Here we discuss the correctness and the complexity of the algorithms. The heuristic correctness depends mostly on the correctness of the methods PossibleRoutes(), PossibleGap() and IncreaseCost(). The first procedure checks whether a route is a "good" candidate for the new loads. The last is done with respect to the current plan P^t. If it does, then the procedure includes it in $\mathcal{R}^t_{pos.}$, otherwise not. The correctness of this method is partly ensured by the fact we have a strong system of inequalities F 1., F 2., C 1. and C 2.. The second function checks whether a gap is a possible place according to the conditions M 1. – 6.. Note that the last two procedures depend on the quality of the $\mathcal{R}^t$. At the beginning of the working day we start with an optimal schedule for all the predetermined orders. Such a plan is of high quality and we have to quarantee that applying an heuristic, thereafter, it preserves a collection of some general assumptions.

Algorithm 3.2.2: IncreaseCost$(r_{j_e}, r_{j_{e+1}}, r^p_k, , R^t_j)$

---

comment: *The function computes the extra cost for inserting r^p_k between the*

comment: *visits r_{j_e} and $r_{j_{e+1}}$.*

1. $extra\_cost = 0$
2. $d_{j_e,p} =$ VincentyDistance(r_{j_e}, r^p_k)
3. $d_{p,j_{e+1}} =$ VincentyDistance$(r^p_k, r_{j_{e+1}})$
4. $d_{j_e,j_{e+1}} =$ VincentyDistance$(r_{j_e}, r_{j_{e+1}})$
5. $d = d_{j_e,p} + d_{p,j_{e+1}} - d_{j_e,j_{e+1}}$
6. $m_j =$ the average velocity of v_j [km/h]
7. $c'_j =$ the cost the company pays when v_j is on the road [$/km]
8. $c''_j =$ the hourly driver wage v_j [$/h]

9. $extra\_cost \mathrel{+}= d * c'_j + \frac{d}{m_j} * c''_j$

10. $w^j_p = (t^{k,p}_e - d^j_{j_e,p} - \frac{d_{j_e,p}}{m_j} > 0)?\ t^{k,p}_e - d^j_{j_e,p} - \frac{d_{j_e,p}}{m_j} : 0$
11. $s^j_p =$ the service time at l^p_k

12. $extra\_cost \mathrel{+}= (w^j_p + s^j_p) * c''_j$

13. **for** $r_{j_e} \in R^t_j \wedge r_{j_e}$ is visited after r^p_k
14. $\quad l^j_{j_e} = b^j_{j_e} + \frac{d}{m_j} - t^{j_e}_l$
15. $\quad$ **if** $l^j_{j_e} > 0$
16. $\quad\quad L^j_{j_e} = \frac{b^j_{j_e} + \frac{d}{m_j} - t^{j_e}_e}{t^{j_e}_l - t^{j_e}_e}$
17. $\quad\quad extra\_cost \mathrel{+}= L^j_{j_e} * c^j_{j_e}$

18. **return** $(extra\_cost)$

For instance, a vehicle capacity must not be exceeded at any time point or we cannot actuate a vehicle, which has finished its duties or its availability window opens later. The latter is ensured by computing the set of possible routes in $\mathcal{R}^t$. The reader can find more details about the quality of the initial schedule as discussed in Section 4.3.1. Lastly, the correctness of the function IncreaseCost() is guaranteed as the Vincenty measure is a positive one and it satisfies the triangle inequality. This is important as the value of d at line 5. in the algorithm must be greater than or equal to zero. The gaps returned by the MinimumCost() heuristic have their total extra cost minimal. However, note that there might be a gap for either the pick-up or the delivery requests with lower extra cost. For instance, such pairs could have the delivery visit preceding the pick-up one. Also note that the gaps for the pick-up and the delivery demands could concide. In the latter case the pick-up request, again, must be visited before the delivery one. Thus, the described algorithm produces the minimum insertion cost with respect to $s_k \in S^t$, which is somehow distributed between the cost for inserting the pickup location and the cost for inserting the delivery location.

About the complexity of the presented algorithm, we first determine the possible routes in line 4. of the MinimumCost() algorithm. This operation requires time linear to the fleet size m, i.e. $O(m)$, as checking the constraints F 1., F 2., C 1., C 2. for every route from $\mathcal{R}^t$ and order $s_k \in S^t$ requires a constant time. Further, collecting all possible gaps and their costs (lines $5. - 17.$) for both the pickup and the delivery request needs a time proportional to the maximal number of visits in $\mathcal{R}^t$. Hence, if $n_{\max}$ is the number of visits in the longest route in $\mathcal{R}^t$, then the complexity of selecting all these spots for a subdemand in the worst-case is $O(m * n^2_{max})$. For each gap the function PossibleGap() takes $O(n_{\max})$ because of constraints M 1. and M 2.. The calculation of extra costs is done using IncreaseCost() for all "good" gaps. It requires a worst-case complexity boundary of $O(n_{\max})$ caused by the calculation of the penalties along the route, which are the result of the possible vehicle diversion between the considered pair of visits. In lines $18. - 27.$ of the algorithm the minimal extra cost amongst all pairs of gaps satisfying the precedence constraint is computed. This calculation needs $O(m * n^2_{\max})$ time in the worst-case scenario. Then the overall complexity according to the worst-case scenario is thus $O(m * n^2_{\max})$. Better complexity results are achieved in practice due to the fact that we consider proper parts of any of the routes in $\mathcal{R}^t_{pos}$. For a fixed route and a fixed visit in it we need to perform a "goodness" test only for the successive visits. Also the set of possible routes could have cardinality several times smaller than the power of $\mathcal{R}^t$. Let it be $m_{pos.} = \mid \mathcal{R}^t_{pos.} \mid$. Thus, lines $5. - 17.$ can be processed in time $o(m_{pos.} * n_{\max})$, where this time $n_{\max}$ is the maximal number of visits in the tail of a route from $\mathcal{R}^t_{pos.}$. The reason for that is because we start to look for a convenient place from the current position of the corresponding vehicle onwards. Last observation is expressed by setting the counter $e = s$ **to** n_j to iterate over parts of the routes rather than the whole routes. Hence, a rough approximation of the lower complexity bound is $o(m_{pos.} * (k * (\frac{n_{max}-1}{2}) + (n_{max} - k) * (\frac{n_{max}-1}{2} + n_{max})))$, where $k \in [1, n_{\max}]$ is the number of the gaps passed the "goodness" test along the route with $n_{\max}$ visits. Thus, $\Theta(m * n^2_{\max})$ could be given as an overall complexity of the decision-making procedure.

3.2.2 Balanced heuristic

The Balanced heuristic aims to distribute the overall workload amongst the vehicles. First it computes the set of all vehicles, which are able to accommodate the given order without violating any of the hard (earliest times and vehicle capacity) and soft (latest times) constraints at the current and the successive visits. If this set is empty then we consider the set of all vehicles that have feasible routes. For every vehicle from this set we compute its future planned work according to the current pickup and delivery routing and scheduling plan. Then the heuristic selects the vehicle with minimum amount of scheduled work and the new order is inserted in a place along its route with minimal additional costs. An interesting case is when more than one vehicle have the same minimum amount of future work. Then the new order should again be included with minimum outlay.

Let the new order $s_k = (r_k^1, r_k^2) \in S^t$ arrives at time t and let $\mathcal{R}^t = \{R_1^t, \ldots, R_m^t\}$ be the set of open routes in time instant t. Let $R_j^t = \{r_{j_1}, \ldots, r_{j_{n_j}}\}$ be a possible route for s_k and let the vehicle assigned to it be $v_j \in V$. Furthermore, let the current position of v_j be r^t. A pair of consecutive visits $(r_{j_s}, r_{j_{s+1}})$ along R_j^t with $s \in \{1, n_j - 1\}$ is a *possible* place for r_k^p $(p = 1, 2)$ if the following conditions are met:

B 1. $\forall l \geq s+1 . q_k \leq Q_j - \sum_{h=1}^{l} q_{j_h}$

B 2. $\forall l \geq s+1 . u_k \leq U_j - \sum_{h=1}^{l} u_{j_h}$

B 3. $d_{j_s}^j + t_{j_s,p}^j \geq t_e^{k,p}$

B 4. $d_{j_s}^j + t_{j_s,p}^j \leq t_l^{k,p}$

B 5. $d_p^j + t_{p,j_{s+1}}^j \geq t_e^{j_{s+1}}$

B 6. $d_p^j + t_{p,j_{s+1}}^j \leq t_l^{j_{s+1}}$

B 7. $t_{e'}^{k,p} \leq t_{l'}^{k,p}$,
where $t_{e'}^{k,p} = \max(t_e^{k,p}, t_e^{j_s} + t_{j_s,p}^j)$ and $t_{l'}^{k,p} = \min(t_l^{k,p}, t_l^{j_{s+1}} - t_{p,j_{s+1}}^j)$

Table 3.4: Balanced constraints.

The constraints B 1., B 2. ensure the future load satisfaction as in the MinimumCost() heuristic. The following pair of conditions requires that the hard and soft constraints of the new demand are fulfilled. In other words the service must take place within its time preferences. The same check is performed with respect to the successive scheduled visit, i.e. $r_{j_{s+1}}$ and the last requirement guarantees the feasibility of the time window at the location of the new request with respect to the gap between r_{j_s} and $r_{j_{s+1}}$. Note that satisfying the above constraints means that if we divert v_j to serve r_k^p it would not cause any late penalties or after-shift time onwards. The reason for this is that serving the new packages does not change the current vehicle schedule I_j^t. Moreover, in order to assign $s_k \in S^t$ to v_j, R_j^t needs to have either at least two different "good" gaps or to have one into which can fit both the pickup and the delivery requests.

3.2.3 Current orders heuristic

The aim of this heuristic is to entirely reroute the currently known orders whenever a new demand arrives at the system. To perform such an update we use the VRP solver Indigo 2.0. The process of reordering the orders in the system consists not only to schedule the new demand into some of the routes, but also to remove already serviced orders. The new schedule must also obeys some additional constraints, which we discuss next. The entire set of currently scheduled orders needs to be split into three sets, i.e. set of *serviced*, set of *undelivered* and set of *unserviced* customers. The set of serviced customers should not influence future reorderings of the rest transportation requests. Furthermore, for each current ordering assigned to a vehicle a list of undelivered customers will be created. They should be visited by the same vehicle in an order possibly different from the one established in the current schedule. Also the set of unserviced customers will be taken into account when the heuristic creates the new plan with the opportunity of schedule them into routes different from the current ones.

In addition to the heuristic description we explain how Indigo can be used only as a decision heuristic. Recall, the solver has two-phased architecture. During the first phase the initial routes are constructed using a simple insertion heuristic (*construction phase*). Next, the improvement of this initial solution is performed by means of an heuristic combination of local search methods. For instance, for producing the optimal solutions in our experiments, we set a combination of *Large Neightbourhood Search* heuristic together with *Simulated Annealing* heuristic. In order to use and later on evaluate only the simple insert heuristic provided by Indigo, every time we apply the **CurrentOrders** rule we update the current schedule without using any improvement strategies. In this section we will use the terms *PDRSP* and *solution* interchangeably as the result of applying the heuristic will be partially a new plan rather than just the result of including the new order somewhere along a route.

Next, let the new order $s_k \in S^t$ arrives at time t and let P^t be the current solution. In this moment each vehicle $v_j \in V$ is somewhere along its route, either at its depot or performing a service or travelling towards its next customer. Recall, in the latter case we assume that the vehicle is at the next destination. Let in the stated three situations the current vehicle position of v_j along R_j^t be r_{j_c}. Then the following conditions must be met during the rerouting update.

The first constraint is an interesting one. It requires that the vehicle has new initial departure time and it is the one at the location l_{j_c} stated in current solution P^t. As usual, the next two constraints quarantee that inserting the new demand within the new route, assuming without loss of generality to be again R_j^t, will not affect in violating the capacity constraints of any future visits in it. The fourth and the fifth restrictions impose that the new order, if included, should not cause after-shift working time. Note the we allow v_j to arrive at the location of r_k^p earlier than its earliest pick-up time, in which case it will wait there. The reason for that is the internally implemented $DriveFirst$ ([43]) waiting strategy (in Indigo), which means that as soon as vehicle finishes its work with one customer, it start heading immediately towards the next one. We also permit to occur a situation where v_j

C′ 1. $\forall v_j . d^j_{j_1} = d^j_{j_c}$

C′ 2. $\forall l \geq s+1 . q_k \leq Q_j - \sum_{h=1}^{l} q_{j_h}$

C′ 3. $\forall l \geq s+1 . u_k \leq U_j - \sum_{h=1}^{l} u_{j_h}$

C′ 4. $d^j_{j_1} + t^j_{j_1,p} \leq d^j_{j_{n_j}}$

C′ 5. $d^j_p + t_{p,j_2} \leq d^j_{j_{n_j}}$

C′ 6. $Q_{j,new} = Q_j - \sum_{r_{j_s} \in R^t_j} q_{j_s}$, where r_{j_s} is an undelivered demand along R^t_j

C′ 7. $U_{j,new} = U_j - \sum_{r_{j_s} \in R^t_j} u_{j_s}$, where r_{j_s} is an undelivered demand along R^t_j

Table 3.5: Current Orders constraints.

arrives too late to perform the service at the new locations. In this case, Indigo will assign a penalty for late services at those locations. It might also happen that v_j arrives later on r_{j_2} according to the current construction cycle of the solver. The reason for that could be the diversion of v_j to perform the new service. In that case the solver will postpone the visit at l_{j_2}, if possible, it will assign r_{j_2} on some of the other routes. The last is induced by the atempts of Indigo to assign the requests without violating their time preferences. If not possible, it will calculate a lateness penalty, if specified a penalty strategy, otherwise it will leave the new order as an unassigned. Finally, the last two constraints express the fact that when Indigo constructs the new solution it should take into account that it might be the case that v_j has already loaded some goods in its rack accumulated by previously serviced pickups, which are still not delivered by the time t. A consequence of this is a decreased initial maximal capacity of v_j in the new solution. The difference is made up by the accumulated value of the capacities for all undelivered requests within R^t_j by the time t. It is the same as a situation, in which v_j starts its working day with already assigned requests, i.e. with a non-empty rack.

3.2.4 Shift profitability heuristic

The Shift Profitability dispatch rule aims to maximize the profit gained during the driver shift in terms of both the number of served customers and the total execution time. Recall, the company pays to the driver minimum hourly rate. Thus, it might be interested in minimizing the number of hours a driver stays on the road as well as maximizing the number of served customers along his route. For instance, if a driver is supposed to leave its last customer at 3 : 15 p.m. according to the current plan, then he will be awarded the whole hourly rate until 4 p.m.. Hence, we can try to assign more visits to his route as long as he could return to the depot no later than 4 p.m.. On the other hand, if there are a significant number of orders and this driver is the best candidate for them, we allow his route to be extended even after 4 p.m. as it seems the vehicle is doing a good work. No extension is

allowed when the end-shift time boundary is reached. This heuristic also tries to assign the new requests into some waiting time windows along a vehicle route, if such exist. A waiting time window along a route of a vehicle is feasible if all the hard constraints are met. Thus, we allow lateness and we compute its penalty, if arised. We call a route, which comply with the above observations *profitable*. Whenever a new demand arrives at the system we compute the set of the profitable routes and, finally, the new errand is included into the one with minimal cost.

Let $s_k \in S^t$ be the new order, which arrives at the time instant t. Let also the current plan be P^t and let we consider the vehicle v_j assigned to the non-empty route $R_j^t = \{r_{j_1}, \ldots, r_{j_{n_i}}\}$. Let further the current position of v_j be r^t and r_k^p be either the pickup or the delivery requests of s_k. Finally, let $(r_{j_s}, r_{j_{s+1}})$ be a pair of visits along R_j^t after r^t. In order to classify this gap as a possible place where s_k can be inserted the next several barriers must not be overstep.

$$
\begin{array}{ll}
\text{S 1.} & a^j_{j_{s+1}} < t_e^{j_{s+1}} \text{ for } s+1 \leq n_j - 1 \\
\text{S 2.} & \forall l \geq s+1 . q_k \leq Q_j - \sum_{h=1}^{l} q_{j_h} \\
\text{S 3.} & \forall l \geq s+1 . u_k \leq U_j - \sum_{h=1}^{l} u_{j_h} \\
\text{S 4.} & d^j_{j_s} + t^j_{j_s,p} \geq t_e^{k,p} \\
\text{S 5.} & d^j_p + t^j_{p,j_{s+1}} \geq t_e^{j_{s+1}} \\
\text{S 6.} & a^j_{j_{s+1}} \leq t_l^{j_{s+1}} \text{ if } s+1 = n_j \\
\text{S 7.} & t_l^{j_{s+1}} - (d^j_{j_{s+1}} - d^j_{j_1}) \notin \mathbf{N} \text{ for } s+1 = n_j
\end{array}
$$

Table 3.6: Shift Profitability constraints.

S 1. imposes the existence of a non-zero waiting time window at the location associated with $r_{j_{s+1}}$, i.e. $l_{j_{s+1}}$. Recall, $w^j_{j_{s+1}} = t_e^{j_{s+1}} - a^j_{j_{s+1}}$, which is zero whenever $a^j_{j_{s+1}} \geq t_e^{j_{s+1}}$ and non-zero otherwise. The last is neccesary condition to consider the gap between r_{j_s} and $r_{j_{s+1}}$ at all. In the case $s+1 = n_j$ this constraint is not required as it would mean that we require the vehicle to arrive at its final location before its start-shift time, i.e. $t_e^{j_{n_j}}$. However, if S 1. is met we must guarantee that including r_k^p between those two visits will not cause any future capacity disbalances along R_j^t, which is expressed by S 2. and S 3.. The next two limitations take into account the satisfiability of the earliest times both at the new location, i.e. l_k^p and at the successive one, i.e. $l_{j_{s+1}}$. Note that by satisfying them in the same time we do not forbid v_j to be late at the locations scheduled onwards. Hence, in this case a penalty is added. However, with S 6. we do not allow after-shift working time and, consequently, no penalties are allowed at the end of R_j^t. The last condition imposes the difference between the end-shift time $t_l^{j_{n_j}}$ and the total execution time of v_j not to be a natural number. In case it is, v_j could not be able to handle a new request. If obeyed, this limitation allows to classify the last spot of R_j^t as a possible "host" for the new loads.

3.2.5 Geographical closeness heuristic

Time features of the current schedule are important characteristics, but one could also take into acount the spatial distribution of the fleet. The Geographical Closeness heuristic does exactly this. The rule computes the subset of nearby vehicles to both pickup and delivery subdemands and tries to assign them to such a candidate, satisfying all soft constraints. It starts with predefined areas around the new locations and if no vehicles are there it iteratively enlarges the zones until a proper fleet member is found. The search for nearby vehicles ends after a particular value of the area parameter is reached.

Area clustering

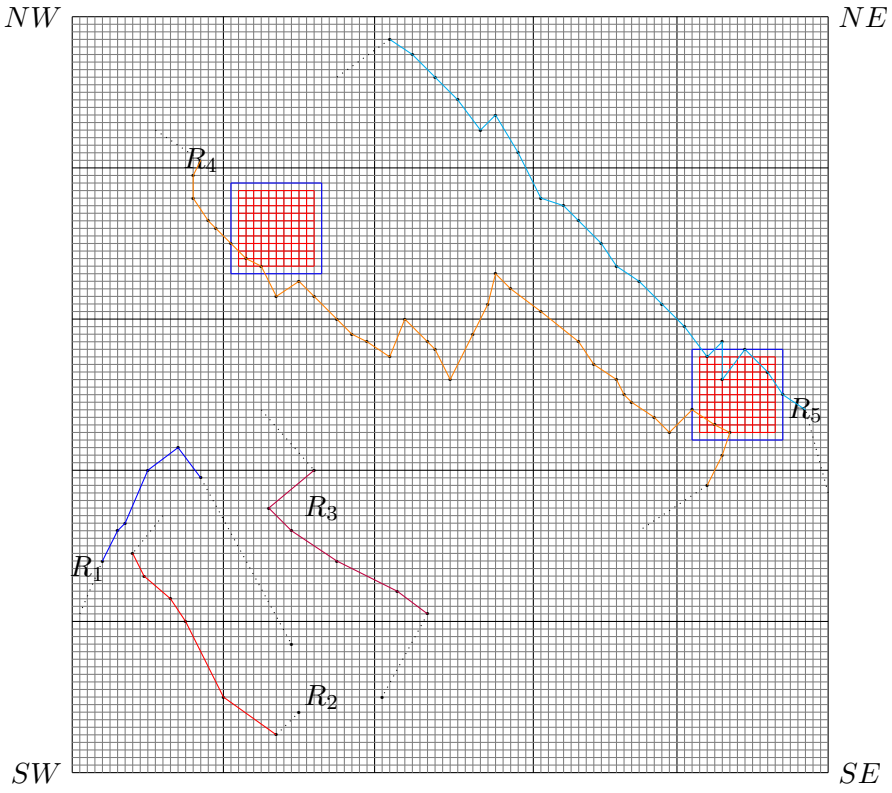


Figure 3.1: Rasterized area of Sydney.

We consider a raster over the metropolitan area of Sydney defined with two parameters. The first one is the density of a network along the Earth parallels locked between 33.4 S° and 34.4 S° latitude coordinates and the second number expresses the intensity of a network alongside the Earth meridians locked between 150.67 W° and 151.67 W° longitude coordinates. Let those parameters are $x = 100$ and $y = 100$, respectively. In general, these quantitative raster dimensions could have arbitrary integer values. However, the numbers here are appropriately selected taking into account the average vehicle velocity and the size

of the smallest covered area, i.e. quadrant. Recall, most members of our fleet develops 50 kilometers per hour on the road except bicycles (15 km/h) and motorbikes (35 km/h). Using the above perameters each pixel in our raster covers an area of approximately 53 square metres or it has a linear size close to 7.3 kilometres.

In Figure 3.1 we have five routes (R_1, R_2, R_3, R_4 and R_5) from the current plan $\mathcal{R}^t$ in time t when a new order $s_k \in S^t$ arrives at the company. The areas in red color are the considered areas around the new locations (l_k^1 on the left side and l_k^2 on the right side). We observe that the only route passing by both the request areas is R_4 and, hence, it is a nearby route with respect to s_k assuming that its assigned vehicle is moving in the south-east direction.

Let $s_k \in S^t$ be the new order, which arrives at the time instant t. Let us consider the vehicle v_j assigned to the route $R_j^t = \{r_{j_1}, \ldots, r_{j_{n_j}}\}$. Let the current position of v_j be r^t. Furthermore, let r_k^p be a request of s_k and $r_{j_s}\ r_{j_{s+1}}$ be two consecutive visits along R_j^t after r^t. Then the gap $(r_{j_s}, r_{j_{s+1}})$ is *feasible* for insertion if the next system of inequalities is satisfied.

G 1.	$\forall l \geq s+1. q_k \leq Q_j - \sum_{h=1}^{l} q_{j_h}$
G 2.	$\forall l \geq s+1. u_k \leq U_j - \sum_{h=1}^{l} u_{j_h}$
G 3.	$d_{j_s}^j + t_{j_s,p}^j \leq t_l^{k,p}$
G 4.	$d_p^j + t_{p,j_{s+1}}^j \leq t_l^{j_{s+1}}$
G 5.	$t_{e'}^{k,p} \leq t_{l'}^{k,p}$, where $t_{e'}^{k,p} = \max(t_e^{k,p}, t_e^{j_s} + t_{j_s,p}^j)$ and $t_{l'}^{k,p} = \min(t_l^{k,p}, t_l^{j_{s+1}} - t_{p,j_{s+1}}^j)$

Table 3.7: Geographical Closeness constraints.

G 1. and G 2. express the usual commodity conditions, G 3. and G 4. ensure there are no delays, and, finally, the time window feasibility check is performed.

3.2.6 Minimize vehicles heuristic

All of the previous heuristics somehow minimize the additional costs needed whenever a new order arrives at the system allowing the use of as many vehicles as convenient. The possibility of running an empty available vehicle is quite high, albeit in reality this would mean an extra driver contract, which additionally increases the costs and, therefore, if might be undesired. Another issue is that sometimes we prefer paying more in order to assign the new errand to a moving vehicle than actuating a new one. This motivated us to design the Minimize Vehicle heuristic. It aims to keep the number of vehicles as low as possible still avoiding unwanted lateness. In other words, whenever a client orders a service only moving vehicles, able to accommodate the new order are considered unless no such a fleet member exists. During the schedule evaluation process we require preservation of the soft constraints

in order to minimize the final lateness. Also this dispatch rule imposes the requirement on the shift deadline to be satisfied when a suitable place for the new demand is evaluated.

Let $s_k \in S^t$ be a new order, which must be transportated between the nodes l_k^1 and l_k^2 in the urban network. Recall, $P^t = (\mathcal{R}^t, \mathcal{I}^t)$ is the current plan and each route $R_j^t = \{r_{j_1}, \ldots, r_{j_{n_j}}\}$ in it is an ordered set of requests. Let the current position of v_j be r^t and let r_k^p be one of the two new requests. A gap $(r_{j_s}, r_{j_{s+1}})$ after r^t is *feasible* if the following requirements are fulfilled :

MV 1.	$\forall l \geq s+1. q_k \leq Q_j - \sum_{h=1}^{l} q_{j_h}$
MV 2.	$\forall l \geq s+1. u_k \leq U_j - \sum_{h=1}^{l} u_{j_h}$
MV 3.	$d_p^j + t_{p,j_{s+1}}^j \leq d_{j_{n_j}}^j$
MV 4.	$d_p^j + t_{p,j_{s+1}}^j \leq d_{j_{n_j-1}}^j$ whenever $n_j \geq 2$
MV 5.	$d_p^j + t_{p,j_{s+1}}^j \leq t_l^{j_{s+1}}$
MV 6.	$t_{e'}^{k,p} \leq t_{l'}^{k,p}$, where $t_{e'}^{k,p} = \max(t_e^{k,p}, t_e^{j_s} + t_{j_s,p}^j)$ and $t_{l'}^{k,p} = \min(t_l^{k,p}, t_l^{j_{s+1}} - t_{p,j_{s+1}}^j)$

Table 3.8: Minimize Vehicles constraints.

As usual MV 1. and MV 2. guarantees the satisfaction of the capacity constraints. The next requirement imposes that v_j must be able to perform service at l_k^p by the shift deadline for that vehicle, i.e. $d_{j_{n_j}}^j$. This constraint itself would allow some lateness at the particular location, but MV 5. avoids such possibility. The latter expresses that the new service must take place no later than the latest time at location $l_{j_{s+1}}$. However, v_j is allowed to be late at the new location[1]. The next constraint MV 4. is needed as it ensures the gap $(r_{j_s}, r_{j_{s+1}})$ to be categorized as feasible only if v_j has not left its last customer (,otherwise v_j would be unavailable for the rest of the day). The last MV 6. limitation requires that the modified time window at l_k^p is feasible taking into account the additional travel time of v_j. Note that we put softer requirements on gap feasibility than usual. The rationale behind it is due to the limited number of considered vehicles. In case of an empty vehicle the constraint MV 4. is naturally not required. Also note that using the collection MV 1. - MV 6. allows waitings to arise. However, the considered subfleet will be comprised of moving vehicles, the total waiting time can only be reduced.

[1]This case is pretty much unlikely to occur as it would mean that the corresponding client has required very tight time preferences for his demand. The inquisitive reader is refered to [14], where the authors discuss a class of DPDPs with tighter client preferences or so-called *Dial-a-Ride* problems. It is a class of problems arising when an urgent transportation of handicapped people is needed. The demands in these problems are characterized with their smooth arrival frequency and very tight time windows.

3.2.7 Immediate cost heuristic

The Immediate Cost heuristic is related to the time a particular order stays in the system. Its goal is to minimize this time whenever a new demand occurs. This is ensured by assigning the new demands to a route where it can be executed as soon as possible. For this purpose, we compute the set of possible routes where the new order can be included and, in addition, the heuristic evaluates the new demand costs with respect to the first "good" spot no matter of the implied delay at some or all of the next locations. The only obeyed constraints are the vehicle shift time as well as the hard and soft constraints at the new locations. Finally, the route with minimum additional costs where the new errand can be inserted is returned. The intuition behind this heuristic is that it would give us a possibility to estimate the immediate reward of assigning the new order. All of the previous heuristics are guided by the minimality of the aditional costs, which often results in later services. However, this increases the time a particular request spends in the system, i.e. the *system total workload*. The implemented new rule tries to achieve a trade-off between that time and the additional expenses for the new obligations. In reality, it might be usefull to know how much we are going to lose if we visit a particular service as soon as possible against a situation, in which we service it later on. The ratio of these estimates, i.e. immediate and delayed reward, can be acceptable within particular boundaries and, hence, we could make a decision to violate some of the constraints imposed by the other heuristics.

Traditionally, let $s_k \in S^t$ be a client request, $P^t = (\mathcal{R}^t, \mathcal{I}^t)$ be the current plan, where each route $R_j^t = \{r_{j_1}, \ldots, r_{j_{n_j}}\}$ is an ordered set of requests. Also let current position of v_j is r^t and let r_k^p is a subdemand of s_k. A gap $(r_{j_s}, r_{j_{s+1}})$ after r^t is *feasible* if the following obligations hold :

I 1.	$\forall l \geq s+1 . q_k \leq Q_j - \sum_{h=1}^{l} q_{j_h}$
I 2.	$\forall l \geq s+1 . u_k \leq U_j - \sum_{h=1}^{l} u_{j_h}$
I 3.	$d_p^j + t^j_{pj_{s+1}} \leq d^j_{j_{n_j}}$
I 4.	$d^j_{j_s} + t^j_{j_s,p} \geq t_e^{k,p}$
I 5.	$d^j_{j_s} + t^j_{j_s,p} \leq t_l^{k,p}$
I 6.	$t_{e'}^{k,p} \leq t_{l'}^{k,p}$, where $t_{e'}^{k,p} = \max(t_e^{k,p}, t_e^{j_s} + t^j_{j_s,p})$ and $t_{l'}^{k,p} = \min(t_l^{k,p}, t_l^{j_{s+1}} - t^j_{p,j_{s+1}})$

Table 3.9: Immediate Cost constraints.

I 1. and I 2. prevent arising future load disbalances in the rack of v_j. I 3. forbids the driver to work after his shift and, hence, no penalties related to such a violation are raised. The following pair ensures on-time services at the new location and the last one checks the initial time preferences for feasibility with respect to the schedule $I_j^t \in \mathcal{I}^t$.

3.2.8 Random heuristic

The last implemented dispatch rule is the Random heuristic. As one can guess by its name it selects randomly one of the other heuristics and it activates it whenever a new order arrives at the system. It uses a uniform random generator as we would like each of the above seven heuristics to have an equal chance to be performed. Such randomization leads us to a behaviour which is much closer to the real situation in the company, than applying systematically one particular heuristic.

3.2.9 Weaken heuristic constraints

We implemented eight dispatch rules, each of which considers different characteristics of the current schedule. In addition, every heuristic imposes a collection of requirements to be fullfiled in order a vehicle to guarantee service at the new customer locations. We described each such set of conditions and discussed their meaning. Nevertheless, we do not address a situation when a given rule fails in giving a recommendation. In practice, one would be interested not only whether a particular heuristic fails, but also how much does it fail? Thus, it might be beneficial to have a way according to which we could compare the failure of the heuristics. For instance, in the worst case, given an order $s_k \in S^t$ and the current plan P^t, let all the heuristics defined above fail to give a recommendation. It is a rare situation, unlikely to occur, but it is possible and we are interested in how one can handle or interprete it.

Table 3.10 shows the constraints we removed from each heuristic to make it feasible and we next explain the motivation behind our choice. The weaker heuristic requirements are only applied to those gaps, infeasible with respect to the initial system of constraints. The latter implies that we smoothly deviate from the set of possible routes, eliminating less places (where the new client could be visited) than before. Thus, we obtain a set of routes, which lies somewhere between $\mathcal{R}^t_{pos.}$ and $\mathcal{R}^t_x$, where $x \in \{acc., \min, prof., geo., act.\}$.

$h \in \mathcal{H}$	removed constraints
MIN	M 3. : $d^j_{j_s} + t^j_{j_s,p} \geq t^{k,p}_e$
BAL	B 3. : $d^j_{j_s} + t^j_{j_s,p} \geq t^{k,p}_e$
CUR	$-$
SHIFT	S 4. : $d^j_{j_s} + t^j_{j_s,p} \geq t^{k,p}_e$
GEO	G 4. : $d^j_p + t^j_{p,j_{s+1}} \leq t^{j_{s+1}}_l$
MAV	MV 5. : $d^j_p + t^j_{p,j_{s+1}} \leq t^{j_{s+1}}_l$
IMM	I 4. : $d^j_{j_s} + t^j_{j_s,p} \geq t^{k,p}_e$
RAND	see above

Table 3.10: Weaken constraints.

Relaxing M 3., B 3., S 4. and I 4. is justified by noting that the satisfaction of the hard time window constraints, i.e. delivering packages after the earliest customer convenient time filters out a big number of available empty routes, which possibly could host the new order. For instance, if a vehicle (v_j) starts work at 9 a.m. (i.e. $d^j_{j_1}$) and there is an order with

its earliest time 9 : 20 a.m. (i.e. $t_e^{k,p}$), and if the travel time of that vehicle is 10 minutes (i.e. $t_{j_1,p}^{j}$), keeping the hard constraint would classify that vehicle as infeasible candidate for the new errand. Therefore, removing this constraint we allow that vehicle to be considered and, even though it has to wait at the new location some time (10 minutes), it might be the best candidate for that particular demand in terms of geographical location or costs. Similar situation occurs when the vehicle is not empty. Let it be again v_j and let it is at its fourth visit, where according to the current plan it is supposed to leave at 9 : 50 p.m.(i.e. $d_{j_4}^{j}$). The travel time of that vehicle is again 10 minutes (i.e. $t_{j_4k}^{j}$) and the release date of the new demand is 10 : 20 a.m.. Hence, the vehicle should wait 20 minutes before it could perform service. Removing that barrier we allow such a waiting to occur. If arised and, therefore, some late services are emerged, then we compute the respective penalty as before. In MinimumCost() dispatch rule that is allowed, so the extra costs would be made up of the additional waiting time and the implied longer delay at some of the successive locations. A comparable situation is with ShiftProfitability() and ImmediateCost() rules. In the first one the vehicle could be late at a new client location only if the waiting window is long enough or if the new demand is assigned at the end of a route. In the latter case we have the same reasoning as for the MinimumCost() heuristic. However, this is not the case with Balanced() heuristic. We keep the requirements for satisfiability of the soft constraints, which guarantee that no penalties are arised, except a cost for possibly longer driver waiting time. We remove no constraint from the CurrentOrders() dispatch rule. Indigo handles all the hard and soft constraints. Next, from the system of constraints for *GeographicalCloseness* rule we get rid of G 4. and, hence, we allow lateness to arise at some of the successive customer locations. Note, that in the same time we keep the requirement to have on-time services at the new locations. In the system of conditions for MinimizeVehicles() rule we eliminate MV 5.. As a consequence an additional delay is incurred at some of the next locations. Therefore, we compute a penalty for increasing the total customer inconvinience.

3.3 Summary

We implemented eight dispatch heuristics, which take into account the current status of the fleet together with the new piece of information. This set is comprised by the rules **Minimum Cost**, **Balanced**, **Current Orders**, **Shift Profitability**, **Geographical Closeness**, **Immediate Cost**, **Minimize Vehicles** and **Random**. We described each of them and we formalized the constraints they observe in the process of evaluating where the new demands could be located amongst the current vehicle routes. We also presented the implementation of **Minimum Cost** and we discussed its complexity. The details about the remaining dispatch rules are discussed in the Appendix. Table 3.11 summarizes the worst-case complexity results of the rule implementations in terms of the fleet size m, the length of the currently longest route $n_{\max}$ and the Indigo complexity constant I. We also weakened some of the conditions imposed by the heuristics in order to have a higher capability for comparing the rules.

Rule	Complexity
MIN	$\Theta(m * n_{\max}^2)$
BAL	$\Theta(m * n_{\max}^2)$
CUR	$\Theta(I * (m * n_{\max}))$
SHIFT	$\Theta(m * n_{\max}^2)$
GEO	$\Theta(m * n_{\max}^2)$
IMM	$\Theta(m * n_{\max}^2)$
MAV	$\Theta(m * n_{\max}^2)$
RAND	$\Theta(m * n_{\max}^2)$

Table 3.11: Worst-case complexity assesments.

Chapter 4

Heuristic Experiments

Experiments must confirm
the theory behind.
by Martin Aleksandrov

This chapter begins with a brief review of the addressed problem followed by the presentation of various statistical analyses we conducted in the process of dataset generation. We produced 330 datasets, whose goal is to serve as representative samples for past (100 datasets), evaluation (100 datasets), test (30 datasets) and seeming (100 datasets) data.

The following section details our experimental framework. Its goal is to capture a large number of various experimental settings. At the same time, one can simulate a fleet dislocation by incorporating seeming client requests into the experiments. We introduced the notion of *solution* and *partial solution* to a given routing problem within our framework and discuss some issues relevant to a scenario, in which *time slices* are used. The latter might be beneficial for the dispatchers in charge of dynamically handling the requests arriving every day at the company. We then define the notion of *heuristic correctness*, which allows not only to evaluate the dispatch rules addressed in the previous chapter, but also to direct the learning procedure.

The experimental scheme is then followed by three specific cases of it, i.e. *once-a-day*, *time-zones* and *fixed-time-span*. For each of these pooling strategies we investigate the statistics related to the heuristic correctness with respect to ten suboptimal solutions produced by Indigo solver. We also report additional descriptive statistics for the individual heuristic cost, time performance and the improvement gained by each rule using the prediction model.

4.1 General Assumptions

This section puts forward a short summary of the ODPDP instance we deal with by describing the main aspects of the problem source as well as the assumptions made during modeling the real case-study. The major request of Bonds Express Couriers is a decision-support tool, which help dispatchers in surmounting the burden composed by multiple daily customer demands. The tool partly is supposed to give recommendations about easy and efficient handling of the unreserved orders by taking into account a large number of the system parameters.

Locations We consider customer and depot locations, which are uniformly distributed within the metropolitan area of Syndey. All of them are independently generated.

Fleet The company has at their disposal heterogenous fleet of vehicles. Each vehicle has (1) maximal capacities, (2) a proper type, (3) its own depot, (4) running cost and (5) driver cost (see Table 1.1). Regarding the fleet we make the following assumptions : (1) each vehicle is assigned to exactly one route, (2) drivers cannot exchange packages, (3) vehicles are diverted only at client locations and (4) there are no warehouses on the map.

Demands Every working day there is a set of known demands, which are requested on previous days. As the day progresses multiple unreserved requests arrive at the system. Each demand, either pre-requested or dynamic, has an attached list of features : (1) arrival time, (2) number of pallets and weight, (3) time windows for the pickup and the delivery locations and (4) an associated service time, needed to process the corresponding demand.

A Problem definition

As a complex constraint satisfaction problem every ODPDP has for a main goal to optimize a particular objective function. In our problem we concentrate on minimizing the overall costs at the end of the day (see Section 2.5 for more details).

4.2 Benchmark Datasets

4.2.1 Original Dataset

Here, we uncover some details about the datasets used in our experiments. Initially, the company provided information about services performed within a single working day. However, this list is insufficient to design experiments based on which thereafter to draw any general conclusions about the overall fleet management and the quality of the services offered by the company. An insufficiency we had to deal with was the incomplete address information regarding the locations where customers request services. In order to overcome this we generated a large number of uniformly and independently distributed geographical locations within the considered area. Nevertheless, we needed to conduct a statistical study about the multivariate distribution of the remaining order features. In Figure 4.1 we show part of the original information provided by the company in the format we adopted.

order_id	pallets	weight	booked	epickup	lpickup	ldel	veh_type	call_type	locp_x	locp_y	locd_x	locd_y
6829586	0	5	555	555	625	610	CAR	$$	-33.7113	150.7806	-34.208	150.8047
6831652	1	400	608	730	830	955	ST	W3	-33.6842	151.458	-34.2952	151.4596
6831654	0	1800	622	700	900	1100	2T	W3	-34.1437	151.2852	-33.7611	151.5266
6831655	0	1	646	700	800	1000	CAR	W3	-33.6284	151.5335	-33.6294	150.9195
6831656	0	180	653	730	1200	1700	ST	W3	-33.9423	151.6548	-33.4538	150.7514
6829596	0	1	700	700	800	830	CAR	$$	-33.9246	151.9669	-33.4946	150.9287
6829597	0	0	700	700	800	1000	CAR	$$	-34.2915	150.9027	-33.5465	150.795

Figure 4.1: Datasets.

As one can see we have thirteen order features as follows : (1) request identifier, unique for each customer order, (2) number of pallets each order requires, (3) the weight of the ordered packages, (4) order arrival time, (5) earliest pickup time, (6) latest pickup time, (7) latest delivery time, (8) minimal vehicle type able to handle the particular order, (9) the type of booking, i.e. permanent, via-phone or via-WWW demand, (10-11) pickup coordinates and, finally, (12-13) delivery coordinates.

4.2.2 Arrival distributions

In Figure 4.2 we show a graph of the arrival distributions based on the data provided by the company. It contains 1413 orders in total, among which there are 65 predetermined, 1128 dynamic as well as 220 demands requested for the following days. As one may notice, between 5 a.m. and 13 p.m. we have a significantly large number of arrivals with their peak soon after 10 a.m. and a value of a little more than 200 orders per hour. Then the frequency tends to stabilize until 15 p.m., followed by a drastic decrease in the late afternoon till the evening (the end of the working day at 19 : 00).

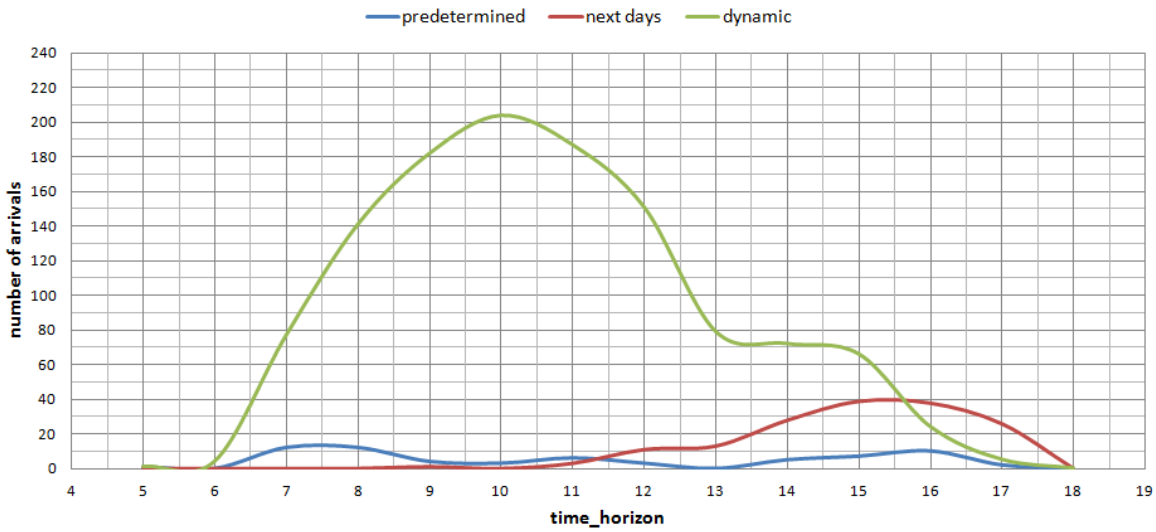


Figure 4.2: Order arrival distributions.

However, this is not the case with the next-days requests. There is a slow increase of the values around 11 a.m. and it continues to rise until it reaches its peak around 3 : 45 p.m. Thereafter the number of customer requests decreases smoothly until the dispatcher end-shift time. An interesting observation is that the curve representing the predetermined orders deviates around similar values along the whole day with two small exceptions. Slightly more orders are to be delivered in the morning, which is natural for these type of demands. For instance, if a customer has previously requested goods and his working process depends on this delivery he most probably would prefer somebody collecting or delivering it in the morning when the working day starts to serving it later along the day. The increase in the afternoon could be supported by the fact that when these demands have been requested the dispatchers in the company had not had previously the information about the dynamic arrivals within the day they are required for service. Thus, those clients have bigger freedom in selecting a convenient time window along the day than a customer, who makes a call the same day. Notice that the following experiments are based only in the presence of predetermined and dynamic data because it would be naturally to assume that the client demands for the following days are part of their statistics.

4.2.3 Regression analyses

The lack of enough available information motivated us to conduct a study of the order feature distributions and their relations. We use a trial version of *Statistica 10* software package[1] to design a variety of fitting distribution and correlation analyses before generating additional datasets. In the initial dataset we had 1193 dynamic plus predefined demands. A detailed study of the data allowed us to drop out 19 outliers which biased the regression curves. Then after a significant number of multiple regression analyses we observed several major regularities between the data features.

- The feature distributions along the day which can be approximated by distributions of random variables are (2), (3), (4), (5), (6), (7), (9), (10-13). It is normal to assume that the order identifier and the vehicle type are assigned after the information about the order has been revealed in the central dispatch unit.

- We split the set of features into four types, i.e. commodity, time, order type and geographical features. Each group of features is independent from the others. In particular, locations are independent among them. The commodity features and the time features are linearly correlated and, thus, we next study the functions relating them.

We refer the reader to [25] for more information about the definitions of the different statistical measures (mean, standard deviation, regression coefficient, etc.). In Figure 4.3 we summarize the results obtained in the linear regression analyses. Each entry in the table is the value of the standardized correlation coefficient between the corresponding features. It is a float number rounded until the second meaningful digit after the decimal point. Value of 0.00 signifies that the features are strongly linearly independent and a non-zero value for a regularity existance. In the former case one could argue about a possible non-linear

[1] `http://www.statsoft.com/`

connection between the respective features. However, this situation is refuted as well by the conducted multivariate analyses. Next, the red-color decimal attributes show a strong correlation dependency amongst the time features. In particular, the earliest pickup, the latest pickup and delivery times depend strongly on the arrival time. The latter result is natural as these preferences are known only after a particular client states an order. In addition, it is likely that the latest times depend more on the earliest time than on the booking time, which is confirmed by the higher values for these feature combinations. We give other piece of attention to the entries in green color, which stand for averagely positive linear dependency between the order weight and the number of pallets it requires. Note that the latter one is a discrete variable and, therefore, the value of the regression coefficient is hard to be interpreted as it is only defined for continuous stochastic samples. On the other hand this feature has enough distinctive values and, thus, we are pleased with the approximation discovered. The reader is referred to [48] for more information about the dependency between two arbitrary random variables. Furthermore, we designed residual analyses in order to improve the quality of the artificially generated datasets afterwards.

F1\F2	pallets	weight	booked	epickup	lpickup	ldel	ord_type	px	py	dx	dy
pallets	1.00	0.62	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
weight	0.62	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
booked	0.00	0.00	1.00	0.98	0.96	0.86	0.00	0.00	0.00	0.00	0.00
epickup	0.00	0.00	0.98	1.00	0.98	0.89	0.00	0.00	0.00	0.00	0.00
lpickup	0.00	0.00	0.96	0.98	1.00	0.95	0.00	0.00	0.00	0.00	0.00
ldel	0.00	0.00	0.86	0.89	0.95	1.00	0.00	0.00	0.00	0.00	0.00
ord_type	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00
px	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00
py	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00
dx	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00
dy	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00

Figure 4.3: Regression matrix.

In order to uncover the process of generating the datasets in more details we next present a general linear regression equation and discuss its components. By x and y we denote the respective predictor and dependant variables, by $\overline{a}$ and $\overline{b}$ the estimates of the real coefficients obtained during the analyses and the last term, ϵ, represents the residual in these analyses. Usually, for large samples the error follows a normal distribution. Moreover, the coefficients have a particular non-zero deviation, which approaching zero with higher values of the regression coefficient. We took all of these fine-grained observations when generating the order tuples for our experiments.

$$y = \overline{a} * x + \overline{b} + \epsilon$$

4.2.4 Predictor distributions

In addition to the functional dependency results established above, we investigate in more detail the distribution of the order arrival time and the order weight as being major sources implying those regularities. The number of orders revealed during the day can be approximated by a non-homogenous and stationary Poisson process as in Figure 4.4 on the left.

On the horizontal axis the working hours are placed along the day, while on the vertical one the cumulative number of arrivals for these periods. A nice consequence of this approximation is that the order interarrival time follows an exponential distribution (see [49] for more details regarding this issue). However, this is not the case with the order weight feature, shown in Figure 4.4 on the right, which follows very steep Gamma distribution. Sligthly more than 1000 orders are composed of less than a hundred kilograms each and all the rest are distributed quite sparsely until a maximal value of 2000 kilograms. Recall, we removed some of the observations, viz. so-called outliers, as they strongly biased the results obtained regarding the regression between the weight and the pallets.

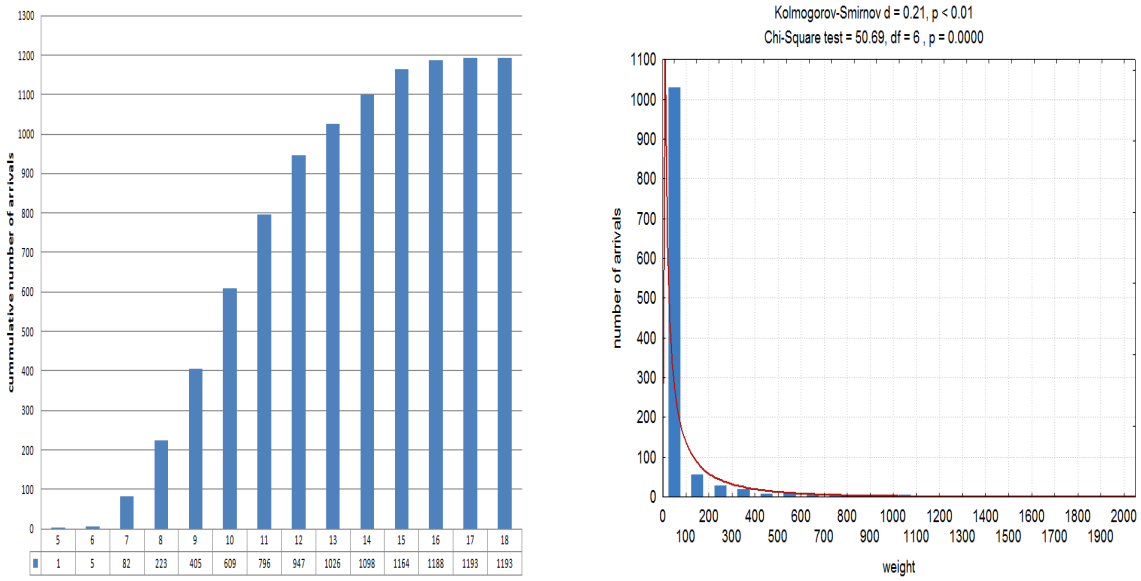


Figure 4.4: Arrivals and weight distributions.

4.2.5 Generated datasets

Here, we pinpoint some details about the creation of the additional datasets. Using the previously measured statistical results we created 330 datasets of orders, denoted as *day1-day230* and *pday100-pday200*, which we describe next.

1. The files *day101-day200* are used to test the performance of the dispatch rules being implemented. From now on we call every tuple within each of these datafiles *a real order*.

2. The purpose of *pday101-pday200* is to represent the stochastic model of the environment for the datasets *day101-day200*. We refer to the individuals of this collection of demands as *seeming or "fake" orders*. In the process of creating these orders we also take into account the descriptive statistics of the customer requests in *day100-day200* as follows : *pday101* is generated based on the statistics obtained for the days *day1-day100*, *pday102* on the distributional features for days *day1-day101*, and so on,

until the last dataset *pday200*, which is created using the descriptive numbers for days *day1-day199*. In reality, the increase of the revealed information in the past days with respect to the current date could help for a better prediction of the future demands.

3. The last thirty datasets, i.e. *day201-230* will serve to assess the quality of the policies learnt, as discussed in the experimental section of the next chapter.

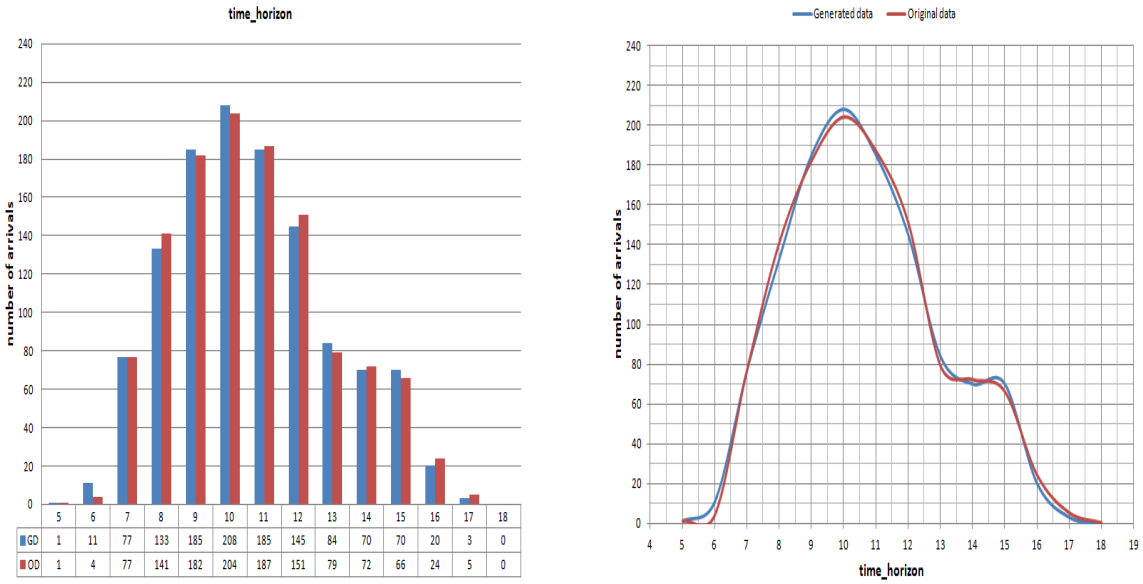


Figure 4.5: Comparison of the original and the generated datasets.

In Figure 4.5 we compare the arrival distributions of the original information and a dataset obtained after we averaged the arrivals over the datafiles *day101-day200*. The histogram on the left shows how much the two samples differ within each work hour, whereas the graph on the right gives a flavour about the similarity of their smooth distribution curves. We also report several descriptive statistics of the daily arrivals for all types of orders in Table 4.1. The average number of real orders per dataset is 1192, which varies within the limits 1074 and 1260. Between 12%-18% of them have been requested on previous days. Besides these numbers we have an average of 1183 "fake" orders per day. We do not generate pre-requested seeming services because all of these are known at simulation time. The columns marked by a notation with index d are the statistics for the dynamic demands, whereas those marked with p are the corresponding numbers for the known orders.

order type	m	σ	min	max	m_p	σ_p	$\min_p$	$\max_p$	m_d	σ_d	$\min_d$	$\max_d$
real orders	1192	34	1074	1260	180	6	158	193	1012	29	907	1067
"fake" orders	1183	45	988	1274	1183	45	988	1274	0	0	0	0

Table 4.1: Real and seeming order statistics.

4.3 An Experimental Framework

A decision-support system in any area of the public or private sector should be carefully designed and must guarantee its efectiveness by giving qualified recommendations, which rely on the overall current status. Usually such a system maintains a large number of statistics, which is a difficult task for the employees. On the other hand, the decisions made should be clear and easily understandable by the end-users. They also must not be misleading or ambiguous as this would lead to mistakes when a final decision is to be made. The latter is of significant importance when a human life depends on such a supportedness. Although, this is not the case in our work, we are interested in verifying the quality of the decisions made by the simple rules we implemented. In order to do so we first define a few appropriate notions and notations, which we believe will help the reader to avoid a possible confusion later on. We also describe three pooling strategies under both scenarios, i.e. with and without the use of stochastic model.

4.3.1 Initial schedule

Let us denote all the orders executed in a particular day by $O = \{o_1, \ldots, o_n\}$. Part of these demands have been requested on some previous days, and therefore they are already known in the morning when the dispatchers have started their shift. We denote these demands by $I = \{i_1, \ldots, i_p\}$. At the beginning of the time horizon we produce a pickup and delivery routing and scheduling plan (PDRSP) P_I for the set I. Also at the end of the working day we create similar timetable for O, i.e. P_O. The positions of the I-orders differ in P_I and in P_O. However, in the latter plan they are the desired ones as these are decided taking into account all orders. Therefore, we are interested in a maximal matching between the positions of the I-orders in both plans. However, in the morning of the working day we have available only the set I, but not O. Nevertheless, we could predict how many orders would arrive the same day using the past data statistics and in this way to improve the quality (with respect to the final plan) of the initial plan as follows : (1) generate a set of additional orders F, simulating the dynamic ones, (2) create a PDRSP $P_{I \cup F}$ for F and I, and (3) project the set I in $P_{I \cup F}$.

4.3.2 Suboptimal schedule

During the heuristic evalution experiments we are concerned that each individual decision about a new order does not depend on possible wrong previous decisions. On the other hand, at the end of the day we have at our disposal all the decisions for the past shift as if we knew them initially. Hence, for a particular dynamic time instant along the day we could compare the dynamic heuristic decisions made regarding the new demands with those available at shift completion when an offline schedule is produced. The latter is done by using the Indigo solver to create a schedule given a problem instance. It does so though a combination of heuristics which, given a problem specification, initially builds the vehicle assignments and then improves them using local search methodologies. At the end of this process we have a visiting timetable for managing the fleet services. We call such a schedule *suboptimal plan.*

4.3.3 Evaluation procedure

Let us think of a solution to a given ODPDP as a set of requests. We denote a suboptimal plan for the whole set O of n orders as $S(O, n)$. We call the positions of the order requests in this timetable *suboptimal positions* as they have been assigned under the assumption that the whole information is known in advance. Furthermore, let $h \in H$ be an heuristic from our repository. We would like to evaluate it in such a way that each individual decision made about a new customer is independent from the other past, current and future heuristic decisions. A possibility to do that is the following : for $o_k \in O \setminus I$ we fix the orders in the suboptimal plan $S(O, n)$, which have arrived before the booking time t_k of o_k. We call this subset of requests *partial schedule* or *partial plan* and we denote it by $S(O, n_k, t_k)$, where $n_k \leq n$ is the number of orders arrived before t_k. Then h makes a decision about where o_k can be inserted and we compare that decision with the location of the same order in $S(O, n)$. In this way, the decision made does not depend on possible wrong previous decisions as the positions of the requests in $S(O, n_k, t_k)$ are suboptimal. We must say that the decisions we are dealing with are within the scope of vehicles. In other words, each decision simply concerns the vehicle that will serve the new loads. A more precise view would be to consider the precise positions along a vehicle route where the new requirements are positioned, but this approach would concentrate more on the local route features than the global ones, which are amongst our objectives. In this context, we consider that an heuristic decision is *correct* if it corresponds to the vehicle that manages the particular order according to $S(O, n)$. After making this comparison against the offline schedule $S(O, n)$ we update the partial plan $S(O, n_k, t_k)$ by including the order o_k in it.

In Figure 4.6 we depict the major procedure in our experimental setting. We start by fixing the known information, which is represented by the set I with p orders. The mapping $f : \mathcal{S} \times N \to \mathcal{S}$ represents this operation. It takes a solution S, a natural number n_0 ($n_0 \leq n$) and it returns a partial schedule for those n_0 orders. Furthermore, without loss of generality we can assume that the dynamic orders $O \setminus I = \{o_1, \ldots, o_{n-p}\}$ arrive in succession (i.e. o_1, o_2 and so on until the last one o_{n-p}) and their arrival times are t_1, t_2, ..., t_{n-p}, respectively.

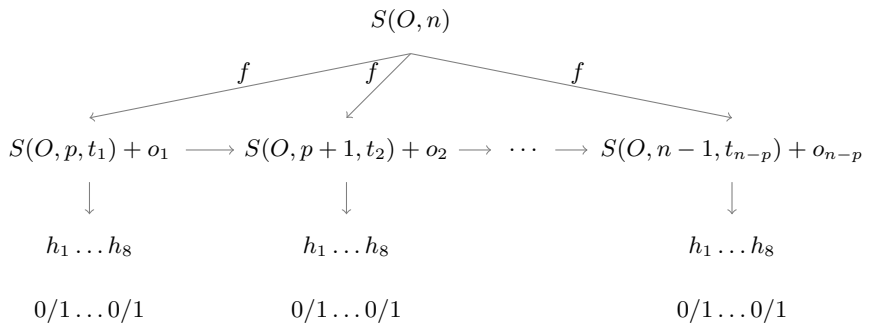


Figure 4.6: Evaluation procedure.

We have implemented eight dispatch rules and, hence, for each order $o_k \in O \setminus I$ every rule should make a decision. The latter is ranked as correct (1) or incorrect (0). Then the same procedure is activated for $n_0+1, n_0+2, \ldots, n-1$ orders, respectively, until all the dynamic orders have been decided.

4.3.4 Request sequences

We now address request sequences and what we call a solution to a given ODPDP in terms of decisions. Recall, every working day in the company constitutes of finite knowledge about customer demands and, therefore, we are interested in finite request and decision sequences.

Definition 13 : (**Request sequence**) Let $R_O = \{r_1, \ldots, r_{2*n}\}$ be the set of requests for a given set of orders $O = \{o_1, \ldots, o_n\}$. Then we call $R^n = [r_1, \ldots, r_{2*n}]$ the *request sequence* corresponding to R_O. It cointains the requests from R_O sorted by their arrival time.

Definition 14 : (**Heuristic scheme**) Let H be a repository of heuristics. Let also $R^n = [r_1, r_2, \ldots, r_{2*n}]$ be a finite sequence of requests arriving in the system at the moments t_1, t_2,..., t_{2*n}, respectively. Then a *finite heuristic scheme* to R^n is the decision sequence $d^n = [d_1, d_2, \ldots, d_n]$, where $d_k \in H$ for $k \in [1, n]$ is the decision made at time instant t_k for managing the new order $o_k \in O$.

Note that given a finite heuristic scheme there are $\mid H \mid^{2*n}$ instantiations of it. Thus we define the following notion of heuristic schedule.

Definition 15 : (**Heuristic schedule**) Let H be a repository of heuristics. Let also $d^n = [d_1, d_2, \ldots, d_n]$ is a finite heuristic scheme to the request sequence $R^n = [r_1, r_2, \ldots, r_{2*n}]$. Then a *finite heuristic schedule* to R^n is the heuristic sequence $h^n = [h_1, h_2, \ldots, h_n]$, where h_k for $k \in [1, n]$ is a concrete heuristic from H (i.e. substitute d_k for $k \in [1, n]$ in d^n by h_k).

4.3.5 Solutions and partial solutions

The Indigo solver may produce a schedule when an input request sequence is given. Such a schedule not only assigns a finite heuristic solution to that sequence, but also contains additional information about the fleet timing at the client locations and details about vehicle rack-space availability. We capture this information with the following definitions.

Definition 16 : (**Fleet schedule**) Let $R^n = [r_1, \ldots, r_{2*n}]$ be a request sequence and H be a repository of heuristics. Then we call the triple $S^n = (h^n, \mathcal{R}^n, \mathcal{I}^n)$ *a fleet schedule* (*solution*) to R^n, where h^n is a finite heuristic schedule to R^n and the next two components constitute a PDRSP $P^n = (\mathcal{R}^n, \mathcal{I}^n)$ for R^n.

Recall, $\mathcal{R}^n$ contains the ordered vehicle request routes and $\mathcal{I}^n$ their itineraries. Each such timetable contains all the details about the fleet management for handling R^n and we call it *schedule itinerary* to R^n. Each unitary element r_k in $\mathcal{R}^n$ is a request and i_k in $\mathcal{I}^n$ is a request itinerary. They are formalized as follows :

$$r_k : < t_k,\, q_k,\, u_k,\, [t_e^k, t_l^k],\, l_k >$$
$$i_k : < a_k,\, w_k,\, b_k,\, s_k,\, e_k >,$$

where q_k, u_k, $[t_e^k, t_l^k]$ and l_k in the first tuple are, respectively, the pallets, the weight, the time window and the location of r_k. The second vector contains the features a_k, w_k, b_k, s_k, e_k, which are the arrival, waiting, starting, service and leaving times at the location l_k, respectively. Note that for each vehicle $v_j \in V$, the route $R_j^n = \{r_{j_1}, \ldots, r_{j_{n_j}}\} \in \mathcal{R}^n$ and its itinerary $I_j^n = \{i_{j_1}, \ldots, i_{j_{n_j}}\} \in \mathcal{I}^n$ are ordered sets. This observation allow us to unify them with sequences of, respectively, requests $[r_{j_1}, \ldots, r_{j_{n_j}}]$ and unitary itineraries $[i_{j_1}, \ldots, i_{j_{n_j}}]$.

Thus, the solution to a request sequence is a pair of heuristic schedule together with a pickup and delivery routing and scheduling plan for managing this sequence. The main idea behind the above definition is that each request sequence represents a routing problem and a solution to this instance contains two components. The first component represents a vehicle allocation of the orders in the particular plan, whereas the second one expresses the fleet execution plan itself.

Recall that $V = \{v_1, \ldots, v_m\}$ is the set of vehicles and that each member v_j of this set can be assigned to exactly one visiting sequence. Also, recall, that any two vehicles $v_{j_1}, v_{j_2} \in V$ cannot have common assignments, i.e. they cannot process the same requests at any time point. Hence, one may be interested in the solution for a specific vehicle or even a vehicle type, which is expressed by the following definition.

Definition 17 : (**Vehicle schedule**) Let $v_j \in V$ be a vehicle, which according to the solution $S^n = (h^n, P^n)$ to R^n processes the request sequence $R_j^n = [r_{j_1}, r_{j_2}, \ldots, r_{j_r}]$ with $j_1 < j_2 < \ldots < j_r$. Then we call the heuristic schedule $h_j^n = [h_1^*, \ldots, h_n^*]$ a *projection* of h^n onto the route of v_j. Each element h_k^* is defined as follows : (1) it is h_k if $k \in \{j_1, \ldots, j_r\}$ and (2) ϵ (no decision is made about r_k) otherwise. Then a *schedule* of vehicle v_j is the triple $S_j^n = (h_j^n, R_j^n, I_j^n)$.

As we mentioned at the beginning of Section 4.3.3, given a solution to a routing problem we are interested in the projection of that schedule over some of the orders. Thus we next define the notion of partial schedule or partial solution to a given request sequence.

Definition 18 : (**Partial fleet schedule**) Let $S^n = (h^n, \mathcal{R}^n, \mathcal{I}^n)$ be a solution to $R^n = [r_1, \ldots, r_{2*n}]$. Then the triple $S^{n,p} = (h^{n,p}, \mathcal{R}^{n,p}, \mathcal{I}^{n,p})$ is called *partial fleet schedule* or *a partial solution* to R^n over the first $p \in [1, n]$ orders. It is defined as follows : (1) $h^{n,p} = [h_1, \ldots, h_p, \epsilon, \ldots, \epsilon]$, where for each $k \in [1, p]$ we have that $h_k \in h^n$ and (2) $\mathcal{R}^{n,p}$ and $\mathcal{I}^{n,p}$ are, respectively, the projections of $\mathcal{R}^n$ and $\mathcal{I}^n$ over the first p requests.

Note that $S^{n,p} = (h^{n,p}, P^{n,p})$ is also a solution to the input sequence $R^p = [r_1, \ldots, r_p]$. However, it it not suboptimal solution to that request sequence. Rather than, it just contains information about the requests arrived by the time t_{p+1} when o_{p+1} has occured. In the terms we used in Section 4.3.3 each position of a request in $S^{n,p}$ is called suboptimal and $S^{n,p} = S(O, p, t_{p+1})$.

In order to avoid further confusions when we work with both types of solutions we will refer to the firstly defined notion as a *complete solution* to a given request sequence. If no such ambiguity exists, we will keep refering it as just a solution to that sequence. In what follows, we define the notion of heuristic correctness, which serves as an evaluation measure of the dispatch rules from our repository.

4.3.6 Heuristic correctness

Here, we suggest several measures to evaluate the "distance" of a local heuristic decision with respect to an offline local and global suboptimal solutions. Generally, we have an order $o_k \in O$ arrived in the moment t_k, a suboptimal plan $S(O, n)$ and a projection $S(O, n_k, t_k)$ of it over the n_k orders known before t_k. Moreover, let H be the repository of the dispatch rules and $H^{Indigo} = \{h^{Indigo}\}$ be the heuristic, whose result is obtained by the combined efforts of the solver to assign a new order.

Definition 19 : (**Precise vehicle measure**) Let $V = \{v_1, \ldots, v_m\}$ be the considered fleet. Then we call a *precise valuation function* the mapping $f_1 : V \times V \to \{0, 1\}$, defined as follows :

- $f_1(v_j, v_k) = 1$, if $j = k$
- $f_1(v_j, v_k) = 0$, otherwise.

This measures assigns 1 only if the two arguments coincide and 0, otherwise. Although, the value can be zero, it might be the case that exists a suboptimal plan, which is somehow symmetric to the produced one (i.e. $S(O, n)$) and where the new demand is actually assigned a vehicle of the same type. Thus, we proceed by defining a vehicle type measure in order to break this kind of symmetry with respect to the type characteristic of the fleet. Note that in the latter case we capture a large set of suboptimal solutions.

Definition 20 : (**Vehicle type measure**) Let $V = \{v_1, \ldots, v_m\}$ be the considered fleet. Then we call a *type valuation function* the mapping $f_2 : V \times V \to \{0, 1\}$, defined as follows :

- $f_2(v_j, v_k) = 1$, if v_j and v_k are of the same type
- $f_2(v_j, v_k) = 0$, otherwise.

The two measures above capture a large number of more or less identical suboptimal solutions. Apart from them one could also argue that exist a suboptimal solution where the new order is handled by a vehicle, which has different, but still feasible type. The latter observation could contribute to determine whether a particular rule has made a good decision or not. Since in our fleet there is a large number of vehicles able to accommodate any new demand we only consider those passing close to both the target and the destination locations of the new order. Therefore, we define a third measure to capture this kind of solution similarity.

Definition 21 : (**Geographical type measure**) Let $V = \{v_1, \ldots, v_m\}$ be the considered fleet and $o_i \in O$ be an order. Then we call a *geographical valuation function* the mapping $f_3 : V \times V \to \{0, 1\}$, defined as follows :

- $f_3(v_j, v_k) = 1$, if v_j and a vehicle, geographically close to v_k with respect to o_i are of the same type
- $f_3(v_j, v_k) = 0$, otherwise.

All the measures f_1, f_2 and f_3 give valuations of the dispatch rules from H with respect to a suboptimal plan produced at later moment within the working day. Next, using the notion of valuation function we define the correctness of an heuristic decision.

Definitions 22 : (**Heuristic correctness**) Let $h \in H$ be a dispatch rule, $V = \{v_1, \ldots, v_m\}$ be the fleet and $f : V \times V \to \{0, 1\}$ be a valuation function. Also let in moment t_k a new order $o_k \in O$ arrives and $S(O, n_k, t_k)$ be the partial plan for the orders known before t_k. Then, if $v_k = h(S(O, n_k, t_k), o_k)$, $v^{Indigo} = h^{Indigo}(S(O, n), o_k)$ and $f(v_k, v^{Indigo}) = 1$, we call that h is *correct* with respect to $S(O, n)$, o_k and the Indigo solver. Otherwise, we call that decision *incorrect*.

4.4 Experimental Results

In this section we consider three pooling strategies, i.e. *once-a-day*, *time-zones*, *fixed-time-span*. For each of them we consider two scenarios, i.e. with and without simulating seeming needs. We evaluate the heuristics performance within each scenario for every strategy. Recall that *time-zones* and *fixed-time-span* strategies are represented by multiple time slices. The procedure depicted in Figure 4.6 assumes the availability of one suboptimal plan. However, one can argue that comparing against this plan is insufficient for a meaningful heuristic ranking. Thus, at each time slice boundary in every setting we produced 5 suboptimal plans, against which we evaluate the correctness of each rule. The latter give us the ability to rate the dispatch rules regarding the different valuation measures (precise, vehicle type and geographical) not only by assigning a boolean value to each of them, but a number ranging between 0 and 5. In addition, we report a set of descriptive statistics regarding the average individual heuristic cost per order, average heuristic cost per time slice, etc.. All these numbers allow us to decide which pooling strategy, among the presented, probably will give the best performance in practice.

We present some notation we are going to use in order to give a more formal view in the following sections. Let O be the set of n orders and T be the working time horizon. Then for a time slice boundary (time instant along the day) $t_i \in T$ we denote by $O^n_{\leq i}$ the set of orders arrived before t_i, by O^n_i the set of orders arrived within the i-th time slice and by $O^n_{>i}$ the set of orders after t_i. As in the different cases we need to distinguish the real and the unreal ("fake") demands we adopt the following notation for them : (1) by R we denote the set of n_R real demands and (2) by D the set of n_D seeming orders.

4.4.1 Once-a-day strategy

This scenario presents a scheme in which we reoptimize once during the working day, viz. at the end of the shift. That is, $T = [t_0, t_1] = [5{:}00,\ 19{:}00]$ or the entire dispacher shift, between 5:00 and 19:00, is one time slice. In addition, the only information available before 5:00 are the predetermined customer demands $I = \{i_1, \ldots, i_p\}$ and we assume that the new demands after 19:00 are collected for the subsequent days. Note that this does not mean that no orders are served after this time. It might be the case that a vehicle has collected a package and the delivery will be realized in an after-shift time. Also note that this pooling strategy is not likely to be executed in practice due to the high dynamicity of the problem we deal with and the large number of the arriving client demands. It is possible that during the day the accumulated number of wrong heuristic decisions is sufficiently large and lead to a bigger and bigger deviation from the desired final values. Although, this fact, we are interested in calculating an heuristic performance baseline, to which we can compare the rest of the pooling scenarios and, thus, *once-a-day* is the perfect candidate to do that.

Without Probabilistic Model In this case we consider only the presence of real customer demands. Then the setting can be expressed as follows :

$$(S(R, p, t_0), R \setminus I, \emptyset),\ (S(R, n_R), \emptyset, \emptyset),$$

where p is the number of real orders before t_0 (i.e. 5:00). Note that the number of triple correspond to the number of time slice boundaries. The first component is a suboptimal or partial plan, the second one is the set of orders arrived within the considered time slice and the third one is aimed to represent the seeming orders when incorporated. Thus, the dispatch rules are evaluated only within one interval (i.e. $[t_0, t_1]$).

With Probabilistic Model Here we consider the case when some seeming orders are simulated and, hence, the scheme is the following :

$$(S(R \cup D, p + n_D, t_0), R \setminus I, D),\ (S(R \cup D, n_R + n_D), \emptyset, \emptyset)$$

This time the number in $R \cup D$ we fix the initial p real demands and all the unreal ones. As opposed to the previous case the third component of the first triple is non-empty. That is, it contains the set of all the seeming requirements arrived after t_0. Also one can notice that the first components differ. The reason for that is due to the fact that a suboptimal schedule is produced for both the real and unreal requirements, whereas in the previous setting we consider only the set of real orders.

4.4.2 Time-zones strategy

As we mentioned above *once-a-day* scenario could serve as a good baseline to which one can compare other realizations of the time slices ideology. Another benefit of it could be in case we deal with a problem which is characterized by a lower dynamicity of the arriving customer requests than ours. In this case the wrong dispatch decisions would be less and it is more likely to achieve a particular objective. Nevertheless, this is not the case with the problem we deal with and, thus, we consider daily time zones, which split the

working horizon. At each slice boundary a new suboptimal plan is produced using all the currently revealed orders. In addition, in case of using the prediction model we also add some seeming clients to simulate fleet busyness. Next, we proceed by describing the triplet sequences (reoptimization points) more formally. The time horizon this time is split into five slices as follows : $T = [t_0, t_1, t_2, t_3, t_4, t_5] =$ [5:00, 9:00, 11:00, 13:00, 15:00, 19:00].

Without Probabilistic Model This scenario observes only the real client demands. Thus,

$$(S(R^{n_R}_{\leq 1}, n^0_R, t_0), R^{n_R}_1, \emptyset),\ (S(R^{n_R}_{\leq 2}, n^1_R, t_1), R^{n_R}_2, \emptyset),\ (S(R^{n_R}_{\leq 3}, n^2_R, t_2), R^{n_R}_3, \emptyset),$$
$$(S(R^{n_R}_{\leq 4}, n^3_R, t_3), R^{n_R}_4, \emptyset),\ (S(R^{n_R}_{\leq 5}, n^4_R, t_4), R^{n_R}_5, \emptyset),\ (S(R, n_R), \emptyset, \emptyset),$$

where n^i_R (for $i \in [0, 4]$) is the number of real orders arrived by the time instant $t_i \in T$. Unlike the previous pooling strategy when we had only one heuristic evaluation interval, here we have five of them. The latter allows us to compare the rules not only in terms of global values (according to the suboptimal schedule $S(R, n_R)$ produced at the end of the day), but also within the different parts of the working day (with respect to the local minumums of the form $S(R^{n_R}_{\leq i+1}, n^i_R, t_i)$ or the suboptimal plan for all the real demands before $t_{i+1} \in T$, in which n^i_R of them are fixed, namely those arrived before $t_i \in T$).

With Probabilistic Model This time the request sequences corresponding to the sets R and D are non-empty and, thus, the model has most of its stochastic components meaningful. In this way between each pair of consecutive time-horizon boundaries the decisions made by the dispatch rules would be supported by the presence of the seeming clients. The model looks like this :

$$(S(R^{n_R}_{\leq 1} \cup D^{n_D}_{>0}, n^0_R + n^0_D, t_0), R^{n_R}_1, D^{n_D}_{>0}),\ (S(R^{n_R}_{\leq 2} \cup D^{n_D}_{>1}, n^1_R + n^1_D, t_1), R^{n_R}_2, D^{n_D}_{>1}),$$
$$(S(R^{n_R}_{\leq 3} \cup D^{n_D}_{>2}, n^2_R + n^2_D, t_2), R^{n_R}_3, D^{n_D}_{>2}),\ (S(R^{n_R}_{\leq 4} \cup D^{n_D}_{>3}, n^3_R + n^3_D, t_3), R^{n_R}_4, D^{n_D}_{>3}),$$
$$(S(R^{n_R}_{\leq 5} \cup D^{n_D}_{>4}, n^4_R + n^4_D, t_4), R^{n_R}_5, D^{n_D}_{>4}),\ (S(R \cup D, n_R + n_D), \emptyset, \emptyset),$$

where each triplet $(S(R^{n_R}_{\leq i+1} \cup D^{n_D}_{>i}, n^i_R + n^i_D, t_i), R^{n_R}_{i+1}, D^{n_D}_{>i})$ in the above sequence can be interpreted as follows : (1) $S(R^{n_R}_{\leq i+1} \cup D^{n_D}_{>i}, n^i_R + n^i_D, t_i)$ is the projection of the suboptimal plan $S(R^{n_R}_{\leq i+1} \cup D^{n_D}_{>i}, n^{i+1}_R + n^i_D)$ over the first n^i_R real demands and the last n^i_D unreal ones, (2) $R^{n_R}_{i+1}$ is the set of real client orders arrived in $[t_i, t_{i+1}] \in T$ and (3) $D^{n_D}_{>i}$ is the set of n^i_D "fake" orders from $t_i \in T$ onwards. Note that along the time horizon T the set of unreal orders is getting smaller, which is natural consequence of the fact that the real information is getting bigger. In other words, as the number of real orders increases, the need of the stochastic model decreases.

4.4.3 Fixed-time-span strategy

Inspired by the aspect of uniform dispatcher shift, here, we consider a division of the time horizon where the slices have equal continuance. Note that the previous scenario is based on the arrival frequency of the customer demands. However, in reality each day is a specific one and there might be such a shift that reoptimizations based on time-zones scenario is

not feasible enough for that case. Therefore, it may be better for the workforce to know in advance when such a global update will be performed without the knowledge of any future expectations about any customer orders. Thus, we implement the *fixed-time-span* pooling strategy capturing these issues. The time horizon is $T = [t_0, t_1, t_2, t_3, t_4, t_5, t_6, t_7]$ = [5:00, 7:00, 9:00, 11:00, 13:00, 15:00, 17:00, 19:00] and the reoptimization conditions are similar to the ones described above.

Without Probabilistic Model This case is when there are no seeming orders. Thus, the dispatch rules make their decisions based on the currently revealed real information. The setting is the following :

$$(S(R^{n_R}_{\leq 1}, n^0_R, t_0), R^{n_R}_1, \emptyset),\ (S(R^{n_R}_{\leq 2}, n^1_R, t_1), R^{n_R}_2, \emptyset),\ (S(R^{n_R}_{\leq 3}, n^2_R, t_2), R^{n_R}_3, \emptyset),$$
$$(S(R^{n_R}_{\leq 4}, n^3_R, t_3), R^{n_R}_4, \emptyset),\ (S(R^{n_R}_{\leq 5}, n^4_R, t_4), R^{n_R}_5, \emptyset),\ (S(R^{n_R}_{\leq 6}, n^5_R, t_5), R^{n_R}_6, \emptyset),$$
$$(S(R^{n_R}_{\leq 7}, n^6_R, t_6), R^{n_R}_7, \emptyset),\ (S(R, n_R), \emptyset, \emptyset),$$

where we used the same notation as in the previous cases. Compared to them, this setting offers more heuristic intervals within which to evaluate the rules and, consequently, we are able to trace the decision-making process more precisely.

With Probabilistic Model When "fake" orders are assumed we have a situation, in which the heuristics receive additional supportiveness when making decisions. In such a scenario the sequence of triplets, which express the heuristic evaluation intervals is as follows :

$$(S(R^{n_R}_{\leq 1} \cup D^{n_R}_{>0}, n^0_R + n^0_D, t_0), R^{n_R}_1, D^{n_D}_{>0}),\ (S(R^{n_R}_{\leq 2} \cup D^{n_R}_{>1}, n^1_R + n^1_D, t_1), R^{n_R}_2, D^{n_D}_{>1}),$$
$$(S(R^{n_R}_{\leq 3} \cup D^{n_R}_{>2}, n^2_R + n^2_D, t_2), R^{n_R}_3, D^{n_D}_{>2}),\ (S(R^{n_R}_{\leq 4} \cup D^{n_R}_{>3}, n^3_R + n^3_D, t_3), R^{n_R}_4, D^{n_D}_{>3}),$$
$$(S(R^{n_R}_{\leq 5} \cup D^{n_R}_{>4}, n^4_R + n^4_D, t_4), R^{n_R}_5, D^{n_D}_{>4}),\ (S(R^{n_R}_{\leq 6} \cup D^{n_R}_{>5}, n^5_R + n^5_D, t_5), R^{n_R}_6, D^{n_D}_{>5}),$$
$$(S(R^{n_R}_{\leq 7} \cup D^{n_R}_{>6}, n^6_R + n^6_D, t_6), R^{n_R}_7, D^{n_D}_{>6}),\ (S(R, n_R), \emptyset, \emptyset),$$

4.4.4 Cost

In this section we report several descriptive statistics of the average individual heuristic cost over the collection of datasets we generated. However, we discuss only the case of *once* strategy and leave the statistics for the remaining scenarios in the Appendix. Recall, we evaluate the heuristics over the datasets *day101-day200* where the number of all orders is more than 100 000 (one hundred thousand). We summarize the statistics in Table 4.2. These are the estimate of the individual cost mean c, the standard deviation σ_c of this cost and its 95%− as well as 99%− confidence intervals. As one can notice MAV rule performed best with respect to the average cost in both scenarios, i.e. with ($3.90) and without ($3.84) the use of seeming clients. The rationale behind this is partly because it tries to assign every new demand to an actuated vehicle. On the other hand, every decision about employing a new driver leads to an additional expense (driver wage). The opposite limit of the average cost interval has been achieved by BAL heuristic with 16.68 dollars when there are no "fake" orders and 19.23 in case of seeming presence. The latter effect can be supported by the fact that the rule tries to uniform the total workload among the entire fleet.

	Without Model				With Model			
	5-19:00				5-19:00			
h	c [**$**]	σ_c	95%−	99%−	c [**$**]	σ_c	95%−	99%−
MIN	3.99	4.43	[3.96,4.02]	[3.95,4.03]	4.30	5.93	[4.26,4.34]	[4.25,4.35]
BAL	16.68	7.23	[16.66,16.71]	[16.65,16.72]	19.23	11.29	[19.19,19.26]	[19.18,19.28]
$SHIFT$	9.45	4.65	[9.43,9.48]	[9.42,9.49]	9.89	6.13	[9.86,9.93]	[9.85,9.94]
CUR	7.07	4.22	[7.04,7.10]	[7.03,7.11]	8.10	4.34	[8.06,8.14]	[8.05,8.15]
GEO	13.04	8.76	[13.02,13.07]	[13.01,13.08]	13.91	9.36	[13.87,13.94]	[13.86,13.95]
IMM	9.42	4.95	[9.40,9.45]	[9.39,9.46]	10.41	6.02	[10.37,10.44]	[10.36,10.45]
MAV	3.84	3.84	[3.81,3.87]	[3.80,3.88]	3.90	3.90	[3.86,3.93]	[3.85,3.95]
$RAND$	10.38	5.69	[10.35,10.41]	[10.34,10.42]	11.38	7.33	[11.35,11.42]	[11.33,11.43]

Table 4.2: Once-a-day pooling strategy : cost statistics.

Thus, it is quite likely a new vehicle to be actuated, which leads to undesired costs for the driver's wage. Due to the large number of individuals in our experiments the length of the confidence intervals vary only between 5 and 10 AUS cents. Therefore, we can be quite confident about the average individual heuristic cost. Even we can make an estimate of the overall final cost using the means in Table 4.2 as follows : (1) calculate the individual cost averaged over the heuristics, i.e. 9.23 and 10.14 for the cases of without and with the seeming model, respectively, and (2) if we expect 1000 orders within a given day, then the expected cost for serving them should vary between 9230 and 10140 dollars. Note that the last are just benchmarks assuming that each rule has been applied an equal number of times during the day. Another interesting observation is about the cost increase in case of having simulated demands. The reason for that is due to the higher number of orders. For instance, a vehicle elapses the distance between two real demands, according to the suboptimal plan for reals, cheaper than according to the suboptimal plan for the real and the unreal requirements. The idea is that between two consecutive real errands in the latter plan there can be multiple "fake" orders and, thus, the travel costs are higher.

4.4.5 Time performance

In what follows we briefly discuss the time performance of the implemented dispatch rules. In Table 4.3 we report the numbers (in miliseconds) obtained within the *once*-setting. They are averaged over all the evaluation data. In the case when we do not benefit from the stochastic model, the rules perform significantly faster. The reason is partly because we have shorter vehicle routes as well as smaller number of active vehicles than in a case with unreal demands. However, in both cases the CUR rule needed quite a lot of time, which is expected since every time an order arrives at the system this heuristic decides not only where to assign it, but also how to rearrange the remaining demands. Among the rest of the rules we have that IMM is at the first place when there are no "fake" orders, which can be explained due to the attempts it makes to assign every new order as soon as possible.

However, in the situation with seeming errands IMM is outperformed by $SHIFT$ heuristic, possibly due to the decreased overall waiting time at customer locations. In the latter case there are less possibilities for the new demand to be inserted and, thus, the CPU time needed by $SHIFT$ rule decreases. On the other "edge" of the time scale (except CUR rule) stands MIN dispatch heuristic with maximum values of 33.29 miliseconds in the first line and 116.44 miliseconds in the second one.

model	MIN	BAL	$SHIFT$	CUR	GEO	IMM	MAV	$RAND$
no	33.29	27.69	10.73	$\geq 2*10^4$	12.52	9.15	27.97	18.27
yes	116.44	100.85	15.33	$\geq 3*10^4$	27.8	15.57	91.93	54.31

Table 4.3: Once-a-day pooling strategy : time performance.

As one can notice the rules give recommendations in a reasonable time with respect to the time the particular demand is requested (the time the calling lasts). We believe that one can conclude similar results in *time-zones*- and *fixed-time-span*-settings and, therefore, we do not present these.

4.4.6 Correctness

The notion of heuristic correctness is schedule-dependent. That is, a given dispatch rule will perform differently with respect to distinct suboptimal plans. Thus, for each time slice boundary in each scenario (i.e. with and without the seeming model) and a pooling strategy (*once*, *time-zones*, *fixed-time-span*) we produced 5 suboptimal plans. Besides this issue for each valuation function (precise f_1, vehicle f_2 and geographical f_3 types) we investigate two kinds of comparisons. The first one is with respect to the local incomplete oracles, positioned at the time slice boundaries. They know the 5 plans at the respective time or, equivalently, some partial information about the real demands. The other type of measure is with respect to the complete oracle, who knows all information in advance, i.e. the 5 suboptimal plans at the end of the day. The latter kind is more important as in practice it would give us a real view on how many correct decisions have been made during the last day. However, the former is also important because it would allow us to pursue local desires besides global ones. One may be interested in a learning process, whose goal is to achieve some desired total costs by following a policy which selects an action taking into account both global and local measures.

Global correctness

This section puts forward a comparison of the global results obtained during the experiments. In Tables 4.4, 4.5 and 4.6 we report the average values of (1) the precise hits for each heuristic (the value of f_1) and (2) the vehicle type plus the geographical type evaluation results ($f_2 + f_3$) for both scenarios. i.e. with and without expected client needs (*pday101-pday200*). We accumulate the values of f_2 and f_3 since both of them concern compatible vehicle types, however, we lose some more detailed information about the correct hits.

	Without Model		**With Model**	
	5-19:00		**5-19:00**	
h	f_1	$f_2 + f_3$	f_1	$f_2 + f_3$
MIN	293	674	265	703
BAL	6	403	2	450
SHIFT	38	577	39	654
CUR	581	874	575	910
GEO	88	625	77	650
IMM	75	616	43	661
MAV	303	666	281	698
RAND	101	571	86	615

Table 4.4: Once-a-day pooling strategy : global valuation statistics.

	Without Model		**With Model**	
	5-19:00		**5-19:00**	
h	f_1	$f_2 + f_3$	f_1	$f_2 + f_3$
MIN	347	591	273	628
BAL	5	388	2	399
SHIFT	48	551	45	593
CUR	615	731	597	652
GEO	89	554	80	582
IMM	78	581	46	616
MAV	360	581	291	614
RAND	113	525	89	554

Table 4.5: Time-zones pooling strategy : global valuation statistics.

	Without Model		**With Model**	
	5-19:00		**5-19:00**	
h	f_1	$f_2 + f_3$	f_1	$f_2 + f_3$
MIN	348	563	281	620
BAL	5	366	2	388
SHIFT	49	521	46	581
CUR	632	649	550	689
GEO	89	528	83	576
IMM	79	557	47	607
MAV	361	554	299	607
RAND	113	499	93	545

Table 4.6: Fixed-time-span pooling strategy : global valuation statistics.

The first observation we would like to discuss is regarding the decrease in the precise numbers when the stochastic model is used. Note that the presence of seeming errands adds an additional workload to the overall fleet. Thus, when Indigo produces a suboptimal plan for this higher number of demands it may significantly disalocates the vehicles with respect to a situation, in which vehicles carry only real packages. Consequently, in the former case the precise vehicle route is more likely to be mistakenly guessed by a heuristic. However, the benefit of the model is observed if one compares the type valuation measures (i.e. the columns 2 and 4 in the tables). When the predictor is in force the correct decisions regarding the vehicle type are more. Hence, the model is useful to predict the least vehicle type to which a new order need to be assigned. This is advantageous as it restricts the fleet of vehicles to a proper subset. Another positive observation is that reoptimizing more than once per day leads to an increase of the precise valuation indicators. Note that these numbers in *time-zones* pooling strategy are noticeably higher than in *once* scenario and are quite similar to *fixed-time-span* statistics. Hence, introducing few time slices is beneficial, while many of them do not improve much the statistics. As opposed to this, the vehicle type measures tend to decrease when multiple reoptimizations are performed.

We continue with an heuristic comparison within the different pooling strategies. The CUR rule performed best regarding all the measures (f_1, f_2 and f_3) in all strategies. The latter is expected as Indigo applies a combination of heuristic approaches when placing the requests, while we are using simple heuristics to achieve that result. However, on the second and third places regarding the precise measure are MAV and MIN, respectively. MAV scores between 303 and 361 correct hits when the model is not used, and between 281 and 299 otherwise. The numbers for MIN are similar with range of $[293, 348]$ when seeming clients are absent, and $[265, 281]$ otherwise. According to the vehicle type measure the results are not so distinguishable. In all cases we have high numbers, however, some of them must be noticed. For instance, CUR achieves 874 correct hits in *once*-setting with the stochastic model and 910 without using it. These values decrease when multiple time slices are used. At the second position in this setting is MIN with 674 and 703 regarding the different scenarios, followed by MAV with 666 and 698. In the other two settings most of the results are getting fuzzy (with respect to $f_2 + f_3$).

BAL rule performed worst in all settings. The rationale behind this is partly because the rule tries to distribute the current total workload and, thus, the probability of actuating a new vehicle is high. However, this is not among the invariants Indigo uses when it evaluates the overall current status and inserts a new errand. An interesting observation is that we can view the hits each rule achieves as a measure of covering between our heuristics and the methods implemented in the oracle. From this perspective, it is possible that BAL heuristic performs better with respect to another VRP solver.

4.4.7 Global vs. local correctness

At the beginning of Section 4.4.6 we mentioned a possibility of an heuristic evaluation with respect to the local minimums. These are computed by incomplete oracles (regarding the entire information), positioned at the time slice boundaries (at moments when a reoptimiza-

tion is performed).In this section we compare the global with these local heuristic correct decisions made in the different scenarios. Although, this comparison is only with respect to the overall daily statistics, we refer the inquisitive reader to the Appendix where we report many more numbers regarding the different time slices. The last could be used to compare the dispatch rules within a particular time slice rather than only a comparison based on the entire-day statistics.

	Without Model		**With Model**	
	5-19:00		**5-19:00**	
h	**local**	**global**	**local**	**global**
MIN	661	591	709	628
BAL	296	388	380	399
SHIFT	436	551	582	593
CUR	658	731	723	652
GEO	850	554	868	582
IMM	487	581	598	616
MAV	660	581	704	614
RAND	545	525	627	554

Table 4.7: Time-zones pooling strategy : global vs. local statistics.

	Without Model		**With Model**	
	5-19:00		**5-19:00**	
h	**local**	**global**	**local**	**global**
MIN	659	563	724	620
BAL	292	366	389	388
SHIFT	433	521	593	581
CUR	673	649	782	689
GEO	847	528	885	576
IMM	483	557	609	607
MAV	659	554	720	607
RAND	543	499	640	545

Table 4.8: Fixed-time-span pooling strategy : global vs. local statistics.

Note that we report the statistics only for the multiple-slice scenarios because when we reoptimize only once during the day the local minimum coincide with the global one and, thus, the values of the local and the global valuations coincide. In Tables 4.7 and 4.8 we give the local and the global values for the accumulated vehicle type valuation function $(f_2 + f_3)$ in both the *time-zones*- and *fixed-time-span*-strategies. As one can notice the numbers are quite fuzzy. Some of the rules perform better (MIN, GEO, MAV, RAND in both time horizon divisions and in addition CUR in *fixed-time-span* setting) and other worse (BAL, SHIFT, IMM in *time-zones* and *fixed-time-span*, and *CUR* only in *time-zones*) with respect

to the global values. We believe this random behaviour is due to the constraints each rule imposes when evaluates the current schedule and, thus, some of them benefit from the local and other from the global "knowledge" about the demands. In addition, we underline that GEO performed significantly better (more than 847 correct hits) with respect to the local minimums than the global ones. We believe this is because it considers features similar to the ones Indigo does.

4.5 Summary

In conclusion we summarize our efforts in this chapter. We study the feature distributions in the original dataset provided by the company and based on the statistics obtained we generated 330 datasets which suit our problem instance. Then we discuss some issues related to the quality of the Indigo solutions followed by the heuristic evaluation procedure we used in all the three pooling strategies (*once*, *time-zones* and *fixed-time-span*). We proceed by defining several notions extending our theoretical framework in Section 2.1. Among these are *solution* and *partial solution*, which we use to formalize the different strategies. We rank the implemented heuristics according to three valuation measures, i.e. *precise*, *vehicle type* and *geographical*. Using these we define the notion of *heuristic correctness* used in the evaluation settings thereafter. Each heuristic decision is compared with respect to ten suboptimal plans (5 with and 5 without "fake" orders) produced at each time slice boundary. We report some major statistics in the main body of the thesis and present the rest in the Appendix. Among these are (1) the average individual cost each heuristic achieves when assigning a new order, (2) the time performance of the rules and (3) their global and local correctness results. We conclude by presenting a comparison between the local and the global rule performances in terms of correct decisions.

Chapter 5

Learning Experiments

Learning is the engine of life.
by Martin Aleksandrov

This chapter gives a flavor on how we can learn dispatch policies within the framework we work with. We start with a short introduction on machine learning approaches and address recently emerging adaptive dynamic programming, which overcomes some of the drawbacks in the classical setting.

The main goal of our work is to learn how heuristic decisions depend on fleet and current schedule features, to be able, given the parameters of the vehicles and their routes, to decide which rule will perform best. Given the distribution of decisions based on the past data, there is no expert able to say which rule has better local influence on the final route realizations at the end of the day, hence we learn exactly that distribution of heuristic decisions. As a learning instrument we used Artificial Neural Networks (ANNs) to approximate the decision distribution density function for each time-horizon division setting, i.e. *once-a-day*, *time-zones* and *fixed-time-span*. Regarding selected features, we consider some that appear more informative, such as the *number of active vehicles*, *free capacity* of the nearby vehicles, etc. We trained our ANNs on a hundred days of past data and, thereafter, we tested them on a disjoint collection of thirty distinct days.

We continue with the description of the recommendation system we designed and the different dispatch policies implemented : **MaxSumPG**, **MaxSumPLG**, **MinSumPG** and **MinSumPLG**. Each one of them implements solutions, which we compare with their suboptimal variants produced by Indigo. As a measure of comparison we present various statistics such as the *value of information*, *overall waiting time*, etc.. Finally, we conclude this chapter with a summary of the results obtained.

5.1 Machine Learning

In the last decades, more and more practitioners are trying to automate the process of computer comprehension and interpretation of texts, pictures, voice, etc. The reason for

that is partly because of the tremendously increasing amount of information, which needs to be reasoned about and, in addition, the unability for a human being to process such huge information to achieve some desired and specific objectives. A machine learning program can be defined as a computer program, which improves its performance at some task through some experience ([46]). Learning programs are used when humans are unable to code their expertise in a deterministic algorithm (e.g., speech recognition, medical diagnosis, etc.) or the solution to the problem instance changes in time (spam filters, forecast of energy consumption, etc.), or even when human expertise does not exist (e.g., exploring scientific domains, navigating a robot on Mars, etc.). Thus, the experience can be evaluated by an expert in the domain or it can be represented as an open-interpretative. The former type of learning is called *supervised*, while the latter *unsupervised*. Many efforts are made in studying the properties of the different learning mechanisms. Although, this is not amongst our goals, we underline that for having a good model one definitely needs enough data, representing the experience of the machine learning program. The more experienced the program is, the better results it produces when an unobserved individual is input to it. In other words, the quality of the query answer strongly depends on the information.

5.2 Neural Network

Artificial Neural Network is a mathematical model intensively used in the field of Artificial Intelligence. Its main goal is to simulate the neural dendrites in the human brain, albeit a main difference is that while the model has a particular structure and the "cells" exchange information synchronously, the neurons in the brain network are chaotically distributed and interact asynchronously. Although, there are many specific applications of Neural Networks, the main goal behind the model is learning a particular unknown-in-advance and arbitrary function by presenting multiple examples of it to the net. For example, Hölldobler et al. in 1999 ([30]) applied a 3-layered recurrent neural network in order to approximate the semantics of a given propositional logic program. The latter can be characterized as a fixed-point of a continuous function (immediate consequence operator) operating over the undelying program lattice. Another instance can be found in [53], where Riedmiller S. and Riedmiller M. investigate neural reinforcement approach to learn local dispatching policies in production scheduling. They relate the process of achieving local objectives with the satisfaction of global properties, which resulted in better scheduling specifications. Neural networks are also applied in the area of network communications. In particular, when the desired objective is maintaining the flow of information in the so-called network *backbone* (Wilde, 1998, [57]) or learning hybrid-protocols in wireless/cordless environments (Kojić et al., 2012, [39]). Although, there are many applications in networking, artificial intelligence and production scheduling areas, we are not aware of significant neural research in logistics. The latter motivated us to use that particular model in order to learn the function, which selects the right action to be executed whenever a new event happens, i.e. new order occurs. We do not present any detailed discussion on the architecture of the network we used. Instead, we view it as a black box, which ranks the heuristics according to the current state. In Figure 5.1 one can obtain a flavour of how this mathematical model looks like. There are three types of layers, i.e. *input*, *hidden* and *output*. Each layer contains a particular number of neurons. Between the layers there are links (synapses) through

which the neurons communicate. The numbers of the input and the output neurons are managed by the researcher and reflect, respectively, to the number of input and the output arguments of the desired function. On the contrary, the number of the intermediate layers, the number of the neurons in each one of them and the level of coherence in the network are application- and problem-dependent. A naive approach to determine these numbers is via *try-and-trial* exhaustive procedure until the network converges appropriately. The latter can be a difficult task until the desired setting is found. However, there is another kind of neural learning, which overcomes this drawback of the original approach. Cascade training differs in the sense that it starts with an empty neural network and then adds neurons one by one, while it trains the neural network. The main benefit of this approach is that you do not have to guess the number of hidden layers and neurons prior to training, but cascade training have also proved better at solving some problems. Another advantage is that each trained neuron can possess a different activation function (Braun and Riedmiller, 1993, [8] and Fahlman, 1988, [22]), which makes the network more adaptive than in the ordinary setting.

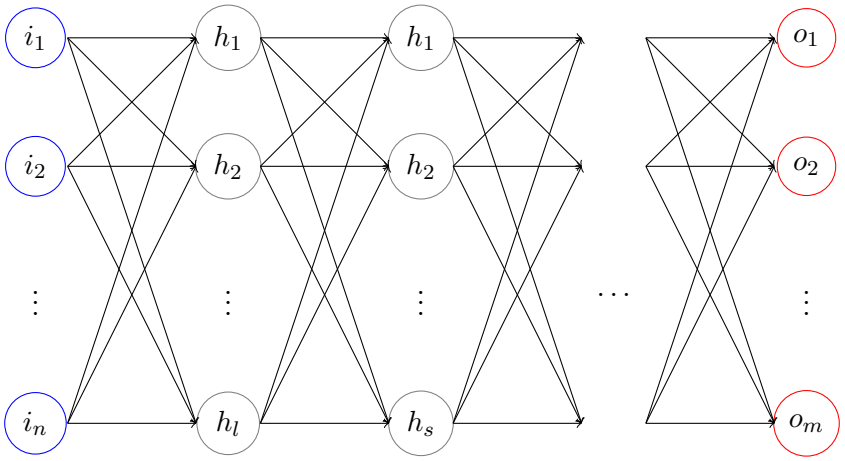


Figure 5.1: Neural network general structure.

The basic idea of cascade training is that a number of candidate neurons are trained separate from the real network, then the most promising is inserted into the neural network. Then the output connections are trained. The process continues until the current network structure converges for the specific problem instance. In our experiments we used cascade training to build 6 neural networks, viz. 2 for each pooling strategy. We will explain the difference within each pair of networks when we discuss the global and the local features in the next section. However, we underline that each pair has (1) one network with 11 input neurons (1 layer), 1 hidden neuron (10 layers) and 18 output neurons (1 layer), and (2) the other one with 11 input neurons, 1 hidden neuron (10 layers) and 26 output neurons (1 layer).

5.3 Features, Datasets and System Scheme

This section presents the scheme under which we conducted the learning experiments. We first present the learning features we considered. The process of selecting the right features has attracted many practitioners to investigate this matter. Many heuristic and exhaustive approaches have been implemented (Yu and Liu, [58], 2003, and Guyon and Elisseeff, [27], 2003) in order to select the most informative characteristics. In fact, we did not devote a special attention to this process and simply present the set of features we selected as seemingly more appropriate. These are divided into *order* and *schedule* features. The order features are summarized in Table 5.1 :

Feature	**Domain**	**Type**
period of the day T	[1,4]	discrete
order number of pallets P	[1,10]	discrete
order weight W	[1,10000]	continuous
minimum vehicle type V	[1,14]	discrete
degree of dynamicity D	[1,10]	discrete

Table 5.1: Demand features.

These order features describe the new client demand. We believe that they are important and heuristic decisions should partly depend on them. Otherwise, it might happen that significantly different errands would be categorized similarly, which must be avoided. For instance, for two orders of 50 and 1500 kilograms, without differentiating the order weight a system would give wrong recommendations regarding the vehicles that could accommodate the demands. Furthermore, we divided the time horizon into four periods, which are characterized with different fleet activity studied from the past data. We encode each such period with feature 1). With feature 4) we introduce an ordering between the different vehicle types and encode them with natural numbers, expressing the least type of vehicle able to accommodate the new requirements. The last feature captures the intensity of the particular arrival moment and it forces the network to take it into account when a recommendation is produced.

In addition to the demand features, we consider the collection of current-schedule features reported in Table 5.2. These are computed together with the previous set of features and they also depend on the characteristics of the arrived dynamic errand. For instance, if the weight of an unreserved demand is 500 kilograms we consider only those active vehicles, which can accommodate that commodity. Moreover, we leave out those active fleet members, which satisfy the quantitative description of the new order, but finish their shift before the earliest time of that demand. Recall, that the latter is a hard constraint and all the drivers finishing their work earlier according to the current schedule would not be available to service the new locations. The same reasoning is valid for the second triplet of features. They describe the members of the fleet, which are nearby with respect to the new locations and at the same time able to handle the new client's requests.

Feature	Domain	Type
number of active vehicles A	[1, 192]	discrete
free pallets capacity in the active vehicles FPA	[0, 480]	discrete
free weight capacity in the active vehicles FWA	[1, 420962]	continuous
number of close vehicles C	[1, 192]	discrete
free pallets capacity in the close vehicles FPC	[0, 480]	discrete
free weight capacity in the close vehicles FWC	[1, 420962]	continuous

Table 5.2: Schedule current features.

The first three schedule features describe the active fleet. However, as we have fixed-fleet size they also describe the inactive vehicles. Feeding the system with these values, it would probably recommend that an actuated vehicle is used and, the system would somehow minimize the number of contracted drivers indirectly, until a moment, in which a rule makes a decision to put a new vehicle on the road. The second group of features integrates the knowledge about the nearby vehicles with respect the new locations into the recommending process. Of course, the final decision will depend on the implemented policy and the limitations imposed by the selected rule. In order to clarify the procedure described so far Figure 5.2 shows schematically the recommendation system we designed.

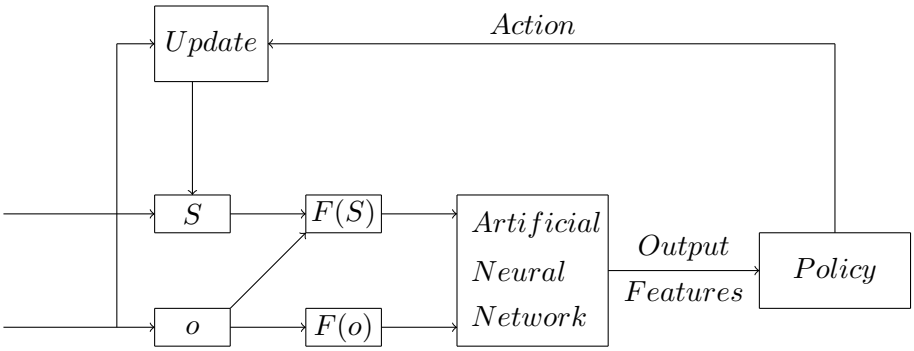


Figure 5.2: Recommendation system scheme.

The system is triggered every time a new order o arrives at the company. It also keeps track of the current schedule S, which contains up-to-date fleet management information. Then it computes the set of demand features $F(o)$ presented in Table 5.1 given the order o. To obtain the set of schedule features $F(S)$ (Table 5.2) the system needs to take into account both the current vehicle timetable S and the errand o. Computed all these, they are given as input values to the ANN, which produces an output vector of features. At this moment we just mention that the output features rank the dispatch rules according to particular measures and, in addition, the system recommends a subset of the fleet, whose vehicles are appropriate to accommodate o. We will discuss these in the next section where we formalize the notion of a dispatch policy. Next, a predefined control selects a rule based on this ranking, according to some policy (according to which the notion of *the best* action

differs). Once an action is chosen, an update operation follows, whose output is a new current schedule. The latter will be used by the system as input parameter when the next demands arrive. The update consists of including o in S according to the recommended action (dispatch rule) and updating the timings at the unvisited customers on the selected vehicle route as well as recalculating their costs and penalties.

More precisely, we next describe the most relevant parts of the scheme above, viz. learning the rankings and selecting an action afterwards. Let $\overline{I} = (F(o), F(S))$ be the vector of input features and let $\overline{O}$ be the vector of the output ones. Thus, $ANN : \overline{\mathcal{I}} \rightarrow \overline{\mathcal{O}}$ is a mapping representing the function, approximated by the neural network, that maps the input vector space $\overline{\mathcal{I}}$ into the output vector space $\overline{\mathcal{O}}$. We use a network to learn this mapping for each pooling strategy, i.e. *once-a-day*, *time-zones* and *fixed-time-span*. Ideally, there would be an expert able to deduce which rule will perform locally best so that the final costs are minimal. Due to the lack of such an oracle the output vector $\overline{O}$ is then given as argument to another mapping $\pi : \overline{\mathcal{O}} \rightarrow \mathcal{A}$, which selects a final action to be executed. We call this mapping a *dispatch policy*. There can be many such guidelines through which we construct a final solution. Some of them are better than others as they control the fleet in a more reasonable way and achieve lower total costs. The best such mappings are called *optimal dispatch policies*. However, finding these can be a difficult task especially when the dimension of the output vector space $\overline{\mathcal{O}}$ is large.

Learning datasets

The format of the datasets used for learning the mapping ANN are now presented. Before that we underline that we conduct two types of learning. One of them takes into account only the global valuations of the rules, while the other observes also the local ones. Therefore, we created two types of datasets for each learning approach. In Figure 5.3 one can see the format of the feature vector representing the "global" datasets. The first 11 values in each row serve as inputs for the network. They represent the vector $\overline{I}$ of order and schedule features. The remaining 18 features constitute the output feature vector $\overline{O}$. There are three groups of elements in it : (1) global *precise* heuristic (f_1) value, (2) global *type* heuristic value ($f_2 + f_3$) and (3) vehicle type range (see Section 4.3.6). For each dispatch rule there is a corresponding value in each of the first two groups. It stands for the evaluation of that rule with respect to 10 suboptimal solutions produced at the end of the day and, thus, it takes values between 0 and 10.

order features					schedule features						precise rule features : global								type rule features : global								subfleet features	
T	P	W	V	D	A	FPA	FWA	C	FPC	FWC	MIN	BAL	SHIFT	CUR	GEO	IMM	MAV	RAND	MIN	BAL	SHIFT	CUR	GEO	IMM	MAV	RAND	w/o	with
1	0	344	5	1	54	93	55868	2	3	917	0	0	0	4	1	1	1	0	7	10	7	1	2	6	3	6	8	7

Figure 5.3: Global learning features.

The last two features express the minimal vehicle type according to those solutions, which is appropriate for the new order. The first is when the prediction model is not used, whereas the second is when there are seeming demands. We decided to include these features as they would allow us to fasten the action application procedure (the search for an appropriate

fleet candidate will be guided). In Figure 5.4 one can observe an example of a dataset tuple used during the second learning technique when we also learn the heuristic grades with respect to the local minimums along the day. They are summarized in an additional group of eight output features. The main idea of this approach is that each policy defined on top of both the local and global rule features will try to minimize not only the global objective, but also the local ones.

order features					schedule features						precise rule features : global								type rule features : local								type rule features : global								subfleet features	
T	P	W	V	D	A	FPA	FWA	C	FPC	FWC	MIN	BAL	SHIFT	CUR	GEO	IMM	MAV	RAND	MIN	BAL	SHIFT	CUR	GEO	IMM	MAV	RAND	MIN	BAL	SHIFT	CUR	GEO	IMM	MAV	RAND	w/o	with
1	0	344	5	1	54	93	55868	2	3	917	0	0	0	4	1	1	1	0	6	6	6	4	1	7	2	6	7	10	7	1	2	6	3	6	8	7

Figure 5.4: Global and local learning features.

More formally, let denote the precise features as $\overline{h_p}$, the type local features as $\overline{h_l}$, the type global features as $\overline{h_g}$, the first vehicle minimal type as T_r and the second one as T_{r+d}. Then the output vector according to the first approach has the form $\overline{O}_1 = (\overline{h_p}, \overline{h_g}, T_r, T_{r+d})$ and $\overline{O}_2 = (\overline{h_p}, \overline{h_l}, \overline{h_g}, T_r, T_{r+d})$, otherwise.

5.4 Dispatch Policies

A policy, in general, is a mapping that, for a given a problem state, selects an appropriate control action. The different policies are mostly compared using the objective values they achieve. However, one can also use other measures, especially, when the problem in hand has multiple parameters. In the latter scenario more complicated evaluation criterion needs to be used in order to evaluate a particular policy.

In our experiments we first learn the dispatch rule ranking and then we apply a dispatch policy on top of it. Note that our notion of policy differs from the general one, however, they can be related in the following way : if $p : \mathcal{S} \to \mathcal{A}$, $ANN : \mathcal{S} \to \overline{\mathcal{O}}$ and $\pi : \overline{\mathcal{O}} \to \mathcal{A}$ represent a policy, a neural network and a dispatch policy, then $p = ANN \circ \pi$. In other words, a policy p is a composition of the mappings ANN and π, where $\mathcal{S}$ is the set of all the problem states, which are described by the input vectors in $\overline{\mathcal{I}}$. A consequence of this is that different policies can be obtained when different neural networks are build or the mapping π is changed. Regarding the neural network, we clarified earlier that for each pooling strategy, viz. *once*, *time-zones* and *fixed-time-span* we built a separate network for each of the learning types, i.e. "global" and "local" ones. Thus, in total we have 6 neural networks capturing the specificities of the distinct horizon divisions. With respect to the action-selecting mapping π we explore four ad-hoc variants of it, which we describe next.

5.4.1 Policy *MaxSumPG*

Let H be the repository of simple heuristics we implemented. The *MaxPmaxG* policy, denoted as π_{PG}, selects the rule with highest total rank composed by the global precise and the global type valuations returned by the neural network. Thus, if $\overline{O} = (\overline{h_p}, \overline{h_g}, T_r, T_{r+d})$ is an output feature vector, then

$$\pi_{PG}(\overline{O}) = \max_{h_i \in H}(\overline{h_p}(i) + \overline{h_g}(i))$$

Note that, selecting the rule with the maximum total sum of both measures does not necessarily imply maximum with respect to some of them.

5.4.2 Policy *MinSumPG*

The previous policy aims at the maximum total heuristic rank. On the contrary, the following policy π_{pg} selects a rule with minimim total rank according to the global features. Thus, given an output vector $\overline{O} = (\overline{h_p}, \overline{h_g}, T_r, T_{r+d})$, then

$$\pi_{pg}(\overline{O}) = \min_{h_i \in H}(\overline{h_p}(i) + \overline{h_g}(i))$$

5.4.3 Policy *MaxSumPLG*

This policy grades the heuristics taking into account not only the global, but also the local ranks, leading to a decision distribution together with the vehicle type lower bounds. The result of pursing this policy leads to selecting actions in a distributed way, adopting an equal weight to each group of features, i.e. precise, local and global types. Then π_{PLG} is defined as follows :

$$\pi_{PLG}(\overline{O}) = \max_{h_i \in H}(\overline{h_p}(i) + \overline{h_l}(i) + \overline{h_g}(i)),$$

,where $\overline{O} = (\overline{h_p}, \overline{h_l}, \overline{h_g}, T_r, T_{r+d})$ is an output feature vector.

5.4.4 Policy *MinSumPLG*

As above we are interested in how worse one policy can be, this time taking into account all the global and local features. The policy π_{plg} can be described as follows :

$$\pi_{plg}(\overline{O}) = \min_{h_i \in H}(\overline{h_p}(i) + \overline{h_l}(i) + \overline{h_g}(i)),$$

Note that this pair of "local" policies together with the pair of "global" ones define ranges of many policies which can be implemented based on the heuristic rankings. Therefore, the control mappings we implemented somehow provide bounds to the performance a policy based on these statistics can achieve.

5.5 Experimental Results

Online algorithms have received a significant systematic research attention because of their computational time properties. The idea arised when Sleator and Tarjan [54] suggested to compare the implemented online algorithm with an optimal offline version. Then a few years later Karlin, Manasse, Rudolph and Sleator [37] coined the term *competative analysis*, which is a framework that allows to evaluate online algorithms. The heuristics implemented are, in fact, online deterministic algorithms, which take a small portion of an input order

sequence together with the current plan and output a set of recommendations. On the contrary, their optimal versions would know the entire order sequence in advance when they make a decision. The latter should naturally result in a better fleet schedule at the end of the day than those obtained incrementally, which do not know any future order input. Although we did not investigate the worst-case theoretical competitive ratios of the described algorithms, we do report their values of advanced information.

The results obtained using the recommendation system described previously with respect to the policies in our attention are shown in Figure 5.5. As one can notice, policy π_{plg} (*MinSumPLG*) achieved the minimum final cost of 12 349.51 dollars for the entire fleet management. This is approximately 33% more than an optimal vehicle control. However, the number of vehicles used following the "winning" policy is almost 3 times the optimal number of vehicles needed to accommodate the same demands. Thus, a multiple-criterion taking into account several parameters might be more appropriate. For instance, policy π_{PG} (*MaxSumPG*) maintains more or less 30 vehicles more than in an optimal management, which is twice less than policy π_{plg}.

	optimal	MaxSumPG	MinSumPG	MaxSumPLG	MinSumPLG
service_time	161.21	161.21	161.21	161.21	161.21
waiting_time	76.40	9.86	21.54	11.18	18.37
travel_time	57.13	136.68	106.01	131.78	93.66
execution_time	294.74	307.75	288.76	304.17	273.23
lateness	0.00	33.66	394.15	312.69	60.96
#unsat	0	209	148	380	80
%unsat	0%	9%	6%	16%	3%
#active_veh	68	98	178	96	177
#av.req./driver	18	13	7	13	7
service_cost	4309.08	4309.08	4309.08	4309.08	4309.08
waiting_cost	2042.05	263.68	575.76	298.96	490.93
travel_cost	1527.12	3653.52	2833.62	3522.45	2503.47
driver_wages	7878.25	8226.28	7718.46	8130.49	7303.48
veh_costs	1314.21	3631.33	3923.69	3585.26	3501.09
penalties	0.00	1342.76	9088.96	3688.95	1544.94
FINAL	9192.46	13200.37	20731.11	15404.70	12349.51
value of info.		0.30	0.56	0.40	0.26

Figure 5.5: Policy comparison in *once* pooling strategy.

Despite the lower number of contracted drivers, the price achieved by π_{PG}-control is around 7% more than by π_{plg}. A possible justification for this result is the the fact that π_{plg} leads often to the selection of rules which actuate new vehicles, possibly among the closest with respect to the new locations. Therefore, the travel costs are less (2503.47 $), but the number of vehicles is high. On the contrary, following policy π_{PG} leads to selecting rules which are likely to select already actuated vehicles. Thus, the number of active drivers is kept low, whereas the travel costs are increased (3653.52$ or 46%). Note that all policies tested lead

to a drastic decrease in the overall waiting times (the red row). That is, the drivers are kept busier than according to the optimal schedule. The best difference is achieved by the policy π_{PG} with only 9 hours and 52 minutes of overall waiting time. However, 9% of the customers are unsatisfied due to late service at their locations. Notice that the policy π_{plg} realizes only 80 delays, which is approximately 3 times less than π_{PG}, but the time spent in lateness is much higher (≈ 60 hours). This means that the system realizes less and larger delays following π_{plg}, whereas pursing π_{PG} leads to many, but shorter late services. The policy performing worst in this scenario is π_{pg}. Its total cost is 20732 with many long delays (148), which leads to very high penalty ($\approx$ 9089\$). The last row in the Table 5.5 represents the *value of information* gained applying each policy (see [41]). It is calculated using the costs achieved by an offline and an online algorithms over the same problem instance. For instance, policies π_{PG} and π_{plg} performed best with ratios of $\frac{13200.37-9192.46}{13200.37} \approx 0.3$ and $\frac{12349.51-9192.46}{12349.51} \approx 0.26$, respectively.

The table in Figure 5.6 contains information about the active fleet management when π_{PG} (*MaxSumPG*) is applied. It is a distribution of the overall statistics achieved by this policy with respect to the vehicle types. Several interesting observations can be drawn from that table. The first row describes the approximated starting times for each type of vehicle. That is, the company should contract some drivers for each vehicle type before the starting time stated in the table. For instance, a motorbike is recommended to be on the road after 8:43 in the morning, a two-tonne van at 13:15 in the afternon and the 14-tonne truck at 17:41 in the late afternoon. The maximum number of vehicles (29) used are from type 1T (1 Tonne Tray). Thus, the cost for controlling them is the highest ($\approx$ 4825\$ or 36% of the entire cost) among all the types. The following red line contains the average numbers of orders each driver serves. These numbers range in the interval $11 - 17$, which is between 1 and 7 orders less than an optimal control parametrized in Figure 5.5.

	all	MB	CAR	SW	SV	1V	1T	2V	2T	4T	6T	8T	12T	14T
start_time		08:43	08:00	07:52	08:01	08:43	09:14	13:15	09:00	10:40	09:54	08:59	08:47	17:41
service_time	161.21	0.08	1.85	2.57	2.30	32.16	60.82	6.23	38.93	12.47	2.22	1.35	0.06	0.07
waiting_time	9.86	0.07	0.40	0.47	0.91	2.18	3.32	0.06	1.72	0.71	0.02	0.01	0.00	0.00
travel_time	136.68	0.38	5.50	7.86	4.05	32.15	53.07	4.32	23.11	5.50	0.55	0.19	0.00	0.00
active_time	297.89	0.45	7.35	10.43	6.35	64.31	113.89	10.55	62.03	17.97	2.77	1.54	0.06	0.07
opt.active_time	218.34	0.92	16.91	24.66	13.53	37.18	75.76	7.18	31.17	10.12	0.57	0.34	0.00	0.00
execution_time	307.75	0.53	7.75	10.89	7.26	66.49	117.21	10.61	63.76	18.68	2.79	1.55	0.06	0.07
lateness	33.66	0.00	0.00	0.01	0.10	8.19	15.39	1.34	7.05	1.42	0.13	0.03	0.00	0.00
#unsat	209	0	0	0	1	54	94	8	42	9	1	0	0	0
%unsat	9%	0%	0%	0%	0%	2%	4%	0%	2%	1%	0%	0%	0%	0%
#active_veh	98	1	4	5	3	20	29	4	17	8	2	3	1	1
#av.ord/driver	13	11	12	15	13	11	14	15	17	12	11	14	16	13
av.dur/driver	3.04	0.45	1.84	2.09	2.12	3.22	3.93	2.64	3.65	2.25	1.39	0.51	0.06	0.07
opt.av.dur/driver	2.23	0.92	4.23	4.93	4.51	1.86	2.61	1.79	1.83	1.26	0.29	0.11	0.00	0.00
service_cost	4309.08	2.05	49.54	68.70	61.39	859.67	1625.84	166.41	1040.54	333.32	59.37	36.03	1.49	1.78
waiting_cost	263.68	1.99	10.59	12.44	24.31	58.36	88.64	1.50	46.05	19.07	0.52	0.21	0.00	0.00
travel_cost	3653.52	10.08	147.04	210.06	108.34	859.32	1418.56	115.57	617.62	146.96	14.80	5.15	0.00	0.00
driver_wages	8226.28	14.12	207.17	291.20	194.04	1777.35	3133.04	283.48	1704.21	499.35	74.69	41.39	1.49	1.78
veh_costs	3631.33	0.92	33.01	66.80	52.69	642.96	1167.54	157.81	1005.10	412.35	63.68	28.43	0.00	0.02
penalties	1342.76	0.00	0.00	0.49	4.10	285.39	524.26	63.19	340.53	104.95	15.97	3.88	0.00	0.00
FINAL	13200.37	15.04	240.18	358.49	250.84	2705.70	4824.84	504.48	3049.84	1016.65	154.35	73.69	1.49	1.81

Figure 5.6: Fleet management under π_{PG} policy.

Other interesting result is the driver's active performance shown in the second row in red.

With the π_{PG} policy the following types are intensively used : 1V (Large Van, $\approx$ 65h), 1T (1 Tonne Tray, $\approx$ 114h), 2T (2 Tonne Tray, $\approx$ 62h) and 4T (4 Tonne Tray, $\approx$ 18h). However, this is not the case of an optimal management (shown right below in red) which distributed the overall workload much more uniformly. For instance, CAR (Car, $\approx$ 17h), SW (Station Wagon, $\approx$ 25h) and SV (Small Van, $\approx$ 13h) are used more than twice longer. This contributes for the lower final cost in the suboptimal plan, since the cost for running these vehicle types is less than the cost for running the most used by the policy π_{PG}. Similar results can also be noticed in the driver's average active duration, where the maximum shifts are performed by the 1T-drivers, who are either serving or travelling towards a client for maximum of 4 hours. On the contrary, the maximal optimal shifts are between 4 and 5 hours for CAR-, SW- and SV-drivers. On the other end of shift duration, are vehicles of type MB (Motorbike), 8T (8 Tonne Tray), 12T (12 Tonne Tray) and 14T (14 Tonne Tray) with less than 1 hour shifts. Even the last two are only run for a few minutes, due to servicing only very few clients ($2, 3$ for instance). Therefore, we think that a better policy would assign (if possible) their jobs to vehicles of "lower" type. This can be confirmed by the suboptimal plan, whose results regarding the average driver's active duration can be seen in the line below, and do policy does not use such large trucks (12T and 14T).

We continue by comparing the policy performance under *time-zones* pooling strategy shown in Figure 5.7. We have 5 time slices, when multiple unreserved demands arrive. The idea is that at each time-slice boundary there is complete reoptimization (that is rerouting and rescheduling) of the order information. For each time slice we show a report with the results, similar to the one above when we reoptimize only one time for the entire day. As above, *MinSumPLG* and *MinSumPG* used many vehicles in order to handle the multiple client needs, which led to lower objective costs. On the contrary, *MaxSumPG* and *MaxSumPLG* used much less vehicles to accommodate the same number of orders. The waiting time is reduced in all policies, especially when the latter policies are applied. Another interesting result about the fleet management is the number of vehicles needed during the different parts of the day (the line with red and orange colors). In the morning the dispatchers need to contract between 29 (optimal schedule) and average of 43 drivers according to the *MaxSumPG* and *MaxSumPLG*, and 90 with respect to the *MinSumPG* and *MinSumPLG*. As the day progresses the need of more drivers emerges with the following distribution : (1) 9:00-11:00 : between 15 and 16 pursuing the "maximal" policies, and approximately 31 following the "minimal" ones, (2) approximately 7 new contracts are needed between 11:00 and 13:00 if *MaxSumPG* and *MaxSumPLG* are applied, and $15 - 19$ otherwise, (3) even less new contracts ($4 - 5$) in the early afternoon (13:00-15:00) according to *MaxSumPG* and *MaxSumPLG* and with respect to the others some of the drivers need to be sent home (between $27 - 33$) and, finally, (5) at the end of the shift $3 - 4$ more drivers are needed following the "maximal" policies and more drivers need to be sent home pursuing the "minimal" ones. All this information can be used to determine how many drivers need to be contracted before a particular time slice boundary and how many of them need to be released from duty during the day. Besides these statistics one can observe that the value of information gained with respect to the optimal solutions decreases through the day, which is expected given the increased amount of information used by the heuristics to make their decisions. More interestingly, we can see that in the afternoon (after 13:00)

	5:00-9:00					9:00-11:00				
	optimal	MaxSumPG	MinSumPG	MaxSumPLG	MinSumPLG	optimal	MaxSumPG	MinPminG	MaxSumPLG	MinSumPLG
service_time	49.05	48.93	48.72	48.99	48.72	91.08	91.08	90.99	91.08	90.95
waiting_time	28.83	18.16	20.69	17.69	20.27	40.50	21.67	29.89	21.68	29.27
travel_time	20.21	35.75	29.90	35.50	30.94	31.99	56.70	40.84	56.90	43.21
execution_time	98.09	102.84	99.31	102.18	99.93	163.57	169.44	161.72	169.66	163.43
lateness	0.00	4.54	1.38	27.24	1.09	0.00	8.52	12.93	9.77	0.74
#unsat	0	29	3	45	4	0	56	17	60	4
%unsat	0%	1%	0%	2%	0%	0%	2%	1%	3%	0%
#active_veh	29	42	90	44	89	45	58	124	58	122
#av.req./driver	44	30	14	28	14	28	21	10	22	10
service_cost	1311.20	1307.99	1302.29	1309.41	1302.29	2434.48	2434.48	2432.10	2434.48	2431.21
waiting_cost	770.55	485.36	553.03	472.93	541.76	1082.55	579.27	798.84	579.57	782.31
travel_cost	540.17	955.52	799.15	948.91	827.12	855.15	1515.50	1091.70	1520.83	1155.06
driver_wages	2621.92	2748.87	2654.46	2731.26	2671.17	4372.18	4529.25	4322.64	4534.88	4368.58
veh_costs	402.96	823.46	772.37	801.09	797.25	708.62	1494.15	1235.20	1446.29	1153.90
penalties	0.00	181.71	48.30	359.94	45.12	0.00	338.96	278.19	364.35	44.93
FINAL	3024.88	3754.04	3475.13	3892.28	3513.54	5080.81	6362.37	5836.03	6345.51	5567.41
		0.19	0.13	0.22	0.14		0.20	0.13	0.20	0.09

	11:00-13:00					13:00-15:00				
	optimal	MaxSumPG	MinSumPG	MaxSumPLG	MinSumPLG	optimal	MaxSumPG	MinSumPG	MaxSumPLG	MinSumPLG
service_time	129.14	129.14	129.06	129.11	129.08	149.40	149.40	149.21	149.40	149.31
waiting_time	59.83	44.55	50.91	44.71	50.96	69.51	57.58	60.50	57.51	60.64
travel_time	44.50	67.67	50.42	67.50	52.29	52.03	63.62	52.12	63.70	52.02
execution_time	233.47	241.36	230.39	241.33	232.32	270.93	270.60	261.82	270.61	261.97
lateness	0.00	10.21	23.99	12.48	1.23	0.00	4.78	3.75	6.94	0.31
#unsat	0	59	23	63	7	0	31	8	34	1
%unsat	0%	2%	1%	3%	0%	0%	1%	0%	1%	0%
#active_veh	56	66	143	64	137	65	70	110	69	110
#av.req./driver	22	19	9	19	9	19	18	11	18	11
service_cost	3451.94	3451.94	3449.86	3451.05	3450.16	3993.34	3993.34	3988.29	3993.34	3990.97
waiting_cost	1599.26	1190.81	1360.71	1195.20	1362.10	1857.99	1539.14	1617.06	1537.17	1621.01
travel_cost	1189.56	1808.82	1347.81	1804.40	1397.63	1390.68	1700.67	1393.09	1702.80	1390.44
driver_wages	6240.75	6451.57	6158.38	6450.65	6209.89	7242.01	7233.16	6998.44	7233.31	7002.42
veh_costs	1018.79	1690.66	1335.03	1667.47	1231.69	1189.87	1481.55	1248.63	1493.83	1208.48
penalties	0.00	387.21	563.71	413.34	65.32	0.00	198.78	100.61	233.83	8.83
FINAL	7259.55	8529.44	8057.13	8531.45	7506.91	8431.89	8913.48	8347.68	8960.97	8219.73
		0.15	0.10	0.15	0.03		0.05	-0.01	0.06	-0.03

	15:00-19:00				
	optimal	MaxSumPG	MinSumPG	MaxSumPLG	MinSumPLG
service_time	161.21	161.21	160.99	161.19	160.99
waiting_time	75.17	68.31	69.46	68.31	69.69
travel_time	57.13	63.49	56.97	63.90	56.20
execution_time	293.51	293.01	287.43	293.39	286.87
lateness	0.00	2.54	4.97	2.97	0.05
#unsat	0	17	6	18	0
%unsat	0%	1%	0%	1%	0%
#active_veh	68	73	97	73	99
#av.req./driver	18	17	13	17	13
service_cost	4309.08	4309.08	4303.32	4308.49	4303.17
waiting_cost	2009.41	1826.04	1856.76	1825.90	1862.75
travel_cost	1527.06	1697.14	1522.92	1707.99	1502.22
driver_wages	7845.56	7832.27	7682.99	7842.37	7668.15
veh_costs	1315.56	1484.55	1357.33	1489.84	1294.15
penalties	0.00	111.10	170.95	124.59	0.64
FINAL	9161.12	9427.92	9211.28	9456.80	8962.93
		0.03	0.01	0.03	-0.02

Figure 5.7: Policy comparison in *time-zones* pooling strategy.

some of the policies outperform the cost produced by the solver. The last result can be explained partly by the facts that Indigo is a complicated heuristic and, thus, sometimes it may be more appropriate to use a simpler method in order to decide where to include a new demand, especially if the degree of dynamism is not that large. Finally, we note the possibility of using different policies within the different time slices; applying this strategy we can maintain the system parameters (e.g. number of vehicles, cost, waiting time) as desired. A similar table concerning the *fixed-time-span* pooling strategy can be found in the Appendix.

5.6 Summary

In this chapter we began with some notes about Machine Learning in general, and then focusing on neural networks, the particular framework under which we conducted our experiments. We described the general structure of a neural network and explained how it is used throughout our research, and features that were considered. These include a set of *order* and *schedule* features, each with a particular domain. Next, we showed and explained the structure of the recommendation system we designed together with the types of learning datasets we used. Using it during our course of work we integrated 5 dispatch policies based on which different fleet schedules were produced for each pooling strategy (*once*, *time-zones* and *fixed-time-span*). Among these are *MaxSumPG*, *MinSumPG*, *MaxSumPLG* and *MinSumPLG*. The best results in terms of final costs were produced by *MinSumPLG* and by *MaxSumPG* when considering both number of used vehicles and total cost. Following these control mappings, the online system we developed achieves costs within 34-43% above those obtained with the offline one. In case of the *time-zones* pooling strategy our system outperformed Indigo within some of the time slices.

Chapter 6

Future Perspective

Every perspective is a reason to continue.
by Martin Aleksandrov.

In this section we consider four future perspectives related with our research. The first one presents the situation when we use multiple vehicles to process a given client demand. We discuss some of the relevant issues such as *stores*, *meeting points*, increased problem dynamics, etc..

The second possible extension affects the subject to include the traffic congestion under consideration. In order to obtain more realistic model of the addressed problem it could be essential to consider a distribution of the vehicle motion within the considered area.

Next, we give a flavour of how hyper-heuristics can be implemented and what is the benefit of using them. These observe a repository of simple dispatching rules and capture higher-order dependencies between them. The key idea is that a hyper-heuristic works on a heuristic search space and it processes sequences of simple heuristics in order to establish the best subsequence combinations between these unitary rules.

Lastly, we discuss the subject of channel fleet management. In our work we considered all the vehicles as available to one "super" dispatcher, who controls them. However, in the real case study the dispatchers manage different channels (subsets of the fleet) and it would be an interesting extension to our work to see what kind of research questions emerge in such a scenario.

6.1 Cross-utilization

The modeled problem does not observe the possibility one package to be processed by more than one vehicle. However, in reality, such a combined vehicle use might improve the quality of the entire fleet schedule as well as it may help achieving lower costs at the end of the working day. An errand calling within the dispatcher working time horizon, to be served the same day, can benefit from the multi-vehicle usage perspective. Thus, we could consider a case where the very same load can be transferred once (or more times) between vehicles from the fleet. In addition, a package can be stored at some location and later on to be load again by ensuring an on-time delivery or it can be transferred directly between vehicle racks.

Package transferring using stores An option to transfer a package between vehicles is to store it somewhere on the map and later on another vehicle to pickup it up from there. Once the transfer order and the pair of the exchange vehicles are decided, then the transfer can be simulated by two new requests. In order to consider some place as an appropriate for a temporary storage of loads the warehouse located there needs at least to be opened after the earliest time of the request considered for transfer. An advantage in this situation is that the locations of the warehouses are known in advance. Thus, we only need to decide when such an exchange would occur. In addition, the stores have fixed maximal capacities, which usually are higher than the free space in any of the vehicles, for instance. Even further, the time windows at the store locations can be used to manage the fleet accordingly, which would allow for processing more short-distance, but higher-commodity services.

Direct package transferring In this case we could model the desired property using the notion of *vehicle meeting points*. The idea is to have common locations within vehicle routes, where they can meet and exchange some goods. The latter can be represented as additional pickup and delivery requests. However, this time they do not have to be assigned to the same vehicle. Rather than, one of the vehicles will deliver the demand to the meeting point and the other will pick it up from there within a tight time interval.

Nevertheless, the above proposal raises some interesting questions related to the modelling of these meeting points :

1. Is the transfer better than a direct service ?
2. If the transfer decision is made, how can we decide when and where to perform such a package exchange ?
3. How far in advance we can decide the number of transfers for a given request ?

In order to answer 1) we need to be able to evaluate a particular transfer without actually doing it. Then we can compare this evaluation with the cost for direct service according to the current schedule. The latter can be realized by a standart heuristic technique. Question 2) requires to take into account several features of the current vehicle busyness. Among these can be an estimate of the future free vehicle space and a list of the locations through which a vehicle is passing by. The first feature will help us in determining when the transfer

can be made, and the second where it can happen. Question 3) is related to the problem dynamics. Each transfer increases the dynamics of the addressed instance, and if many such exchanges are decided, that contribution is significant and undesired. Consequently, if we know in advance how many transfers are needed for a specific demand, then it would be easier to handle that additional computational burden.

6.2 Traffic Congestion

The information about traffic condition is an important improvement if we want to shorten the "distance" between the real problem and our model. We believe that traffic congestion can be modelled as a distribution of the travel time between any two locations within the considered geographical area with respect to the time line along the working day. The traffic in the main city area is, generally, more intensive than in the areas around it, i.e. regional centers. This difference can be expressed in the model, and a vehicle which intersects the boundary between two traffic-distinct areas will have its travel time computed accordingly.

6.3 Hyper-heuristics

During the learning experiments we investigate a situation when linear effects between the features are present. However, it might be the case that additional higher-order dependencies exist. The study about these regularities could bring us closer to the desired exact solution to the problem in hand. The recent literature addresses the emerging topic of using hyper-heuristics in a sequentional decision-making processes (survey of the hyper-heuristics [9], production scheduling [10]). The key idea behind a hyper-heuristic is that it operates over a repository of simple heuristics. That is, while the search over the problem state space is conducted under the supervision of these unitary rules, a typical hyper-heuristic approach tries to establish control over them by observing sequences of decisions representing that search. During the second phase of our experiments we already consider such a control. Although, we currently select actions in an ad-hoc manner, the policy learning can be improved if we observe additional higher-order heuristic interactions.

6.4 Channel fleet management

This thesis considers a case in which whenever a new order arrives all the vehicles are available to one "super" dispatcher, who controls them. However, in the real case study the dispatchers control distinct channels which reflect on different subsets of the entire fleet. Each channel corresponds to a particular vehicles type range. In addition, the dispatchers cannot exchange information between each other. Thus, we can view the whole situation as a two-team *game* with incomplete knowledge. One team represents a "super" customer, who generates the flow of order information implied by the clients and the other will be the dispatcher team. However, without an information exchange, the dispatchers do not seem to act like a team, they need to achieve a common objective, i.e. minimizing the overall final costs. An interesting question regarding this extension is the following : "Should we split the number of known orders into proper sets according to the number of channels and

then construct suboptimal plans for each of the subsets OR we first construct a suboptimal plan for all the orders and then project it over the different channels ?". In the former case each dispatcher will have an available suboptimal schedule for his own channel and in the second one he will have a plan regarding his vehicles, but taking into account the information in the remaining channels. In the latter case the dispatchers do not explicitly exchange information amongst them. Rather than some common information will be shared implicitly as being incorporated into the routes of his vehicles. It would be interesting to see in which situation the dispatchers manage the fleet more reasonable.

Appendix A

Appendix References

The thesis titled "Heuristics and Policies for Online Pickup and Delivery Problems" is supported by materials which clarify the details about the program implementations and contain additional results obtained during the various experiments we designed. These supplements are available through one of the following ways :

- http://centria.di.fct.unl.pt/∼admartin/
- aleksandrov.d.martin@gmail.com
- m.aleksandrov@fct.unl.pt

Bibliography

[1] E. Angelelli, M.W.P. Savelsbergh, and M.G. Speranza. Competitive analysis for dynamic multi-period uncapacitated routing problems. *Networks*, 49(4):308–317, 2007.

[2] S. Anily and R. Hassin. The swapping problem. *Networks*, 22:419–433, 1992.

[3] P. Awasthi and T. Sandholm. Online stochastic optimization in the large: Application to kidney exchange. *IJCAI'09 Proceedings of the 21st international jont conference on Artifical intelligence*, 21:405–411, 2009.

[4] R. Baldacci. Recent advances in vehicle routing exact algorithms. Available at `http://people.dii.uniroma2.it/OR%20Group/Baldacci.pdf`., 2008.

[5] R. Baldacci, N. Christofides, and A. Mingozzi. An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming*, 115:351–385, 2008.

[6] A. Beham, M. Kofler, S. Wagner, and M. Affenzeller. Agent-based simulation of dispatching rules in dynamic pickup and delivery problems. *Logistics and Industrial Informatics, LINDI 2009*, 2:1–6, 2009.

[7] G. Berbeglia, J. Gordeau, I. Gribovskaia, and G. Laporte. Static pickup and delivery problems: A classification scheme and survey. *TOP*, 1(15):1–31, 2007.

[8] H. Braun and M. Riedmiller. A direct adaptive method for faster backpropagation learning: the rprop algorithm. *IEEE International Conference on Neural Networks*, 1:586–591, 1993.

[9] E.K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and R. Qu. Hyper-heuristics: A survey of the state of the art. *Computer Science Technical Report*, 1(NOTTCS-TR-SUB-0906241418-2747), 2010.

[10] E.K. Burke, G. Ochoa, S. Petrovic, and A. Vázques-Rodríguez. Dispatching rules for production scheduling: a hyper-heuristic landscape analysis. *Evolutionary Computation, 2009. CEC '09. IEEE Congress*, pages 1873–1880, 2009.

[11] B. Chandran and S. Raghavan. Modeling and solving the capacitated vehicle routing problem on trees. *The Vehicle Routing Problem: Latest Advances and New Challenges Operations Research/Computer Science Interfaces*, 43:239–261, 2008.

[12] N. Christofides, A. Mingozzi, and P. Toth. Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Mathematical Programming*, 20(1):255–281, 1981.

[13] L. Chun-Mei. Pickup and delivery problem with stochastic travel times for semiconductor supply chains. *AIP Conference Proceedings*, 1368(1):197–200, 2011.

[14] J. Cordeau and G. Laporte. The dial-a-ride problem (darp): Variants, modeling issues and algorithms. *4OR*, 1:89–101, 2003.

[15] C. E. Cortés, A. Núñez, , and D. Sáez. Hybrid adaptive predictive control for a dynamic pickup and delivery problem including traffic congestion. *International Journal of Adaptive Control and Signal Processing*, 22(2):103–123, 2008.

[16] J. Côté, M. Gendreau, and J. Potvin. Large neighborhood search for the pickup and delivery traveling salesman problem with multiple stacks. *Networks*, 60(1):19–30, 2012.

[17] G. B. Dantzig, R. D. Fulkerson, and Johnson S. Solution of a large-scale travelling-salesman problem. *Operation Research*, 2:393–410, 1954.

[18] M. Desrochers, J.K. Lenstra, and M.W.P. Savelsbergh. A classification scheme for vehicle routing and scheduling problems. *European Journal of Operational Research*, 46:322–332, 1990.

[19] M. Dessouky and L. Quan. An exact algorithm for the multiple vehicle pickup and delivery problem. *Transportation Science*, 38(4):503–514, 2004.

[20] G. Dong, Y. Kong, K.K. Lai, and J. Tang. An exact algorithm for vehicle routing and scheduling problem of free pickup and delivery service in flight ticket sales companies based on set-partitioning model. *J Intell Manuf*, 22:789–799, 2011.

[21] H. L. A. Emile and H. M. K. Jan. *Simulated Annealing and Boltzmann machines - A Stochastic Approach to Combinatorial Optimization and Neural Computing*. Wiley-Interscience series in discrete mathematics and optimization. Wiley, 1990.

[22] S. E. Fahlman. An empirical study of learning speed in back-propagation networks. *Carnegie Mellon University, Computer Science Department*, 88-162:17, 1988.

[23] L. Ganapathy, L. Priyanka, N. Sambandam, and P. Vachajitpan. Heuristic methods for capacitated vehicle routing problem. *The Vehicle Routing Problem: Latest Advances and New Challenges Operations Research/Computer Science Interfaces*, ThaiVCML, 2009.

[24] M. Gendreau, F. Guertin, J. Potvin, and R. Séguin. Neighborhood search heuristics for a dynamic vehicle dispatching problem with pick-ups and deliveries. *Transportation Research Part C*, 14(3):157–174, 2006.

[25] M. Golder and S. Golder. Random variables and probability functions. Available at `https://files.nyu.edu/mrg217/public/lecture5_handouts.pdf`., 2005.

[26] I. Gribkovskaia and G. Laporte. One-to-many-to-one single vehicle pickup and delivery problems. *Operations Research/Computer Science Interfaces Series*, 43(2):359–377, 2008.

[27] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3:1157–1182, 2003.

[28] H. Hernández-Pérez and J. Salazar-González. The multi-commodity pickup-and-delivery travelling salesman problem. Available at `http://hhperez.webs.ull.es/BertinoroPDTSP.pdf`., 2005.

[29] L. Holder. Np-completeness. Available at `http://www.eecs.wsu.edu/~holder/courses/CptS223/spr09/slides/npc.pdf`., 2009.

[30] S. Hölldobler, Y. Kalinke, and H. Störr. Approximating the semantics of logic programs by recurrent neural networks. *Applied Intelligence*, 11(1):45–58, 1999.

[31] L.M. Hvattum, G. Laporte, and A. Lokketangen. A branch-and-regret heuristic for stochastic and dynamic vehicle routing problems. *Networks*, 49(4):330–340, 2007.

[32] S. Ichoua, M. Gendreau, and J. Potvin. Exploiting knowledge about future demands for real-time vehicle dispatching. *TRANSPORTATION SCIENCE*, 40(2):211–225, 2006.

[33] S. Irnich. A multi-depot pickup and delivery problem with a single hub and heterogeneous vehicles. *European Journal of Operational Research*, 122(2):310–328, 2000.

[34] P. Jaillet, H. Mahmassani, and J. Yang. Real-time multi-vehicle truckload pick-up and delivery problems. *Transportation Science*, 38(2):135–148, 2004.

[35] P. Jaillet and M. R. Wagner. Online vehicle routing problems: A survey. *Operations Research/Computer Science Interfaces Series*, 43(2):221–237, 2008.

[36] M. Johnson. Np-completeness of the travelling salesman problem. Available at `http://www.dur.ac.uk/matthew.johnson2/teaching/tc/lectures/lecture8.pdf`., 2009.

[37] A. R. Karlin, M. S. Manasse, L. Rudolph, and D. D. Sleator. Competitive snoopy caching. *Algorithmica*, 3:77–119, 1988.

[38] P. Kilby, P. Prosser, and P. Shaw. Dynamic vrps : A study of scenarios. *Report APES*, 53, 1998.

[39] N. Kojić, I. Reljin, and B. Reljin. A neural networks-based hybrid routing protocol for wireless mesh networks. *Sensors 2012*, 12:7548–7575, 2012.

[40] L. Kopmanz, W.R. Pulleyblankx, T.K. Ralphsy, and Jr. L.E. Trotter. On the capacitated vehicle routing problem. *INFORMS*, 2001.

[41] R. Krishnamurti, G. Laporte, and S. Mitrović-Minić. Double-horizon based heuristics for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B*, 38(7):669–685, 2004.

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

[43] G. Laporte and S. Mitrović-Minić. Waiting strategies for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B*, 38(7):635–655, 2004.

[44] A. Larsen, O. Madsen, and M. Solomon. Partially dynamic vehicle routing models and algorithms. *Journal of the Operational Research Society*, 53:637–646, 2002.

[45] M. Mes, P. Schuur, and M. van der Heijden. Look-ahead strategies for dynamic pickup and delivery problems. *OR Spectrum*, 32(2):395–421, 2010.

[46] T. M. Mitchell. *Machine learning*. McGraw Hill series in computer science. McGraw-Hill, 1997.

[47] S. Mitrović-Minić. Pickup and delivery problem with time windows: A survey. *SFU CMPT TR*, 12:669–685, 1998.

[48] J. Nešlehová. On rank correlation measures for non-continuous random variables. *Journal of Multivariate Analysis*, 98(3):544–567, 2007.

[49] OpenCourseWare. Poisson processes. Available at `http://ebookbrowse.com/gdoc.php?id=284222304&url=502311512833ccb714d76b849a8283e3`., 2011.

[50] S.N. Parragh, K.F. Doerner, and R.F. Hartl. A survey on pickup and delivery problems part ii: Transportation between pickup and delivery locations. *Journal für Betriebswirtschaft*, 58(2):81–117, 2008.

[51] H.N. Psaraftis. Dynamic vehicle routing problems. *Vehicle Routing: Models and Studies*, B.L.Golden, A.A. Assad(Eds.):223–248, 1988.

[52] H.N. Psaraftis. A multi-commodity, capacitated pickup and delivery problem: The single and two-vehicle cases. *European Journal in Operation Research*, 215(3):572–580, 2011.

[53] S. C. Riedmiller and M. A. Riedmiller. A neural reinforcement learning approach to learn local dispatching policies in production scheduling. In T. Dean, editor, *IJCAI*, pages 764–771. Morgan Kaufmann, 1999.

[54] D. Sleator and R. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.

[55] M.R. Swihart and J.D. Papastavrou. A stochastic and dynamic model for the single-vehicle pick-up and delivery problem. *European Journal of Operational Research*, 114(3):447–464, 1999.

[56] T. Vincenty. Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations. *Survey Review*, 22(176):88–93, 1975.

[57] P. De Wilde. A neural network model of a communication network with information servers. *NEURAL COMPUTING : APPLICATIONS*, 7(1):26–36, 1998.

[58] L. Yu and H. Liu. Feature selection for high-dimensional data: A fast correlation-based filter solution. In *ICML'03*, pages 856–863, 2003.