DYNAMIC OPTIMIZATION FOR SAME-DAY DELIVERY OPERATIONS

A Dissertation
Presented to
The Academic Faculty

By

Mathias A. Klapp

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of School of Industrial and Systems Engineering

Georgia Institute of Technology

December 2016

DYNAMIC OPTIMIZATION FOR SAME-DAY DELIVERY OPERATIONS

Approved by:

Dr. Alejandro Toriello
Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Alan L. Erera
Industrial and Systems Engineering
Georgia Institute of Technology

Dr. George L. Nemhauser
Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Martin W. P. Savelsbergh
Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Barrett W. Thomas
Tippie College of Business
University of Iowa

Date Approved: November 8, 2016

The line between disorder and order lies in logistics...

Sun Tzu

To Francisca, my partner in life.

ACKNOWLEDGEMENTS

I would first like to sincerely thank my advisors Drs. Alan Erera and Alejandro Toriello. We formed an exceptional team and it was a true privilege for me to take part of it. I am thankful for their genuine interest in my education, constant advice, technical expertise, and experienced opinion. Also thanks to the other members of my Thesis committee for their willingness to help: Drs. George Nemhauser, Martin Savelsbergh and Barrett Thomas. In my opinion, there is no better group of operations researchers to evaluate this research.

Thanks to the other members of the ISyE faculty who taught me and gave me advice: Shabbir Ahmed, Hayriye Ayhan, Santanu Dey, Marc Goetschalckx, Dave Goldberg, Dave Goldsman, Anton Kleywegt, Paul Kvam, Arkadi Nemirovski, Sebastian Pokutta, Andy Sun, and Enlu Zhou. Also thanks for the administrative service of Amanda Ford, Jennifer Harris, Pam Morrison and Mark Reese, and many others.

To all my fellow graduate students and new friends made in Atlanta; you are too many to list all of you. To the ISYE Ph.D. students who came before me for their initial help and advice, to my contemporaries for the mutual support and time spent together doing research and studying, and to the ones who came later for their new energy and constant reminder to graduate. A special thanks to the ISYE graduate soccer team for all those intramural league victories.

To Drs. Juan Carlos Muñoz and Juan Carlos Ferrer for helping to pursue the academic career path.

To the Pontificia Universidad Católica de Chile and CONICYT at the Chilean Ministry of Education for their financial support.

To my patient parents Maria Julieta Belmar and Carlos Klapp for their constant love, to Monica Guitart and Enrique Otero, to the rest of my family and lifelong friends.

Finally, the biggest acknowledgement goes to my beloved wife Francisca. I am extremely grateful for her company and support during this journey.

TABLE OF CONTENTS

Acknowledgments v

List of Tables x

List of Figures xi

Chapter 1: Introduction 1

1.1 Motivation 1

1.2 Objectives 4

1.3 Contribution 5

1.3.1 Chapter 2: The One-Dimensional Dynamic Dispatch Waves Problem 6

1.3.2 Chapter 3: The Dynamic Dispatch Waves Problem for Same-Day Delivery . 7

1.3.3 Chapter 4: Order Acceptance Mechanisms for Same-Say Delivery 8

1.4 Literature Review 9

1.4.1 Vehicle routing problems 9

1.4.2 MDP and approximate dynamic programming 10

1.4.3 Last-mile logistics and dynamic dispatch and delivery problems 11

Chapter 2: The One-Dimensional Dynamic Dispatch Waves Problem 15

2.1 Introduction 15
2.2 The One-Dimensional Dynamic Dispatch Waves Problem 17
2.2.1 Problem Definition 17
2.2.2 MDP formulation of the DDWP 18
2.3 *A Priori* Solutions for the Stochastic DDWP 20
2.3.1 The Deterministic Case 20
2.3.2 The stochastic case and *a priori* policies 23
2.4 Dynamic Policies for the Stochastic DDWP 25
2.4.1 *A Priori*-Based Rollout Policy 26
2.4.2 Approximate Linear Programming for the DDWP 26
2.5 Computational Experiments 30
2.5.1 Design of Instance Set 1: Stationary Conditional Arrival Probability 31
2.5.2 Results for Instance Set 1 31
2.5.3 Design of Instance Set 2: Uniform Arrivals 34
2.5.4 Results for Instance Set 2 35
2.6 Conclusions 36
2.7 Appendix of chapter 2 38
2.7.1 Proof of Proposition 2.4.1 38
2.7.2 Proof of Property 2.4.2 41
2.7.3 Proof of Property 2.4.3 42
2.7.4 Proof of Theorem 2.4.4 43

2.7.5 ALP solution pruning 56

Chapter 3: The Dynamic Dispatch Waves Problem for Same-Day Delivery 58

3.1 Introduction 58

3.2 DDWP problem formulation 60

3.3 The Deterministic DDWP 62

3.4 *A priori* policies 65

3.4.1 Practical considerations when solving the *a priori* model 68

3.5 Dynamic Policies 74

3.5.1 *A Priori*-Based Rollout Policy 74

3.5.2 Greedy *a priori*-based prize-collecting TSP Heuristic 76

3.6 Computational Experiments 77

3.6.1 Design of data sets 77

3.6.2 Set 1: Base experiments 78

3.6.3 Set 2: Second set of experiments 86

3.7 Conclusions 91

Chapter 4: Order Acceptance Mechanisms for Same-Day Delivery 95

4.1 Introduction 95

4.2 Problem formulation 98

4.2.1 System states 100

4.2.2 Actions, transitions and costs 101

4.2.3 Dynamic programming model 102

4.3 The deterministic DDWP and lower bounds . . . 104

4.4 Solution policies for the stochastic case . . . 107

4.4.1 Myopic policy . . . 109

4.4.2 *A priori* policy . . . 110

4.4.3 Myopic policy with fixed *a priori* dispatch . . . 112

4.4.4 Full rollout of the *a priori* policy . . . 113

4.5 A generic heuristic . . . 116

4.5.1 Heuristic acceptance rollout policy . . . 117

4.6 Computational Experiments . . . 118

4.6.1 Design of data sets for base experiment . . . 119

4.6.2 Base experiments . . . 121

4.6.3 Experiment extension #1: Analysis of performance sensibility over processing time . . . 126

4.6.4 Experiment extension #2: Analysis of performance sensibility under varying levels of information dynamism . . . 127

4.7 Conclusions . . . 129

4.8 Appendix of chapter 4 . . . 131

4.8.1 Local Search Neighborhoods . . . 131

4.8.2 Random Destruction . . . 138

4.8.3 Heuristic Solution For the prize-collecting TSP . . . 138

References . . . 146

Vita . . . 147

LIST OF TABLES

1.1 Examples of same-day delivery programs in the U.S. as of October 2016 1

2.1 Lower bounds and heuristic policies' costs computed 30

2.2 Overall performance of heuristics in Instance Set 1 31

2.3 Overall performance of heuristics in Instance Set 2. 35

2.4 Average gap percent reduction of DALP for cases with intermediate variability in Instance Set 2. 36

3.1 Example of a feasible solution s for an instance with 25 probabilistic orders 68

3.2 Heuristic policies computed in our experiments 77

3.3 Average results of heuristic policies 80

3.4 Average $time_{on}$ and $time_{off}$ versus p_{start} and n 86

3.5 Average cost and reduction percentage over AP for each policy under different settings of α 87

4.1 Policies computed in experiments 119

4.2 Average results averaged for each policy 122

4.3 Average performance indicators of RP versus cutoff time (t^{ct}) 129

LIST OF FIGURES

1.1 Description of logistics processes over time for same-day and next-day services . . 3

2.1 Examples of vehicle operations described in the distance versus time graph. 21

2.2 Example of state and action for the deterministic DDWP. 22

2.3 Feasible *a priori* dispatch options. 26

2.4 Percentage gap between PIR lower bound and optimal solution values for Instance Set 1 32

2.5 Average percentage gap between heuristic solution costs and optimal costs for Instance Set 1 33

2.6 Average percentage gap between heuristic solution costs and lower bound for Instance Set 1 33

2.7 Average percentage gap between heuristics cost and lower bound in Instance Set 2. 35

2.8 Network Structure in (Z,Y)-domain 48

2.9 Network for i^{th} order subproblem. 50

2.10 Example of a convex combination of three operations in i^{th} order subproblem. . . . 53

2.11 Example of a convex combination of two operations where subproblem for order i is not additive in the argument (Z,Y). 54

2.12 Same example with two operations where subproblem for order i is additive in the argument (Z,Y). 55

3.1 Example of a cut operation where a new dispatch profile is created (dashed flow) from an existing one (continuous flow) by cutting a route into two. 71

3.2 Example of a merge operation where a new dispatch profile is created (dashed flow) from an existing one (continuous flow) by merging two dispatches into one. 72

3.3 Example of an exchange operation where a dispatch gives one wave to its successor (dashed flow). 72

3.4 Example of a Start operation where the first dispatch is enlarged/reduced (dashed flow). 73

3.5 Distribution of gap over all instances 81

3.6 Average gap versus n, p_{start}, p_{out}, and σ 82

3.7 Average fill rate versus n, p_{start}, p_{out}, and σ 83

3.8 Average number of vehicles dispatched versus n, p_{start}, p_{out}, and σ 85

3.9 Average number of waves per vehicles dispatch versus n, p_{start}, p_{out}, and σ 85

3.10 Average number waves waited before initial dispatch versus n, p_{start}, p_{out}, and σ . 86

3.11 Average results for RP under different settings of α 88

3.12 Example of two different solutions to the same instance realization. The left one minimizes total cost and the right one minimizes lost penalties first. Green orders are served and red ones are left unattended. Each order's ready wave is labeled at its node and route dispatch/return waves are shown in the upper-left corner. 89

3.13 Pareto charts showing each policy's average fr and $duration/order$ for different settings of p_{start} 90

3.14 Pareto charts showing each policy's average fr and $duration/order$ for different settings of p_{out} 91

3.15 Pareto charts showing each policy's average fr and $duration/order$ for different settings of σ 92

4.1 Order disclosure time and order ready time for an accepted order. 96

4.2 Flowchart of system transitions and actions in state (t,a,w) for the DDWP-IA . . . 102

4.3 Evolution of the orders disclosed over time ($N_i(t)$) and orders ready for each dispatch wave ($n_{i,w}$) for a particular realization (ω) of arrivals at node $i \in I$. 104

4.4 Expected orders realized at time t ($\mathbb{E}(N_i(t))$) and expected orders ready for dispatch at wave w ($\bar{n}_{i,w}$) at a particular node $i \in I$ for a particular Poisson arrival process. . . 111

4.5 Average fill rate (fr) and distance per order for RP policy accepted versus number of nodes (n) 123

4.6 Average time per acceptance decision and average time per dispatch decision versus number of nodes (n) 124

4.7 Average $cost/request$ versus number of nodes (n) and online arrival intensity (λ^*) . 124

4.8 Average $cost/request$ versus offline order probability (p_{start}) and online arrival intensity (λ^*) 125

4.9 Average number of routes and waves waited before dispatch ($iWait$) for RP policy versus number of nodes (n) and online arrival intensity (λ^*) 126

4.10 Average $cost/request$, fr, and $distance/order$ versus order processing time (p) . . 127

4.11 Average gap^{LB} versus order processing time (p) 128

4.12 Example of request arrivals re-scaled from $t^{ct} = 252$ to $t^{ct} = 126$ and to $t^{ct} = 378$. 128

4.13 Example of a cut operation where a new dispatch plan is created (dashed flow) from an existing one (continuous flow) by adding an extra return to the depot. . . . 133

4.14 Example of a merge operation where a new dispatch plan is created (dashed flow) from an existing one (continuous flow) by removing one return to the depot and merging two dispatches. 134

4.15 Example of an Insert operation where an new dispatch is inserted launching previous routes a wave earlier (dashed flow). 135

4.16 Example of a Delete operation where last dispatch is deleted pushing preceding ones later (dashed flow). 135

4.17 Example of an Enlarge operation where the last dispatch is enlarged and launched a wave earlier pushing the previous dispatch earlier too (dashed flow). 136

4.18 Example of a Reduce operation where one dispatch is reduced and launched one wave later pushing the previous dispatch later too (dashed flow). 137

SUMMARY

Same-day delivery (SDD) is a distribution service where consumers place orders online on the same day that these are processed and delivered to the customer's location. This service enhances consumer satisfaction and, therefore, has been implemented by multiple e-commerce companies. However, the increasing competitiveness within the e-commerce sector and the complex operation involved puts downward pressure on last-mile logistics costs.

Providing SDD services requires two core logistics processes: (1) order management at the stocking location, including request acceptance and processing; and (2) order distribution from the stocking location to final delivery locations, including order dispatch and delivery via vehicle routes. Unlike other last-mile services, in SDD processes are highly interrelated, simultaneously executed, and have a high degree of information dynamism, meaning customer orders are incrementally revealed during the operating period. While new technologies allow us to modify and adapt operations in real time, traditional decision support tools such as deterministic optimization are not adequate for this challenge; there is a need for dynamic decision support tools for SDD.

In this Ph.D. thesis we formulate and solve dynamic optimization problems to improve the operation of SDD systems, integrating both order management and distribution. We also study how our approaches perform over an extensive family of computationally simulated instances and provide managerial insights for SDD practitioners, including structural solution properties that any common SDD service should have, the trade-offs between common SDD objectives, and the logistics cost of operational constraints in SDD.

The thesis consist of four chapters. The first one is an introduction where we describe the context and challenges in same-day delivery systems, we present and discuss our thesis objectives, a literature review, and a summary of our contributions for each following chapter. In the second and third chapters we study the tradeoffs within SDD logistics distribution processes by defining the Dynamic Dispatch Waves Problem (DDWP), which considers a single delivery vehicle operat-

ing over a day partitioned in a discrete set of dispatch decision epochs (waves). At any wave, the vehicle (if available) can wait for one wave, or can be loaded and dispatched to serve a subset of disclosed requests. We study how to dynamically select the dispatch times from the depot and the orders to be loaded in each dispatch, and how to design the delivery routes so that we minimize vehicle travel costs plus penalties for unattended requests at the end of the day.

We begin the work on the DDWP in the second chapter by studying the case where customer destinations are placed over a line segment with the depot at one end. This assumption simplifies routing decisions to focus on the vehicle dispatch and order selection aspects. We efficiently solve the deterministic problem, develop two tractable ways to get lower bounds, provide an optimal *a priori* solution and design multiple dynamic policies.

We extend these results in the third chapter to a generalized model where customer locations can have any network topology, adding vehicle routing decisions to the already present order selection and dispatch problems; this makes the evaluation of an action's cost NP-hard. Therefore, we design heuristic dynamic policies and show their benefits over optimal *a priori* ones with computational experiments. Also, we analyze the tradeoff between two common objectives in SDD: minimizing total costs, including vehicle travel time, versus maximizing order coverage. We empirically find that there are fundamental differences in the solution's structure for both cases in terms of vehicle dispatch frequency, route duration, and initial waiting time at the depot, that one should expect significant sacrifices in travel costs and routing efficiency in order to maximize request coverage, and that the cost of an additional order served becomes more expensive as order coverage increases.

In the fourth chapter we formulate a new model that integrates both order management and distribution decision-making including an immediate request acceptance mechanism. When a delivery request arises, a decision is made immediately to accept (offer service) or reject (with a penalty). All accepted delivery requests are included in dynamically-updated dispatch plans that serve each request by the end of the operating day. This differs from the DDWP model, where

the acceptance of a newly arrived request can be postponed until the order gets loaded into the vehicle for dispatch. We develop a framework for dynamic decision policies for such systems, where a system state is maintained that includes a feasible dispatch plan (with potentially multiple planned trips) serving all accepted delivery requests along with a set of potential future delivery requests that have not yet realized. This dispatch plan is used to guide order acceptance decisions, and it is update dynamically when new information is available. This type of operation imposes additional constrains to decision making and may increase the solution's cost; we experimentally estimate this cost increase, extend our heuristic policies to this case and provide a meta-heuristic to have a fast order acceptance process. We also study the interactions between order warehouse management and distribution, and accepted requests are not available for immediate loading and dispatch, and instead must wait to be processed (picked and packed) before they can be loaded into the delivery vehicle. We estimate the cost sensitivity of the distribution system over different values of orders processing times.

CHAPTER 1

INTRODUCTION

1.1 Motivation

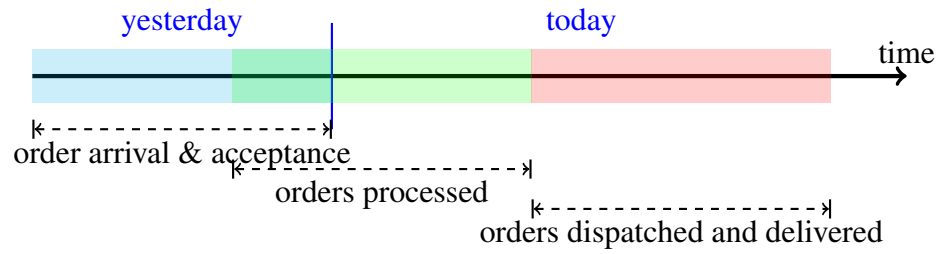
We define **same-day delivery (SDD)** as a business-to-consumer distribution service that processes, dispatches and delivers orders to the customer's location on the same day the order from the customer is placed online. SDD services are increasingly being offered by retailers and logistics companies in the U.S. and are an example of the service improvements within e-commerce that helped sales grow 15.8% annually and represent over 7.5% of U.S. retail industry sales as of the second quarter of 2016 [29]. We present examples of these SDD services in Table 1.1; Amazon has the biggest deployment, covering 27 metropolitan areas, and has also implemented "Prime Now", an even faster one-hour delivery service. We identify two classes of SDD providers: retailers offering items primarily from owned stocks (in distribution centers or retail stores) such as Amazon and Walmart, and logistics providers serving as intermediaries that pick up packages from stocking locations and deliver them to customers, such as Google, Deliv and Instacart as well as USPS, FedEx and UPS.

Table 1.1: Examples of same-day delivery programs in the U.S. as of October 2016

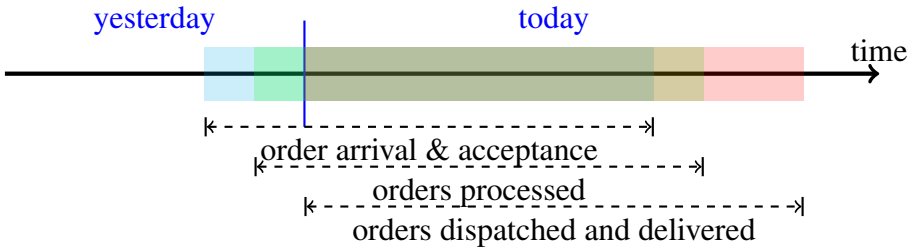
Service	**What**	**Number of US metro areas**
Amazon SDD	items from warehouses	27
Instacart	personal grocery shoppers	23
Deliv	items from associated retailers	17
Amazon Prime Now	items from warehouses	14
Google Express	items from associated retailers	6
Walmart	items from its physical stores	6

SDD services are designed to satisfy the increasing demand for "instant gratification" when ordering consumer products online, and at the same time to discourage physical store visits as well as slower delivery services, *e*.g., two-day (TDD) and next-day delivery (NDD) [44, 71]. However, the retail sector is extremely competitive and operates with very low margins, putting downward pressure on logistics costs. Consider Amazon's 2015 annual report [4], which states that its fulfillment cost accounts for 12.5% of its $107 billion worldwide sales, while its operational margin represents only 2.1%; a small increase in Amazon's logistics costs can eliminate profits. This competitive context is especially true for last-mile distribution; unlike most other parts of the supply chain, it does not scale well and offers low freight consolidation opportunities due to the large variety of SKUs and small volume handled per delivery.

To analyze home delivery services, we need to discuss its two core logistics processes: (1) **order management** at the stocking location (depot), including request acceptance and order processing; and (2) **order distribution** from the stocking location to customers' locations, including order dispatch and delivery via vehicle routes. Request acceptance refers to the promise of service to the customer; order processing indicates the preparation of the order before dispatch; dispatch decisions select the dispatch times and the orders to serve in a delivery vehicle trip; and delivery routing decisions must select the order of customer visits for each vehicle route dispatched. In slower delivery services these processes usually occur sequentially over time and most orders to be dispatched and delivered in a day have already been accepted and processed before the distribution starts; in Figure 1.1a we present a typical NDD setting. The operation dynamics are different for SDD and, as depicted in Figure 1.1b, the executions of these processes overlap in time and should be planned simultaneously under a high **degree of dynamism**, meaning that only a small fraction of all relevant information needed to plan the daily operation is known prior to the initial decision epoch. Missing information is revealed incrementally during the operating period after the execution starts. Traditional decision support tools such as deterministic optimization are not adequate for this challenge. Moreover, the appearance of new technologies, such as immediate communi-



(a) Next-day delivery



(b) Same-day delivery

Figure 1.1: Description of logistics processes over time for same-day and next-day services

cations and tracking devices, *i*.e., RFID chips, GPS systems and internet-connected smartphones, allows us to observe and adapt operations in real time; there is a need for dynamic decision support tools able to continuously replan operations.

A solution to this dynamic problem is a **dynamic policy** that determines decisions depending on the information state at each decision epoch. In contrast, simpler ***a priori* policies** specify certain decisions in advance, and may allow simple changes over time via pre-established recourse rules. Furthermore, one has to consider how these solutions incorporate potential future states of the system, and it may be insufficient to consider **myopic policies** that take actions reacting only to newly disclosed information. If probabilistic knowledge is available regarding future uncertainties of the order realization process, we may outperform myopic solutions by designing **proactive policies** that incorporate past as well as future information into decision making.

In addition, routing problems that arise from SDD distribution have distinctive characteristics that make them unique compared to classic vehicle routing problems (VRP), and yield challenging

research problems even without demand uncertainty. First, SDD is heavily constrained by available time resources, such as driver's workday limits and the daily operation, rather than vehicle volume and weight capacities that suffice in other cases. Second, ours is a delivery problem for customized orders to be picked up at a depot and delivered to the customer's location, rather than a pick-up problem at the customer or a generic commodity delivery that can be pre-loaded into vehicles before its customer order realizes. This difference implies specific order release times at the depot determining the earliest possible time to load the order into a vehicle; this differs from typical service time window constraints at customers that govern many VRP operations. The delivery setting also removes the ability to modify a route and insert more customers into it while the vehicle is enroute. Third, in SDD a vehicle can be dispatched multiple times from the depot in one day and all routes executed by the same vehicle cannot overlap in time; this differs from classic VRP models that execute one route per vehicle and period. Finally, vehicle dispatch operations in SDD are also frequently organized such that a fixed and finite number of possible vehicle dispatch times are used from each depot. This type of organization is often best due to many factors, including constraints associated with driver shifts and efficiencies gained by organizing warehouse activity using wave picking. We will refer to these feasible dispatch times as *dispatch waves.*

1.2 Objectives

Our main goal is to formulate mathematical optimization models and algorithms that support SDD operations and recommend decisions for both the order management and distribution processes. Specifically, we establish four research objectives to address:

1. To model vehicle dispatch decisions and study their tradeoffs within SDD distribution.

2. To model the stochastic and dynamic request arrival process that determines order realizations.

3. To integrate vehicle dispatch and delivery routing decisions within SDD distribution.

4. To model and integrate request acceptance decisions with order dispatch and delivery decisions.

As a secondary objective we are interested in providing managerial insights for SDD practitioners, to develop structural solution properties that any common SDD operation should have, to compare solutions and performance between different SDD objectives, to estimate the marginal value of dynamic and proactive policies versus *a priori* and myopic ones, and to measure the cost incurred by imposing typical operational constraints in SDD.

1.3 Contribution

This Thesis is organized as follows. The remainder of Chapter 1 provides a literature review. In Chapter 2 we present completed work [44] that addresses the first and second research objectives; in Chapter 3 we present work under review [43] that builds on the results of Chapter 2 and addresses the third research objective. In both, we state, solve and study the single-vehicle *Dynamic Dispatch Wave Problem* (DDWP) under different problem settings. Finally, in Chapter 4 we develop the DDWP with Immediate order Acceptance (DDWP-IA), which deals with the fourth research objective in Section 1.2; it also revisits the second one.

The DDWP studies SDD distribution and the interaction between order dispatch and delivery decisions. It models a depot where orders realize dynamically throughout the day with stochastic release times. At any decision epoch (**wave**) the system state has information about (1) a set of *open* orders, which are ready to be delivered to the customer, and (2) probabilistic information describing *potential* orders that may realize before the end of the day. The operator chooses whether to wait for one wave, or to dispatch a single delivery vehicle loaded with a subset of open orders. The vehicle can be dispatched multiple times throughout the day, and each route defines a dispatch duration that impacts the vehicle's future availability. The objective is to minimize expected vehicle travel plus penalties for unattended orders at the end of the day.

1.3.1 Chapter 2: The One-Dimensional Dynamic Dispatch Waves Problem

We begin the work on the DDWP in Chapter 2 by focusing on dispatch decisions and study the simplified case where customer destinations are placed over a real line segment with the depot at one end. We develop two tractable lower bounds for the optimal expected cost: (1) A perfect information relaxation (PIR) and an (2) approximate linear programming (ALP) bound; each one also gives an approximation for the cost-to-go value useful for approximate dynamic programming (ADP). The PIR is based optimizing the DDWP for each scenario realization knowing all uncertain information in advance. We solve this deterministic problem by establishing basic properties of an optimal solution used to build a fast dynamic program. The ALP bound is based on generating a suboptimal solution for the dual LP formulation of the DDWP's Markov decision process (MDP) model in which the cost-to-go values for each state are the variables of an LP of exponential size. We derive a polynomially sized equivalent problem that recovers the optimal solution for deterministic instances.

We develop heuristic solution policies based on ADP and test them in two sets of simulated experiments. First, we provide an optimal *a priori* solution by reducing the problem to an equivalent deterministic one. Experimentally, its performance is good in both sets of experiments with an average cost within 9.2% and 5.6% of the best lower bound. Nevertheless, we prove that the benefit of a fully dynamic policy can be unbounded versus the optimal *a priori* policy in the worst-case scenario. Accordingly, we propose two dynamic policies that differ by the nature of the approximate cost-to-go function used to choose an action at a given state of the system. The first one rolls out the *a priori* policy and cuts the gap respectively by 28.7% and 20.6%, while the second one is an ALP-based approximation that provides an extra improvement in gap reduction of 2.6% and 4.9%, respectively. We find that the maximum benefit of our dynamic policies versus the optimal *a priori* one is for order sets of around 20 to 50. Many same-day delivery applications might expect similar daily order volumes. Also, dynamic policies' benefits decrease as orders become more

likely to appear at the start of the horizon (less information dynamism). It is precisely in the most dynamic and uncertain environments that our policies offer the most benefit.

1.3.2 Chapter 3: The Dynamic Dispatch Waves Problem for Same-Day Delivery

In Chapter 3 we generalize the DDWP to a model where customer locations can have any network configuration. This integrates vehicle routing decisions to the order dispatch problems and makes the design of the least cost vehicle route that covers a specific set of customers an NP-hard problem. Accordingly, we design heuristic dynamic policies motivated by the solution methodology from Chapter 2. First, we formulate an arc-based integer programming (IP) model for the deterministic case that provides a PIR for the stochastic-dynamic case. The IP is a prize-collecting version of the VRP where vehicle routes cannot overlap in time and where customers have earliest dispatch times from the depot, sometimes called release dates in the literature; we also design a local search heuristic to speed up the search for good solutions. We then exploit the deterministic IP solution to design an *a priori* policy solved via a transformation to a deterministic instance, and implement its corresponding roll out to get a dynamic policy. We also provide a computationally faster dynamic greedy policy that only plans proactive decisions before the operation starts.

We show the benefits of dynamic policies over the optimal *a priori* solution with computational test instances that suggest that their marginal cost reduction averages 9.3%, mostly due to a 5.4% increase in the percentage of orders covered, which is highly desirable for SDD services. The cost reduction and order coverage increase are especially high when the instance dynamism and variability is high.

Moreover, we analyze the tradeoffs between two common objectives in SDD: minimizing total costs, including vehicle travel time, versus maximizing order coverage. We empirically find that there are fundamental differences in the solution's structure for both cases in terms of vehicle dispatch frequency, route duration, and initial wait at the depot, that SDD practitioners should expect significant sacrifices in travel costs and routing efficiency to maximize order coverage, and

that the distance cost of an additional order served becomes significantly more expensive as order coverage increases.

1.3.3 Chapter 4: Order Acceptance Mechanisms for Same-Say Delivery

In this chapter we formulate and solve the DDWP with Immediate Acceptance (DDWP-IA), a model that integrates both order management and distribution processes by accepting or rejecting each request immediately after its realization. If an order gets accepted, its delivery should be guaranteed; otherwise, it is permanently lost. An order distribution system operates simultaneously with order acceptance and dynamically chooses at waves whether to dispatch or not a single vehicle (if available) loaded with a subset of accepted orders ready for service. We also study the impact of order processing times at the stocking location. Accepted requests are not available for immediate loading and dispatch, and instead must wait to be processed (picked and packed) before they can be loaded into the delivery vehicle. We formulate the DDWP-IA as a semi-Markov decision process model that assumes dynamic arrival of requests over a continuous time horizon until a cutoff time after which no more orders are accepted. We develop a framework for dynamic decision policies for such systems, where a system state is maintained that includes a feasible dispatch plan (with potentially multiple planned trips) serving all accepted delivery requests along with a set of potential future delivery requests that have not yet realized. This dispatch plan is used to guide order acceptance decisions, and it is update dynamically when new information is available. We extend our IP model to the deterministic DDWP-IA case and develop and test methods for determining an initial optimal *a priori* dispatch plan, and for updating the plan via a roll-out procedure that includes heuristic approaches designed to speed up the update step. Our methods are compared against two common-sense myopic benchmarks and a perfect information lower bound in a computational study using a family of simulated instances. We show via experiments that the cost-per-request of the best benchmark is 9.7% higher than our proposed best dynamic policy, which has a gap of 21% over the perfect information bound. We also estimate that a cost increase

of 4.4% results when imposing immediate order acceptance on same-day delivery systems, and we study cost sensitivity to different assumptions on request processing times within the facility; our experiments show that when processing times take 18% of the daily operation the cost per request can increase by 116% versus a similar system where orders become immediately available for dispatch after they realize. Moreover, our dynamic policy's gap over the deterministic bound becomes tighter as processing times increase.

1.4 Literature Review

The literature review is divided in three sections. First, we cover classic deterministic and dynamic-stochastic vehicle routing literature. The second section covers approximate dynamic programming techniques and the third one specifically covers last-mile logistics, dynamic dispatch and delivery problems.

1.4.1 Vehicle routing problems

The deterministic DDWP qualifies within the large family of routing problems; these problems have been extensively studied by operations researchers. The Vehicle Routing Problem (VRP) and Traveling Salesman Problem (TSP), are the basic building blocks for routing problems; see *e*.g., the texts [8, 39] for the TSP, and [28, 36, 67] for the VRP. In particular, our problem is related to the Pickup and Delivery Vehicle Routing Problem (PD-VRP) [58] with all orders' pickups at the depot. Another directly related problem is the VRP with release dates (VRP-rd) that considers a delivery vehicle transporting orders from a depot with release times specifying the earliest time when the order can be loaded into a vehicle and dispatched. In [9] the authors study simplified versions of the problem with requests located on the one dimensional line. In [24], the authors extend the work on the VRP-rd to a general network topology with service time windows (TW) and capacitated vehicles. In addition to being deterministic, its objective differs from the DDWP because it focuses on minimizing travel cost subject to covering all orders. Our deterministic

DDWP selects the orders to be served during the day to maximize profit, which is important when a system operates with a fixed fleet of vehicles and limited time resources; this generalizes the Prize-Collecting Traveling Salesman Problem (PCTSP) [13]. Also, the DDWP does not include TWs to consider full routing flexibility in vehicle dispatch decisions.

Dynamic and stochastic VRPs are problem extensions where some parameters are unknown during planning and/or operations. The simplest stochastic VRP problems are *a priori* optimization models, where fixed operating (recourse) rules are used to modify the solution during operation; see [22, 28, 34] for recent surveys. Dynamic and online VRPs are problems where information is revealed over time during the operating period, and routing and scheduling decisions are updated in response; see [41, 47, 53, 56, 64]. Different stochastic and dynamic VRPs focus on uncertainty in different sets of parameters. Some examples are the VRP with stochastic customer demands [2, 17, 37, 52, 61], the VRP with stochastic travel times [27, 42, 46, 48, 49, 50, 63, 65] and the VRP with probabilistic customers (VRP-PC) where orders realize over time with uncertainty; see [23, 31, 40, 45, 70] for examples of *a priori* models and [3, 5, 6, 7, 14, 72] for dynamic models. Our DDWP is related to VRP-PCs but differs from those considered in this literature, which are designed primarily for pick-up operations. In pick-up operations, a vehicle route can be modified after dispatch, both by re-routing existing customers and adding new customers. In contrast, our order delivery model assumes customized orders to be delivered from a depot to the customer; it is not possible to add new customers to a route that has been previously dispatched without returning the vehicle to the depot.

1.4.2 MDP and approximate dynamic programming

The DDWP is also a dynamic and stochastic problem and can be modeled as Markov Decision Processes (MDP); see [57, 15]. Optimizing MDPs that arise from routing problems is usually intractable due to the *curse of dimensionality* originating from large state and/or action spaces. However, there are several computationally efficient lower bounding techniques; see [30] for a

survey. An example of a tractable bounding procedure is the *a posteriori* bound [61] or Perfect Information Relaxation (PIR) [19] that disregards the solution's "non-anticipative" dynamics and computes the optimal deterministic cost for each possible realization of the random parameters; typically, Monte Carlo sampling is used to estimate this value. Another bounding technique is the Approximate Linear Programming (ALP) method [32, 60] that looks for suboptimal solutions of the MDP's dual LP formulation, in which the cost-to-go values for each state are the variables of an LP that has exponentially many variables and constraints. The fundamental idea is to eliminate the exponential number of state variables by enforcing a dependence on a previously determined low-dimensional set of basis functions. Moreover, its solution can be used as a cost-to-go approximation in heuristic policies. The ALP approach has been successfully applied in stochastic routing before, *e*.g., [1, 65].

In terms of approaches, the curse of dimensionality necessitates approximate dynamic programming (ADP) solution techniques, *e*.g., [35, 37, 52, 55, 61]. One widely used ADP method is to develop approximations for the optimal cost-to-go function and use it to select an approximately optimal action at any encountered state. The texts [15, 55] thoroughly discuss ADP. Rollout algorithms [16, 38] have been widely applied for stochastic routing models, *e*.g., [37, 61, 65].

1.4.3 Last-mile logistics and dynamic dispatch and delivery problems

We now refer to some of the most relevant problems to last-mile delivery in the literature related to order acceptance, dispatch and delivery. Challenges related to and last-mile and city logistics are discussed in recent surveys [25, 59].

The dispatch component of SDD relates to the Order Batching Problem (OBP) [33, 51], which determines an optimal assignment of pick orders to batches and the pick tour sequence for each batch in warehouse operations. The Dynamic Order Batching Problem (DOBP) is an extension with orders arriving dynamically while the decision maker processes previous orders; see [20]. Recently, [26] present an analytical model to determine the timing and the number of batches in

an order fulfillment system. To the best of our knowledge, none of these models allow a detailed selection of the orders within a batch, and solely contemplate threshold rules such as consolidating the batch when a number (to be determined) of orders have arrived.

The Dynamic Multiperiod Routing Problem (DMPRP) introduced in [6, 7, 72] models dispatch operations in a depot over a time horizon partitioned in days. Dynamic requests with service time windows are disclosed over time, and each day the decision maker chooses which customers to serve or postpone, and the corresponding vehicle routes. The extension in [3] covers the dynamic-stochastic case where probabilistic information about future request arrival times and service time windows is available. In this case, there is a fleet of capacitated vehicles available at the depot. The problem is solved heuristically, running a daily prize-collecting VRP over the set of open orders; the prize for each order is set according to an increasing function of proximity to the service deadline and a decreasing function of geographic proximity to probabilistic future orders. Note that this model assumes that a vehicle is available for dispatch at most once at every decision epoch (day). In SDD operations, it may be sensible to dispatch a vehicle multiple times per day and each route may span multiple periods within the day; in this case, if a vehicle is dispatched at period t on a route of duration x, it cannot be dispatched again until a time after its return to the depot at $t+x$. A recent extension in [68] considers a single vehicle operation and order arrivals over a time horizon with multiple days (3-10) and multiple time periods per day (360 minutes). At the start of a day, a vehicle route is designed visiting accepted orders carried on from previous days. A decision is triggered upon the arrival to a customer location and the dispatcher chooses whether to accept each newly arrived requests for SDD injecting it into the route or to postpone it for the next day. The modified route determines where the vehicle travels next and the objective is to maximize the number of requests accepted for SDD. Unlike the DDWP, it does not consider order pick-ups at the depot and vehicle travel costs in the objective function. The problem is solved via approximate value iteration (AVI) methods [55] to estimate the value function of an MDP via offline simulation. To deal with the remaining curses of dimensionality, the authors restrict the

feasible set of actions to node insertions and aggregate the state space into a vector consisting only of temporal attributes, disregarding geographic customer information.

Perhaps the closest model in the literature to the DDWP is the Same Day Delivery Problem for Online Sales [71], which considers a model for an order delivery system with dynamic order arrival and a fleet of vehicles, each performing multiple trips per day. However, there are fundamental differences in assumptions and solution methodologies between this setting and the DDWP. The model in [71] assumes narrow request service time windows, *i*.e., one or two hours within a 10-hour day operating period; these tight time constraints correspond to rapid delivery settings, like Amazon's "Prime Now" service. Serving customers is the priority in this model, and the objective is to maximize the number of orders served, while ignoring vehicle travel costs. Alternately, the DDWP that we consider generalizes the objective to consider a weighted combination of costs for not serving customers and vehicle travel costs. An additional difference is the choice of solution approach; [71] finds solutions by adapting a scenario-based-planing heuristic (see [14]), which uses a consensus function to choose a best plan among those developed for a set of simulated scenarios at each decision epoch. A recent extension of this problem discussed in [69] explores the value of planning preemptive vehicle returns to the depot. Under this setting, the dispatcher is allowed to abort the current route by inserting a detour to the depot when a new route dispatched later that considers newly disclosed requests produces more value than the remaining portion of the current route. The problem is solved using an extension of the methodologies in [68] using AVI, insertion heuristics for route updates, and state space aggregation. Unlike this approach, the DDWP does not consider preemptive returns to the depot after a vehicle has been dispatched. Instead, it proactively plans returns to the depot in an optimal *a priori* solution. Our experiments for the DDWP suggest that the value of preemptive returns is marginal when returns to the depot are planned using probabilistic information. These benefits may be further reduced, if we consider order processing times at the depot that rule out an immediate service of newly disclosed orders. The method proposed in [68] outperforms greedy order acceptance rules, but there is no comparison

to lower bounds. The value lost by disregarding geographic locations in the value function and by restricting the action set is also unclear.

Regarding request acceptance, a related problem to the DDWP is the Home Delivery Problem (HDP) discussed by [21]. It develops a real-time model for last-mile grocery delivery that focuses on the interactions between order promise of delivery times and order dispatch. They dynamically determine whether to accept an order and, if so, set a time slot for order delivery. The objective is to maximize the total profit resulting from service revenues minus costs. The problem is solved via insertion heuristics of new orders in available spaces of the current routes. This problem is motivated by NDD or overnight delivery, and its main drawback for SDD is that all orders are received prior the execution of the delivery service.

Another example of closely related work is found in [12, 11], where the authors develop a VRP model with time windows and multiple delivery routes per vehicle. Small and highly constrained problem instances are solved via an exact optimization approach, while larger instances are solved with an adaptive large neighborhood search (ALNS) heuristic. The optimization model objective is to maximize the number of served orders and travel costs in hierarchical order, while multiple trips per vehicle are induced by the inclusion of a route duration constraint; results show that this constraint leads to short routes, with an average of four customers served. This model is useful for settings in which the time between order placement and vehicle dispatch must be very short, *e*.g., in the dispatch of perishable goods such as meals. A fundamental difference between this model and the DDWP is that we treat route duration as an unconstrained decision. In [10], the authors extend the model to a dynamic setting that incorporates immediate order acceptance decisions made using a scenario-based planning approach that estimates the insertion profit via ALNS for multiple simulated future order scenarios.

CHAPTER 2

THE ONE-DIMENSIONAL DYNAMIC DISPATCH WAVES PROBLEM

2.1 Introduction

We formulate the **Dynamic Dispatch Waves Problem (DDWP)** as a Markov Decision Process (MDP). The DDWP models the dynamics of a single dispatch facility (depot) where customer order requests arrive dynamically throughout an operating day. At any decision epoch, which we call a *wave*, the logistics operator maintains a set of known open requests with known delivery destinations and a set of potential requests that may arrive before the end of the day. At each wave, the operator decides whether or not to dispatch vehicles loaded with known orders, and the vehicle routes for dispatched orders. The objective is to minimize expected operational costs and expected penalties for unserved open requests at the end of the day. Such penalties could represent the cost of direct dispatch or revenue lost due to unserved customers.

We study the interaction between two important decisions in SDD distribution systems: dynamic dispatch and vehicle routing. Dispatch decisions refer to selection of the times at which vehicles are dispatched and the orders that they deliver, while vehicle routing decisions refer to the sequences of deliveries for each dispatched vehicle. Two fundamental tradeoffs exist. First, there is a tradeoff between waiting and dispatching a vehicle to serve requests. When a vehicle is dispatched, the queue of open requests is reduced but an opportunity to serve future requests during the route is missed. On the contrary, when an available vehicle is not dispatched, we reduce the time remaining in the operating day and potentially increase the likelihood that future requests cannot be served. Second, there is a tradeoff between dispatching longer, time consuming vehicle routes versus shorter ones. On one hand, a route serving many requests uses more total travel time, and therefore keeps the vehicle away from the depot longer, but requires less time per customer

visited. On the other hand, a shorter route uses more time per customer, but returns to the depot faster and enables the vehicle to be reused sooner for future orders.

To simplify the vehicle routing decisions, this paper focuses on problem instances where a single vehicle is available to make deliveries to customer locations on the positive real line with the depot as the origin; travel times and vehicle operating costs are proportional to distances between points.

We consider the following to be our main contributions.

1. We formulate the DDWP to capture the basic aspects of dynamic dispatch, order selection, and routing decisions for same-day delivery.

2. We develop an approach for determining optimal *a priori* solutions to the stochastic one-dimensional variant by reducing this problem to an equivalent deterministic problem where all customer request arrival times are known in advance.

3. We show that, although *a priori* policies work well in practice, there exist problem instances for which these solutions are arbitrarily worse than optimal dynamic policies. Accordingly, we provide two schemes to obtain dynamic policies for the one-dimensional problem. The first is a rollout of the *a priori* policy, and the second is an approximate dynamic programming approach that uses an approximate linear program (ALP) to approximate the cost-to-go function. We empirically show the benefits of dynamic policies with computational experiments over two sets of representative instances.

The remainder of the paper is organized in the following manner. Section 2.2 formulates the model, and Sections 2.3 and 2.4 respectively cover *a priori* and dynamic policies. Finally, Section 2.5 outlines the results of a computational study, and we conclude with Section 2.6. An online supplement contains all technical proofs not included in the body of this document.

2.2 The One-Dimensional Dynamic Dispatch Waves Problem

2.2.1 Problem Definition

Consider a dynamic dispatch and routing problem for a single vehicle operating over a fixed-duration operating period (*i*.e., a day). The vehicle is dispatched from a depot, located at one end of a line segment, to serve a set of customer delivery requests. After completing a route, the vehicle returns to the depot and may be dispatched again until the end of the operating period. At each decision epoch, the vehicle (if available) may be dispatched to serve any *open* customer requests, those that have arrived and are ready for dispatch. In addition to information about open orders, probabilistic information describing unknown future order requests is also available. The objective is to minimize vehicle operating costs and penalties for unserved requests. We consider a specific class of problems of this type:

1. Let $\mathscr{W} := \{1,\ldots,W\}$ be the set of *waves* (decision epochs) during the operating period, where waves are counted backwards so that the waves number represents the "waves-to-go" before $w = 0$, the deadline for the vehicle to return to the depot. Let $\mathscr{W}_0 = \mathscr{W} \cup \{0\}$.

2. Let $N := \{1,\ldots,n\}$ be the set of all *potential* customer requests $i \in N$, where each i is characterized by: (1) a known destination represented by a round-trip travel time of d_i from the depot; (2) a penalty cost $p_i > 0$ that must be paid if the request arrives but is not served by $w = 0$; and (3) a random arrival time τ_i drawn from a request-dependent distribution with support $\mathscr{W} \cup \{-1\}$, where -1 indicates "no arrival". We assume that order arrival times are independent between different customers, which is reasonable, since one customer's behavior should not affect the others'. Let N be ordered such that $d_i \leq d_j$ for $i < j$. Note that our probabilistic model enumerates all possible request arrivals. Another way of modeling this problem is to define a fixed set of locations at which orders appear with potentially multiple arrivals per wave. This alternative probabilistic model is implicitly captured in our setting

by adding several requests with an identical customer location.

A vehicle located at the depot at any wave $w \in \mathscr{W}$ can be dispatched to serve some subset S of the set of open (revealed and unattended) requests $R \subseteq \{i \in N : \tau_i \geq w\}$ at wave w. Once a vehicle leaves the depot at wave w, it cannot serve any request arriving at $\tau < w$ until it returns for reloading; we assume that once dispatched, a vehicle must finish its route and cannot serve any other request until it returns for reloading. Serving request set S requires time, and we assume that no additional service time is required beyond vehicle travel time. Given request locations along the line, the time required by the vehicle to serve S is then $d_S := \max_{i \in S} d_i$; we assume the vehicle operating cost for this dispatch is αd_S, where α is the cost per unit distance. A vehicle dispatched at w returns to the depot at $w - d_S$. S is therefore constrained by $d_S \leq w$, but we assume no other constraints on S, such as capacity, consistent with motivating SDD applications where time is the binding resource. Total system cost is measured by the sum of the vehicle operating costs over all dispatches plus the sum of the penalties p_i for all $i \in R$ at the terminal wave ($w = 0$).

For purposes of analysis, we suppose in this paper that the d_i values are scaled such that they are all integer, and time between consecutive waves is constant and equal to the time required for the vehicle to complete a round trip with dispatch travel time 1.

2.2.2 MDP formulation of the DDWP

We now formulate an MDP for the DDWP. At each wave $w \in \mathscr{W}_0$, the system state is given by $(w, R, P) \in \mathscr{S}$, where $\mathscr{S}$ is the state space, w represents the waves-to-go, R is the set of open requests, and P is the set of remaining *potential* requests with an unknown arrival time $\tau < w$. Requests not in R or P have been already served and so the pair (R, P) belongs to the set $\Xi := \{(R, P) : R \cup P \subseteq N, R \cap P = \emptyset\}$. The maximum number of waves and the three possible states for each requests (open, potential and served) define a bound on the cardinality of the state space given by $\mathscr{O}(3^n W)$.

In any non-terminal state (w, R, P) with $w \geq 1$, we choose between waiting with the vehicle

at the depot, and dispatching the vehicle to serve a set of requests $S \subseteq R$, which is equivalent to selecting a route of duration $d \leq w$ serving the set $\{i \in R : d_i \leq d\}$. Define then the action space $\mathscr{A}_R^w := \{d_i \,|\, \forall i \in R : d_i \leq w\}$, with cardinality $\mathscr{O}(n)$. Selecting an action d in a given state (w, R, P) transforms the state as follows. If a dispatch of length d is selected, R is partitioned into the new set of unattended requests $R_d := \{i \in R : d_i > d\}$ and the set of served requests $\bar{R}_d = R \setminus R_d$. Time moves forward to $w - d$ and state (w, R, P) becomes $(w - d, R_d \cup F_d^w, P \setminus F_d^w)$ where $F_d^w := \{i \in N : w > \tau_i \geq w - d\}$ is the set of newly arriving requests. If no dispatch occurs $(d = 0)$, the new state is $(w - 1, R \cup F_1^w, P \setminus F_1^w)$.

Let $C_w(R, P)$ be a set function representing the minimum expected cost-to-go at state $(w, R, P) \in \mathscr{S}$. The optimal expected cost C^* is defined recursively over $w \in \mathscr{W}_0$ in (2.1), where $\hat{R}$ is the set of open requests at the start of the horizon $(w = W)$. First, at $w = 0$ the cost-to-go is simply the sum of penalties of unserved requests, and subsequently, for each $w \in \mathscr{W}$ the cost-to-go at state (w, R, P) is equal to the minimum cost between no dispatch and a dispatch to any distance $d \in \mathscr{A}_R^w$:

$$C_0(R, P) = \textstyle\sum_{i \in R} p_i \qquad \forall (R, P) \in \Xi \tag{2.1a}$$

$$C_w(R, P) = \min_{d \in \mathscr{A}_R^w \cup \{0\}} \left\{ \alpha d + \mathbb{E}_{F_d^w} \left[C_{w - \max\{1, d\}} \left(R_d \cup F_d^w, P \setminus F_d^w \right) \right] \right\}, \quad \forall w \in \mathscr{W}, (R, P) \in \Xi \tag{2.1b}$$

$$C^* = \mathbb{E}_{\hat{R}} \left[C_W \left(\hat{R}, N \setminus \hat{R} \right) \right]. \tag{2.1c}$$

Formulation (2.1) is a generalization that considers an uncertain set $\hat{R}$, meaning that the initial set of open requests is not disclosed when computing the problem's expected cost, but a useful special case is when $\hat{R}$ is known. The optimal action $d_w^*(R, P) \in \mathscr{A}_R^w \cup \{0\}$ that attains $C_w(R, P)$ is

then defined as a set function for each state (w,R,P). The vector of optimal actions for each state is called an optimal policy. We can also express optimality conditions using a standard LP dual reformulation of (2.1),

$$C^* = \max_{\{\mathbf{C}\geq 0\}} \mathbb{E}_{\hat{R}}\left[C_W\left(\hat{R}, N\setminus\hat{R}\right)\right] \tag{2.2a}$$

$$\text{s.t. } C_0(R,P) \leq \textstyle\sum_{i\in R} p_i, \qquad \forall (R,P)\in\Xi \tag{2.2b}$$

$$C_w(R,P) \leq \mathbb{E}_{F_1^w}\left[C_{w-1}\left(R\cup F_1^w, P\setminus F_1^w\right)\right], \qquad \forall w\in\mathscr{W}, (R,P)\in\Xi \tag{2.2c}$$

$$C_w(R,P) \leq \alpha d + \mathbb{E}_{F_d^w}\left[C_{w-d}\left(R_d\cup F_d^w, P\setminus F_d^w\right)\right], \quad \forall w\in\mathscr{W}, (R,P)\in\Xi, d\in\mathscr{A}_R^w, \tag{2.2d}$$

which very clearly shows the difficulty in finding an optimal policy; formulation (2.2) has exponentially many variables, exponentially many constraints and exponentially many terms in the expectations.

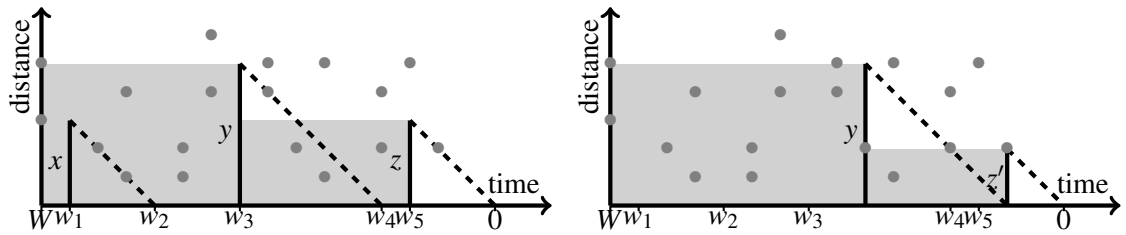
2.3 *A Priori* Solutions for the Stochastic DDWP

In this section we develop *a priori* policies for the DDWP defined in (2.1). We begin studying the deterministic version of the problem to understand the structure of optimal *a priori* policies.

2.3.1 The Deterministic Case

Suppose arrivals are known with certainty at the beginning of the horizon, and let the set of arriving requests be $N_A := \{i\in N \,:\, \tau_i > 0\}$. Requests still arrive dynamically over the operating period, and thus it remains infeasible to serve a request with a vehicle dispatch prior to its arrival time.

Figure 2.1a gives an instance where arrival times and destinations for each request $i\in N_A$ are represented by a coordinate (τ_i, d_i) in a distance versus time graph. Also depicted is an example



(a) Suboptimal Operation. (b) Operation that could be optimal.

Figure 2.1: Examples of vehicle operations described in the distance versus time graph.

vehicle dispatch plan. The vehicle starts at the depot at wave W and waits until w_1 when it is dispatched a distance x. Then, it returns at $w_2 = w_1 - x$ and waits until w_3 to execute a second dispatch of distance y, and so on. Requests covered by this operation are those with coordinates inside the shaded areas. We now state and prove three properties that at least one optimal vehicle dispatch plan should satisfy:

Property 2.3.1 (Decreasing consecutive dispatches)**.** *For all dispatch pairs starting at two waves $w > w'$ with respective dispatch durations d and d', we have $d > d'$.*

Proof. If $d' \geq d$, by deleting the dispatch at w we reduce operational cost with unaltered coverage. □

Property 2.3.2 (No wait after a dispatch)**.** *The vehicle does not wait once the first dispatch has occurred.*

Proof. If a solution waits for w waves after a dispatch at w, we can shift forward each vehicle dispatch that occurs prior to wave w exactly w waves in time without reducing the set of covered requests. □

Property 2.3.3 (Dispatch duration equals round-trip time to some request)**.** *The duration of each dispatched route equals $d_S = \max_{i \in S} d_i$, where $S \subseteq R$ is the set of requests served by the route.*

Proof. If $d > d_S$, by setting $d = d_S$ we reduce operational cost with unaltered request coverage. □

Figure 2.1b depicts an operation that satisfies all properties. A direct consequence of these properties is that we can formulate a deterministic dynamic program with a reduced state space.

Let the set of possible dispatch durations be $\mathscr{D} := \{d_i \mid \forall i \in N_A : d_i \leq \tau_i\}$. We can find an optimal dispatch plan via a dynamic program with states (w,x), where w is the current wave and x is the duration of the previous dispatch completed at wave w ($x = 0$ if no dispatches have occurred prior to w). Figure 2.2 is an example of the system at state (w,x), where the last dispatch was of duration x at wave $w+x$ and covered all requests shaded in light gray. Requests shaded in medium gray will never be served by an optimal dispatch plan satisfying the previous three properties and are thus lost, and the requests shaded in dark gray could be covered by the next dispatch at wave w.

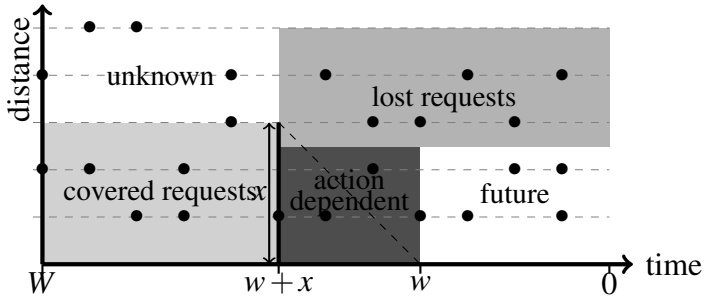


Figure 2.2: Example of state and action for the deterministic DDWP.

An action given state (w,x) is defined as the next dispatch duration $d \in \mathscr{A}_{w,x}$, where

$$\mathscr{A}_{w,x} = \{d_i \mid \forall i \in N_A : d_i \leq w, d_i < x,\ w \leq \tau_i < w+x\}, \quad \forall x \in \mathscr{D} : w+x \leq W \tag{2.3a}$$

$$\mathscr{A}_{w,0} = \{0\} \cup \{d_i \mid \forall i \in N_A : d_i \leq w,\ \tau_i \geq w\}. \tag{2.3b}$$

If no dispatches have occurred by w ($x = 0$), an optimal vehicle operation may wait until $w-1$, *i*.e., $d = 0$.

Define $C_w(x)$ as the cost-to-go function in state (w,x). Optimality equations are given by (2.4),

where $C_W(0)$ is the minimum cost for the deterministic DDWP:

$$C_0(x) = \sum_{i \in N_A} p_i, \qquad \forall\, x \in \mathscr{D} \cup \{0\} \tag{2.4a}$$

$$C_w(0) = \min_{d \in \mathscr{A}_{w,0}} \left\{ \alpha d - \sum_{i \in N_A:\, d_i \le d,\, \tau_i \ge w} p_i + C_{w-\max\{1,d\}}(d) \right\}, \qquad \forall\, w \in \mathscr{W} \tag{2.4b}$$

$$C_w(x) = \min_{d \in \mathscr{A}_{w,x}} \left\{ \alpha d - \sum_{i \in N_A:\, d_i \le d,\, w \le \tau_i < w+x} p_i + C_{w-d}(d) \right\}, \qquad \forall\, w \in \mathscr{W},\, x \in \mathscr{D} : w + x \le W. \tag{2.4c}$$

In this dynamic program, we initialize by assuming that all arrived requests are not served by $w = 0$. When we execute a dispatch of duration d, we incur its operating cost while also saving the penalties of the requests served. This dynamic program has $\mathscr{O}(nW)$ states, $\mathscr{O}(n)$ possible actions for each state, and the cost of each action can be evaluated in constant time by computing its cost incrementally over the dispatch distance. So, these equations are solvable in $\mathscr{O}(n^2 W)$ operations.

2.3.2 The stochastic case and *a priori* policies

Consider again the stochastic DDWP defined in (2.1). We next develop the optimal static *a priori* solution in which a schedule specifying the waves at which to dispatch the vehicle and the duration of each dispatch is determined only with information revealed at the start of the horizon in wave W.

The operating cost of such an *a priori* solution is known, and the penalties paid for unserved requests depend on the future arrivals. This observation motivates an approach for determining an *a priori* solution that minimizes expected cost. This problem is equivalent to solving a deterministic DDWP instance in which each potential request $i \in N$ is copied W times and assumed to arrive at *every* wave $w \in \mathscr{W}$ for which its probability of arrival is positive, with an adjusted penalty for

not serving the request at wave w equal to $p_i P(\tau_i = w | \tau_i < W)$. Known requests ($\tau_i = W$) are not copied and arrive only at wave W with probability one. Thus, the recursive equations to find an optimal *a priori* policy are a natural extension of the deterministic system (2.4),

$$C_0^{AP}(x) = \sum_{i \in N: \tau_i = W} p_i \quad + \sum_{i \in N: \tau_i < W} \mathbb{P}(\tau_i > 0 \mid \tau_i < W) p_i, \qquad \forall x \in \mathscr{D} \cup \{0\} \quad (2.5a)$$

$$C_w^{AP}(0) = \min_{d \in \mathscr{A}_{w,0}} \left\{ \alpha d \quad - \sum_{\substack{i \in N: \\ \tau_i = W,\, d_i \le d}} p_i \quad - \sum_{\substack{i \in N: \\ \tau_i < W,\, d_i \le d}} \mathbb{P}(\tau_i \ge w | \tau_i < W) p_i \quad + C^{AP}_{w - \max\{1,d\}}(d) \right\}, \ \forall w \in \mathscr{W} \quad (2.5b)$$

$$C_w^{AP}(x) = \min_{d \in \mathscr{A}_{w,x}} \left\{ \alpha d \quad - \sum_{\substack{i \in N: \\ \tau_i < W,\, d_i \le d}} \mathbb{P}(w \le \tau_i < w + x | \tau_i < W) p_i \quad + C^{AP}_{w-d}(d) \right\}, \quad \forall w \in \mathscr{W}, x \in \mathscr{D} : w + x \le W, \quad (2.5c)$$

where the optimal expected cost is given by $C_W^{AP}(0)$. Note that this policy is found by solving a deterministic DDWP, and so, it satisfies Properties (2.3.1), (2.3.2) and (2.3.3). In addition, we can preprocess the values of all probabilities in (2.5) and keep the $\mathscr{O}(n^2 W)$ running time.

Our *a priori* problem requires knowledge of available orders at wave W. To estimate the expected cost of implementing this heuristic policy, we simulate a set $m \in \{1, \ldots, M\}$ of vectors $\tau(m)$; each one containing an arrival time $\tau_i(m)$ for each order $i \in N$. We solve this heuristic for each realization $m \in M$ assuming that the set $\{i \in N : \tau_i(m) = W\}$ is known, and then we take the average cost over all M realizations. For consistency in computational results, we will use the same M realizations when comparing performance of lower bounds and different solutions for a given instance.

We can improve the performance of an *a priori* policy by allowing simple recourse actions during the operation. Let a policy be represented by the ordered set of k dispatches, each with dispatch distance d^j and wave w^j: $\{(d^j, w^j)\}_{j=1}^k$. The policy satisfies $w^j - w^{j+1} = d^j$ and $d^j > d^{j+1}$ for $j = 1, \ldots, k-1$. Consider the following recourse actions:

1. *Postponement and cancellation*: Consider dispatch j scheduled at wave w^j. Any open request i with $d_i \le d^{j+1}$ can be covered by dispatch $j+1$. So, if no request i has arrived

since the previous dispatch time w^{j-1} with $d^{j+1} < d_i \leq d^j$, then postpone dispatch j by modifying its scheduled time $w^j \leftarrow w^j - 1$ and its duration $d^j \leftarrow d^j - 1$ if $d^j - 1 > d^{j+1}$, otherwise cancel dispatch j. A rescheduled dispatch is considered again for postponement and cancellation iteratively.

2. *Marginal profit adjustment*: Given a dispatch j that has not been postponed or cancelled, adjust its distance d^j to maximize its marginal profit. This is accomplished by choosing the actual dispatch distance d equal to the location d_r of an open request $r \in N$ with $d^{j+1} < d_r \leq d^j$, such that it maximizes the following marginal profit

$$V^{mg}(r,w) := \left(\sum_{\{i \in N : d^{j+1} < d_i \leq d_r,\ \tau_i \geq w\}} p_i - d_r \right). \tag{2.6}$$

If $V^{mg}(r,w) \leq 0$ for each possible request r, then again postpone the dispatch to $w^j \leftarrow w^j - 1$ with new dispatch duration $d_j \leftarrow d^j - 1$ if $d^j - 1 > d^{j+1}$, otherwise cancel dispatch j. If the dispatch is adjusted such that $d < d^j$, we also postpone it to time $w^{j+1} + d$ to potentially serve more customers with no increase in cost.

2.4 Dynamic Policies for the Stochastic DDWP

A priori policies, particularly when adjusted via recourse actions, may yield reasonable solutions to many problems. However, there exist instances for which an optimal adjusted *a priori* policy is arbitrarily worse than an optimal dynamic policy.

Pathological *A Priori* Instances Consider a family of instances with 2 requests, $W = 4$, and a parameter $z \geq 0$. Let locations be $d_1 = 1$ and $d_2 = 2$, and penalties $p_1 = z + 1$ and $p_2 = z^2 + z + 2$. Request 1 arrives at $\tau_1 = 1$, while request 2 arrives at $\tau_2 = 3$ with probability $u = \frac{z}{z+1}$ and $\tau_2 = 2$ with probability $v = \frac{1}{z+1}$. There are four possible *a priori* solutions (See Figure 2.3). Either of the last two options (c) or (d) are optimal *a priori* policies with simple recourse, and both have

expected cost of $3+z$.

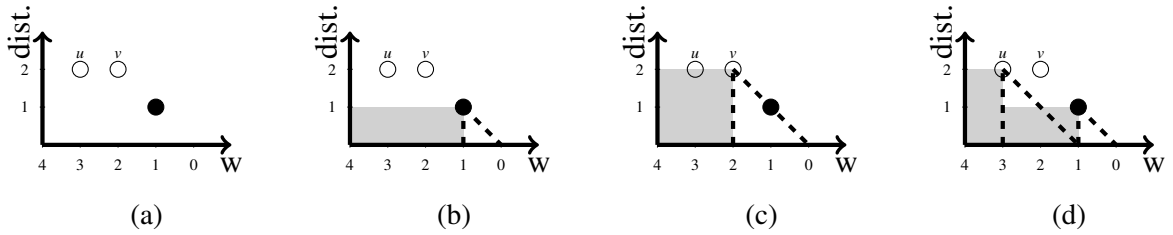


Figure 2.3: Feasible *a priori* dispatch options.

The optimal dynamic policy is different. If request 2 arrives at $w = 3$, it dispatches to $d = 2$ at $w = 3$ and then $d = 1$ at $w = 1$ for total cost of 3; otherwise it dispatches to $d = 2$ at $w = 2$ for cost of $3+z$. The expected cost of this policy is $3u+(3+z)v = 3+\frac{z}{z+1} < 4$. As $z \to \infty$, the optimal cost is bounded, while any *a priori* policy's cost is unbounded.

2.4.1 *A Priori*-Based Rollout Policy

One approach to build a dynamic policy is to roll out the *a priori* policy. At each wave $w \in \mathcal{W}$ when the vehicle is available, we recompute an optimal *a priori* policy given updated information regarding requests (open, potential, and served); if the policy dictates a dispatch $d > 0$ at w, the decision is executed and a new *a priori* policy is then computed at $w-d$, otherwise a new *a priori* policy is computed at $w-1$. Computing such a rollout policy requires $\mathcal{O}(n^2W^2)$ operations, *i*.e., it solves $\mathcal{O}(W)$ *a priori* problems.

2.4.2 Approximate Linear Programming for the DDWP

Heuristic dynamic policies can be generated via the dual MDP reformulation (2.2). Because this formulation has exponentially many variables and constraints, the ALP approach restricts its feasible region in such a way that the resulting optimization model is tractable and so it yields a lower bound for the optimal expected cost-to-go that can be used within a rollout policy.

We can generate a lower bound $C_w^{ALP}(R,P)$ for the cost-to-go function of the DDWP at any feasible state (w,R,P) by representing $C_w(R,P)$ as a linear function of a predetermined set of basis functions, and then solving the resulting restriction of (2.2) to obtain multipliers for these basis functions. Let $C_w(R,P) \approx C_w^{ALP}(R,P)$, where

$$C_w^{ALP}(R,P) := \sum_{i\in R} a_i^w + \sum_{i\in P} b_i^w - \sum_{k=1}^{w} v_k, \tag{2.7}$$

and where a_i^w represents the cost of request i if it is open at wave w, b_i^w represents the cost of potential request i if it hasn't arrived by wave w, and v_k represents the incremental value of each wave k.

Proposition 2.4.1. *Applying the restriction* (2.7) *to* (2.2) *yields a model equivalent to*

$$C^{ALP} = \max_{\{\mathbf{a},\mathbf{b},\mathbf{v},\mathbf{s},\mathbf{u}\}} \sum_{i\in N} \left(\mathbb{P}(\tau_i = W)\, a_i^W + \mathbb{P}(\tau_i < W)\, b_i^W\right) - \sum_{w=1}^{W} v_w \tag{2.8a}$$

$$s.t.\ a_i^0 = p_i, b_i^0 = 0, \qquad \forall i \in N \tag{2.8b}$$

$$s_{iw} \geq a_i^w - a_i^{w-1}, \qquad \forall i \in N, w \in \mathscr{W} \tag{2.8c}$$

$$s_{iw} \geq b_i^w - f_{iw} a_i^{w-1} - \bar{f}_{iw} b_i^{w-1}, \qquad \forall i \in N, w \in \mathscr{W} \tag{2.8d}$$

$$\sum_{i\in N} s_{iw} \leq v_w, \qquad \forall w \in \mathscr{W} \tag{2.8e}$$

$$u_{iw}^d \geq a_i^w - \mathbb{I}_{(d_i > d)} a_i^{w-d}, \qquad \forall i \in N, w \in \mathscr{W}, d \in \mathscr{A}_N^w \tag{2.8f}$$

$$u_{iw}^d \geq b_i^w - g_{iw}^d a_i^{w-d} - \bar{g}_{iw}^d b_i^{w-d}, \qquad \forall i \in N, w \in \mathscr{W}, d \in \mathscr{A}_N^w \tag{2.8g}$$

$$\sum_{i\in N} u_{iw}^d \leq \sum_{k=w-d+1}^{w} v_k + \alpha d \qquad \forall w \in \mathscr{W}, d \in \mathscr{A}_N^w \tag{2.8h}$$

$$\mathbf{u}, \mathbf{s} \geq 0, \tag{2.8i}$$

where $f_{iw} := \mathbb{P}(\tau_i = w-1 \mid \tau_i < w)$ *is the conditional probability that potential request i at wave w*

arrives at the next wave, $g_{iw}^d := \mathbb{P}(\tau_i \geq w - d) \mid \tau_i < w)$ *is the conditional probability that potential request* i *at wave* w *arrives in one of the next* d *waves, and also* $\bar{g}_{iw}^d := 1 - g_{iw}^d$ *and* $\bar{f}_{iw} := 1 - f_{iw}$.

Any set of values $\{\mathbf{a}, \mathbf{b}, \mathbf{v}\}$ *used to compute* $C_w^{ALP}(R,P)$ *which are feasible for* (2.8b)-(2.8i) *yield a lower bound of the cost-to-go function at any state* (w,R,P)*:* $C_w^{ALP}(R,P) \leq C_w(R,P)$*. In particular, we have* $C^{ALP} \leq C^*$.

The proposition's proof is in the online supplement. Model (2.8) has interesting properties which give economic intuition and accelerate computation times; each of these properties is proved in the online supplement.

Property 2.4.2 (Bounds)**.** *We may assume* $0 \leq a_i^w \leq p_i$ *and* $0 \leq b_i^w \leq g_{iw}^w p_i, \forall i \in N, w \in \mathscr{W}_0$ *without loss of optimality.*

Intuitively, Property 2.4.2 implies that the individual cost per open request at any wave is nonnegative and cannot exceed the penalty for leaving the request unattended, and that the individual cost for any potential request at any wave is nonnegative and cannot exceed the penalty discounted by the arrival probability.

Property 2.4.3 (Lost requests)**.** *Without loss of optimality, we may assume that* $a_i^w = p_i$ *for any* $i \in N, w \in \mathscr{W}_0 : d_i > w$*, and* $b_i^w = g_{iw}^w p_i$ *for any* $i \in N, w \in \mathscr{W}_0 : d_i \geq w$.

Property 2.4.3 says that the cost of having an open request i at time w with an impossible dispatch ($d_i > w$) is equal to p_i. A similar idea motivates the expression for b_i^w.

The following theorem, also proved in the online supplement, describes the performance of the ALP lower bound in the deterministic case.

Theorem 2.4.4 (Strong duality for the deterministic case)**.** *Assume request* i*'s arrival wave* τ_i *is deterministic for each request* $i \in N$*. Then the bound given by (2.8) is tight, i.e., equal to the optimal cost of the deterministic DDWP given in (2.4).*

The result gives further motivation to use ALP for the DDWP, since the approximation is able to recover optimality in the deterministic case. Furthermore, it relates the ALP and the *a priori* solution: if we transform a stochastic instance into a deterministic one as described in Section 2.3, the ALP matches the *a priori* solution, and both can be used heuristically. However, the ALP can also be used without the transformation, so it can be viewed as a generalization of the *a priori* rollout policy.

We next apply (2.8) to approximate the optimal action $d_w^*(R,P)$. Given a feasible $(\mathbf{a},\mathbf{b},\mathbf{v})$ to (2.8b)-(2.8i), we have a closed linear form lower bound for the expected cost-to-go function measured after a decision with dispatch distance $d \in \mathscr{A}_R^w$ has been taken via

$$\begin{aligned}\mathbb{E}_{F_d^w}\left[C_{w-d}\left(R_d \cup F_d^w, P \setminus F_d^w\right)\right] &\geq \mathbb{E}_{F_d^w}\left[C_{w-d}^{ALP}\left(R_d \cup F_d^w, P \setminus F_d^w\right)\right] \\ &= \mathbb{E}_{F_d^w}\left[\textstyle\sum_{i\in R_d \cup F_d^w} a_i^{w-d} + \sum_{i\in P\setminus F_d^w} b_i^{w-d} - \sum_{k=1}^{w-d} v_k\right] \\ &= \textstyle\sum_{i\in R_d} a_i^{w-d} + \sum_{i\in P}\left(g_{iw}^d a_i^{w-d} + \bar{g}_{iw}^d b_i^{w-d}\right) - \sum_{k=1}^{w-d} v_k. \end{aligned} \tag{2.9}$$

A similar expression can be obtained to underestimate the expected cost-to-go measured after the vehicle waits for one wave at the depot. We use these bounds to compute an approximately optimal action $d_w^{ALP}(R,P)$.

Any feasible set of values $\{\mathbf{a},\mathbf{b},\mathbf{v}\}$ provides an underestimate of the expected cost-to-go in (2.9). In particular, the tightest lower bound is achieved when maximizing (2.9) subject to (2.8b)-(2.8i). This is a *post state and decision re-optimization* of the ALP in which the values of $\{\mathbf{a},\mathbf{b},\mathbf{v}\}$ are recomputed at each wave w when the vehicle is ready at the depot, and for each potential action $d \in \mathscr{A}_{R^w(m)}^w \cup \{0\}$. We compute the approximate optimal action by

$$d_w^{ALP}(R,P) = \underset{d\in\mathscr{A}_R^w\cup\{0\}}{\operatorname{argmin}} \begin{cases} \max_{\{(\mathbf{a},\mathbf{b},\mathbf{v})\in(2.8b)-(2.8i)\}} \sum_{i\in R} a_i^{w-1} + \sum_{i\in P}\left(f_{iw}a_i^{w-1} + \bar{f}_{iw}b_i^{w-1}\right) - \sum_{k=1}^{w-1} v_k, & \text{if } d=0 \\ \alpha d + \max_{\{(\mathbf{a},\mathbf{b},\mathbf{v})\in(2.8b)-(2.8i)\}} \sum_{i\in R_d} a_i^{w-d} + \sum_{i\in P}\left(g_{iw}^d a_i^{w-d} + \bar{g}_{iw}^d b_i^{w-d}\right) - \sum_{k=1}^{w-d} v_k, & \text{else.} \end{cases} \tag{2.10}$$

This involves solving $\mathscr{O}(nW)$ linear programs sharing the same feasible set of solutions. Its per-

formance can be improved by applying LP warmstart and ruling out suboptimal dispatch distances (see the online supplement for details). The procedure is repeated for each realization $m = 1, \ldots, M$ of arrivals to estimate its expected cost.

2.5 Computational Experiments

We present two sets of computational experiments using different families of randomly generated instances. Our goal is to test the quality of the various heuristics and to obtain qualitative insights regarding solutions. The two sets of experiments differ in their models of the request arrival process. In the first set, we assume that the conditional likelihood of a request arrival by the next dispatch at wave w is constant over time but may vary by request. In the second set, we use an arrival distribution that assigns probabilities for the arrival time (or the non-arrival event) for each request using a mean arrival that varies by request. All heuristics were programmed in Java and computed using a 2.1GHz Intel Core i7-3612QM processor with 8 GB RAM, using CPLEX 12.4 when necessary as the LP solver.

Table 2.1 summarizes the lower bounds and heuristic policies' costs that we computed for the instances in this study. We do not include the ALP lower bound, as our preliminary experiments revealed it to be weaker than the PIR bound. Similar behavior has been observed in other stochastic routing contexts, *e*.g., [65].

For each particular instance, we simulated $M = 100$ realizations of the arrival time vector τ, and use this common set to estimate lower bounds and policies' expected costs via Monte Carlo sampling.

Table 2.1: Lower bounds and heuristic policies' costs computed

Type	**Procedures**
Lower bound	perfect information relaxation (PIR)
A priori policies	Static *a priori* policy (AP) & *a priori* policy with recourse actions (APR)
Dynamic policies	dynamic *a priori* policy rollout (DAP) & dynamic ALP policy (DALP)

2.5.1 Design of Instance Set 1: Stationary Conditional Arrival Probability

The first set of experimental instances model arrivals using a stationary conditional arrival distribution for each request. Therefore, for each $i \in N$, the probability that it arrives at wave w given that it has not yet arrived is independent of w, *i*.e., $\mathbb{P}(\tau_i = W) = \mathbb{P}(\tau_i \geq w-1 \mid \tau_i < w) = \theta_i$ and $\mathbb{P}(\tau_i = -1) = (1-\theta_i)^{W-1}$.

We construct instances with different size, geography, and time flexibility as follows. Let (n, ℓ, r) define an instance where n is the number of potential requests over the horizon; ℓ is the maximum distance between a request and the depot; and $r := W/\ell$ is the ratio between the total number of waves W and ℓ. We consider all combinations of $n \in \{5, 10, 20, 40, 60, 80, 100\}$, $\ell \in \{5, 10, 20\}$, and $r = \{1, 2, 3\}$ and generate 20 random instances for each combination by varying the vectors $\{\theta_i\}$, $\{p_i\}$, and $\{d_i\}$ as

$\{\theta_i\}$: probability parameter θ_i for each i drawn i.i.d. from a continuous uniform distribution, $U(\frac{1}{2W}, \frac{2}{W})$;

$\{p_i\}$: penalty parameter p_i drawn with equal probability from the values $\{0.25\ell, 0.5\ell, 0.75\ell, \ell\}$

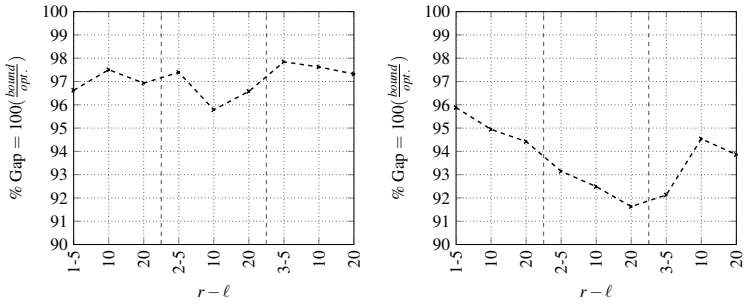
$\{d_i\}$: distance parameter d_i drawn with equal probability from the values $\{1, \ldots, \ell\}$.

2.5.2 Results for Instance Set 1

Figure 2.4 reports the average duality gap between the PIR bound and the optimal expected cost for small instances ($n \in \{5, 10\}$) where the fully dynamic-stochastic problem is solvable to optimality.

Table 2.2: Overall performance of heuristics in Instance Set 1

Heuristic	**%GAP vs OPT (small instances)**	**%GAP vs lower bound**	**Time per sample-instance** (secs)
AP	11.53%	12.14%	0.0124
APR	5.27%	9.24%	0.0122
DAP	1.97%	6.59%	0.1489
DALP	1.65%	6.42%	0.5869



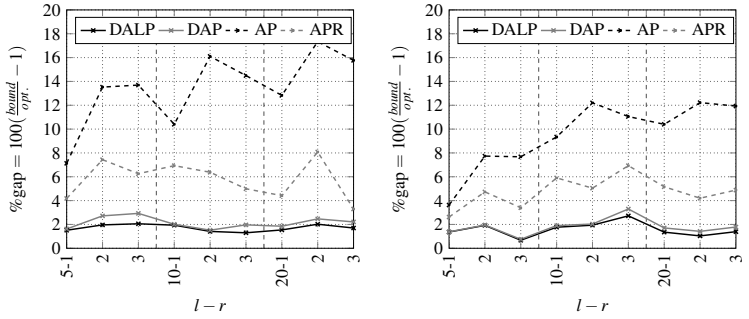
(a) $n = 5$ (b) $n = 10$

Figure 2.4: Percentage gap between PIR lower bound and optimal solution values for Instance Set 1

Table 2.2 presents average gap and solution times for each heuristic. In case of the ALP-based policy (DALP), we employed a hybrid approach that executes DAP until the operation reaches wave $x\ell$ and, afterwards, executes an ALP-based policy. The motivation is the two policies' complementary behavior. The ALP tends to be too conservative initially, when the remaining horizon includes many possibilities it has to under-approximate, while DAP simply assumes "averages" for the future; conversely, towards the end of the horizon the ALP can more accurately assess possible future recourse actions, and thus can make better decisions. Also, the linear programs in the ALP tend to have highly degenerate polytopes for instances with high flexibility, making them difficult to solve. After searching over a grid of different values in preliminary experiments, we concluded that $x = 1.1$ yields the best gap while still keeping computation times low. This contrasts with naive implementations of ALP policies, which can be computationally demanding.

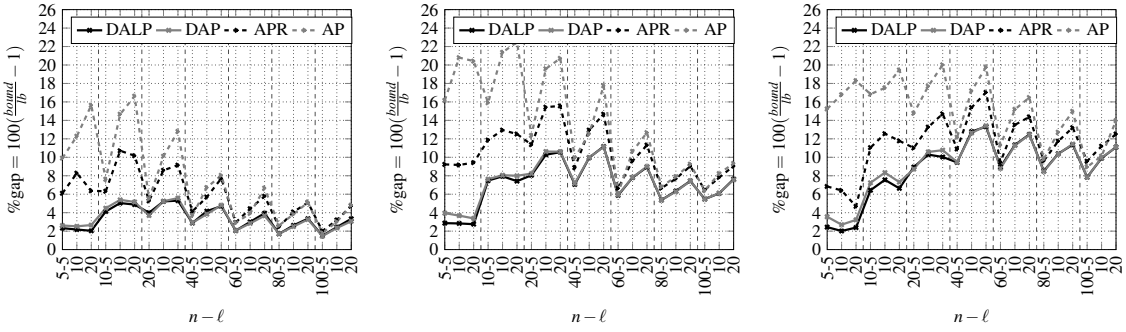
For small instances with $n = 5$ or $n = 10$, Figure 2.5 shows the average relative gap to the optimal solution. The dynamic *a priori* policy rollout (DAP) and the dynamic ALP-based policy (DALP) dominate the a priori solutions and achieve an average gap of 1.97% and 1.65%, respectively.

For larger instances, the gap is computed with respect to the PIR bound. Figure 2.6 details



(a) $n = 5$ (b) $n = 10$

Figure 2.5: Average percentage gap between heuristic solution costs and optimal costs for Instance Set 1



(a) $r = 1$ (b) $r = 2$ (c) $r = 3$

Figure 2.6: Average percentage gap between heuristic solution costs and lower bound for Instance Set 1

average gaps over all classes of instances.

As expected based on each heuristic's recourse possibilities, APR outperforms AP and both are outperformed by the two dynamic policies (DAP and DALP). Also, the gap differences between AP, APR and the dynamic policies decrease with n. This suggests that dynamic solutions produce a bigger gap improvement for instances with more request arrival granularity, *i*.e., where an early or late arrival can significantly impact costs unless corrective actions are taken. Conversely, for instances with more requests the marginal value of dynamic solutions is smaller. This may be due to risk pooling effects between requests, *e*.g., if one out of 100 requests arrives early, another one

will likely arrive late and the relative disturbance will be minor. Moreover, the relative gap of both dynamic policies as a function of n reaches a maximum and then decreases as n grows. This confirms their effectiveness for large n. Also, all four heuristics' gaps increase as a function of r; the level of flexibility translates into solution complexity for our heuristics. Additionally, the gap tends to increase with ℓ; this is likely related to an increase in the problem's complexity. Finally, the ALP-based policy has an average gap smaller than DAP. For less flexible instances ($r = 1$) both approaches average a relative gap of 3.4%, but when the variability and recourse flexibility increases to $r = 2$ and $r = 3$ it improves over DAP, from 7.3% to 7.1% for $r = 2$ and from 9.0% to 8.8% for $r = 3$. Although small, this improvement was consistently observed across all instances.

2.5.3 Design of Instance Set 2: Uniform Arrivals

The previous arrival distributions defined by a single parameter could be hiding interesting interdependencies between mean, variance, probability of arrival, and degree of dynamism. We defined a second set of experiments with a fixed number of requests ($n = 20$), waves ($W = 30$) and maximum location ($\ell = 10$). The distance vector $\mathbf{d}$ and penalty vector $\mathbf{p}$ are set as in the previous experiments, but arrivals have distributions with a probability p_{start} of arrival at the beginning of the horizon (*i*.e., the degree of dynamism), a probability p_{out} of not showing up, and a discrete uniform probability $\frac{1-p_{start}-p_{out}}{(\min\{W-1,\mu_i+v\}-\max\{1,\mu_i-v\})+1}$ of arriving during the operation at wave $w = \max\{1,\mu_i-v\},\ldots,\min\{W-1,\mu_i+v\}$, where μ_i is a request-dependent parameter drawn i.i.d. from a discrete uniform distribution $U(0,W-1)$ for each $i \in N$. The parameter v represents the arrival variability (half of the arrival range). We created 20 instances for each set of parameters (v,q,w) in the set

$$\{(v,p_{out},p_{start}) : v \in \{0,1,2,4,8,30\}, p_{out} \in \{0,0.2,0.4\}, p_{start} \in \{0,0.2,0.4,0.6,0.8,1\} : p_{out}+p_{start} \leq 1\}. \quad (2.11)$$

Table 2.3: Overall performance of heuristics in Instance Set 2.

Upper Bound	**%GAP**	**Time per sample-instance** (secs)
AP	7.54%	0.0006
APR	5.62%	0.0006
DAP	4.46%	0.0066
DALP	4.24%	0.442

2.5.4 Results for Instance Set 2

Table 2.3 presents overall results for each heuristic over the second set of experiments. We notice that our simple recourse rules in APR capture $58\% = \frac{7.54-5.62}{7.54-4.24}$ of the total gap improvement that the best dynamic heuristic captures over the static solution AP. Figure 2.7 presents average relative gaps over instances with different settings of parameters $p_{out} - p_{start}$ or v.

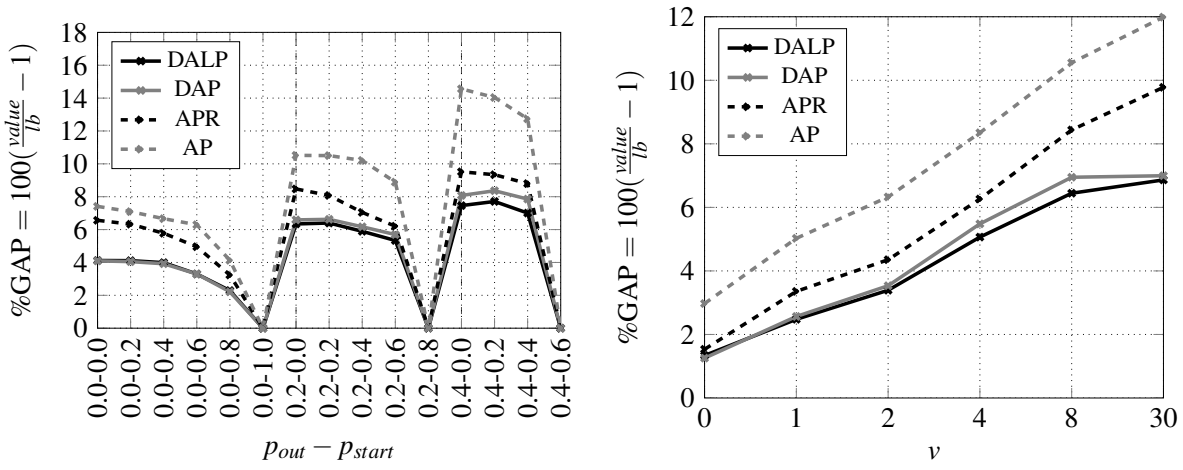


Figure 2.7: Average percentage gap between heuristics cost and lower bound in Instance Set 2.

From these graphs we conclude that the relative gaps of all four policies decrease as p_{start} increases; the more information available at the initial wave, the closer we can get to a deterministic problem. There is zero gap in the extreme deterministic cases ($p_{out} + p_{start} = 1$). The value of dynamic solutions also decreases when p_{start} increases, which is expected, since a smaller p_{start} implies a larger degree of dynamism and more importance is placed on recourse actions. Regarding

the request arrival probability, the gap increases with p_{out} (unless $p_{out} + p_{start} = 1$). This means that it is harder to optimize an instance for which there is a bigger probability of no arrival. The value of dynamic solutions also grows with p_{out}. With respect to the variability of the instance, the gap increases as v increases. This may be due both to a decrease in the lower bound's quality and to an increase in the optimal expected cost. Finally, the dynamic heuristics yield larger marginal costs savings when v increases. This means that the more variability the system has, the more important it is to implement a dynamic solution. There is also a range of intermediate variability for which DALP clearly dominates DAP. In this range, the additional complexity of ALP yields the most benefit. Table 2.4 provides four examples of instance families within this rage; their average percent reduction in relative gap of DALP over DAP is 15.0%.

Table 2.4: Average gap percent reduction of DALP for cases with intermediate variability in Instance Set 2.

Family (p_{out}, p_{start}, v)	**DALP %GAP**	**DAP %GAP**	**% reduction over DAP**
(0.4, 0.2, 4)	8.23%	9.67%	14.9%
(0.4, 0.2, 8)	10.79%	12.41%	13.1%
(0.4, 0.4, 4)	7.46%	8.95%	16.6%
(0.4, 0.4, 8)	8.68%	10.31%	15.8%
Aggregate	**8.79%**	**10.34%**	**15.0%**

2.6 Conclusions

We have formulated the dynamic dispatch waves problem (DDWP) to capture the basic aspects of dispatch and routing decisions for same-day delivery. This papers initiates work on the DDWP by studying the single-vehicle stochastic case where customer destinations are placed over the line.

We develop a set of tractable solution policies that differ in their solution dynamism, from an *a priori* solution to fully dynamic policies. Our computational experiments indicate that the performance of an *a priori* policy is good, especially when we include heuristic improvements. In computational tests over two instance sets this policy yields an expected cost within 9.24% and

5.62% of the best lower bound. Nevertheless, we prove that the benefit of a fully dynamic policy can be unbounded in the worst-case scenario. Accordingly, we proposed and experimentally tested two dynamic policies that differ by the nature of the approximate cost-to-go function: the rollout of the *a priori* solution and an ALP-based dynamic policy. The rollout of the *a priori* policy computes this policy at the start of the horizon, but only implements the first action, then updates all known information and re-computes a new a priori solution. In both sets of instances it cuts the gap of our *a priori* policy with recourse by 28.7% and 20.6%, respectively. We have also found that a dynamic policy that incorporates the ALP approach yields the best possible results. Its marginal improvement as gap reduction for both sets of experiments is 2.6% and 4.9%, respectively. In instance families with intermediate variability, this gap reduction grows to 15.0%.

A final conclusion of our study concerns the relative value of dynamic policies. With all other things being equal, the benefit of a dynamic policy over the optimal *a priori* solution eventually decreases as n grows, *i*.e., as the number of potential orders increases. This is unsurprising, since for larger numbers of potential orders one would expect an averaging effect. We found the maximum benefit in dynamic policies for order sets of around 20 to 50; for smaller numbers, the exact optimal solution is still tractable, whereas for larger numbers the *a priori* policy is close to optimality. Many same-day delivery applications, such as grocery home delivery, might expect maximum daily order volume around these numbers. Similarly, dynamic policies' benefits decrease as orders become more likely to appear at the start of the horizon. In other words, if many of the orders are not placed in the same day at all, but rather are carried over from the previous day, an *a priori* policy performs quite well. It is precisely in the most uncertain environments, where orders can appear at any moment, that new models such as ours offer the most benefit.

Future work on the DDWP needs to consider the solution on a general network topology, and thus become more applicable for SDD operations in urban networks. This problem is quite challenging; in addition to dispatch decisions, it needs to deal with difficult vehicle routing problems. Given this additional difficulty, one could deal with this problem by designing heuristics based

on insights from the one-dimensional case. It would also be interesting to extend this model to multiple vehicles that could pool the risk associated with leaving orders unattended and therefore reduce costs. Other extensions could be incorporating vehicle service times at each location or including customer service time windows instead of a deadline at the end of the day. In general, same-day delivery offers many new challenges to the logistics research community.

2.7 Appendix of chapter 2

2.7.1 Proof of Proposition 2.4.1

Proof. Applying restriction (2.7) to (2.2) yields the LP

$$C^{ALP} = \max_{\{\mathbf{a},\mathbf{b},\mathbf{v}\}} \mathbb{E}_{R_T}\left[\sum_{i\in\hat{R}} a_i^W + \sum_{i\in N\setminus\hat{R}} b_i^W\right] - \sum_{w=1}^{W} v_w \tag{2.12a}$$

$$\text{s.t. } \sum_{i\in R} a_i^0 + \sum_{i\in P} b_i^0 = \sum_{i\in R} p_i, \qquad \forall (R,P)\in\Xi \tag{2.12b}$$

$$\sum_{i\in R} a_i^w + \sum_{i\in P} b_i^w - \mathbb{E}_{F_1^w}\left[\sum_{i\in R\cup F_1^w} a_i^{w-1} + \sum_{i\in P\setminus F_1^w} b_i^{w-1}\right] \le v_w, \qquad \forall w\in\mathscr{W}, (R,P)\in\Xi \tag{2.12c}$$

$$\sum_{i\in R} a_i^w + \sum_{i\in P} b_i^w - \mathbb{E}_{F_d^w}\left[\sum_{i\in R_d\cup F_d^w} a_i^{w-d} + \sum_{i\in P\setminus F_d^w} b_i^{w-d}\right] \le \alpha d + \sum_{k=w-d+1}^{w} v_k, \quad \forall w\in\mathscr{W}, (R,P)\in\Xi, d\in\mathscr{A}_R^w. \tag{2.12d}$$

Model (2.12) has a polynomial number of variables for a given n and W, but it has exponentially many terms within the expectations and constraints. We prove Proposition 2.4.1 in two steps. First, we compute a closed form for the expectations in model (2.12). Then we show a one to one equivalence between both domains.

The expectations in (2.12) are given by

$$\mathbb{E}_{R_T}\left[\sum_{i\in\hat{R}} a_i^W + \sum_{i\in N\setminus\hat{R}} b_i^W\right] = \sum_{i\in N}\left(\mathbb{P}(\tau_i = W)a_i^W + \mathbb{P}(\tau_i < W)b_i^W\right) \tag{2.13a}$$

$$\mathbb{E}_{F_1^w}\left[\sum_{i\in R\cup F_1^w} a_i^{w-1} + \sum_{i\in P\setminus F_1^w} b_i^{w-1}\right] = \sum_{i\in R} a_i^{w-1} + \sum_{i\in P} f_{iw}a_i^{w-1} + \bar{f}_{iw}b_i^{w-1} \tag{2.13b}$$

$$\mathbb{E}_{F_d^w}\left[\sum_{i\in R_d\cup F_d^w} a_i^{w-d} + \sum_{i\in P\setminus F_d^w} b_i^{w-d}\right] = \sum_{i\in R_d} a_i^{w-d} + \sum_{i\in P} g_{iw}^d a_i^{w-d} + \bar{g}_{iw}^d b_i^{w-d}. \tag{2.13c}$$

Replacing them in (2.12) yields

$$\max_{\{\mathbf{a},\mathbf{b},\mathbf{v}\geq 0\}} \sum_{i\in N}\left(\mathbb{P}(\tau_i = W)a_i^W + \mathbb{P}(\tau_i < W)b_i^W\right) - \sum_{w=1}^{W} v_w \text{ s.t.} \tag{2.14a}$$

$$\sum_{i\in R} a_i^0 + \sum_{i\in P} b_i^0 = \sum_{i\in R} p_i \qquad \forall (R,P)\in\Xi \tag{2.14b}$$

$$\sum_{i\in R}\left(a_i^w - a_i^{w-1}\right) + \sum_{i\in P}\left(b_i^w - f_{iw}a_w^{w-1} - \bar{f}_{iw}a_i^{w-1}\right) \leq v_w \qquad \forall w\in\mathscr{W}, \forall (R,P)\in\Xi \tag{2.14c}$$

$$\sum_{i\in\bar{R}_d} a_i^w + \sum_{i\in R_d}\left(a_i^w - a_i^{w-d}\right) + \sum_{i\in P}\left(b_i^w - g_{iw}^d a_w^{w-d} - \bar{g}_{iw}^d a_i^{w-d}\right) \leq \sum_{k=w-d+1}^{w} v_k + \alpha d \quad \forall w\in\mathscr{W}, \forall (R,P)\in\Xi, \forall d\in\mathscr{A}_R^w, \tag{2.14d}$$

where we still have an exponential number of constraints. We prove that (2.8) is equivalent to (2.14) by showing equality between both domains.

1. $(2.8b) \iff (2.14b)$: Suppose that $(\mathbf{a},\mathbf{b},\mathbf{v})$ satisfies $(2.14b)$. If $R=\{i\}$ and $P=\emptyset$ we get $a_i^w = p_i$, and if $R=\emptyset$ and $P=\{i\}$ we get $b_i^w = 0$. Now, suppose that $(\mathbf{a},\mathbf{b},\mathbf{v})$ satisfies $(2.8b)$ and add $a_i = p_i$ and $b_j = 0$ over any feasible pair of sets $(R,P)\in\Xi$ to get $(2.14b)$.

2. $(2.8c),(2.8d),(2.8e) \iff (2.14c)$: Suppose that $(\mathbf{a},\mathbf{b},\mathbf{v})$ satisfies $(2.14c)$. For each $w\in\mathscr{W}$, choose a particular $(R,P)\in\Xi$ as follows: put $i\in R$ if $a_i^w - a_i^{w-1}$ is greater than the value of $\max\{0, b_i^w - f_{iw}a_i^{w-1} - \bar{f}_{iw}b_i^{w-1}\}$, and put $i\in P$ if $b_i^w - f_{iw}a_i^{w-1} - \bar{f}_{iw}b_i^{w-1}$ is greater than

$\max\{0, a_i^w - a_i^{w-1}\}$.

Then, for (i,w) set $s_{iw} = \max\left\{0, a_i^w - a_i^{w-1}, b_i^w - f_{iw}a_i^{w-1} - \bar{f}_{iw}b_i^{w-1}\right\}$ and we get

$$v_w \geq \sum_{i\in R} a_i^w - a_i^{w-1} + \sum_{i\in P} b_i^w - f_{iw}a_w^{w-1} - \bar{f}_{iw}a_i^{w-1} \tag{2.15a}$$

$$= \sum_{i\in N} \max\left\{0, a_i^w - a_i^{w-1}, b_i^w - f_{iw}a_i^{w-1} - \bar{f}_{iw}b_i^{w-1}\right\} \tag{2.15b}$$

$$= \sum_{i\in N} s_{iw}. \tag{2.15c}$$

Now suppose that $(\mathbf{a}, \mathbf{b}, \mathbf{v}, \mathbf{s})$ satisfies $(2.8c), (2.8d), (2.8e)$, select any pair $(R,P) \in \Xi$ and we have

$$v_w \geq \sum_{i\in N} s_{iw} \geq \sum_{i\in R} s_{iw} + \sum_{i\in P} s_{iw} \geq \sum_{i\in R} \left(a_i^w - a_i^{w-1}\right) + \sum_{i\in P} \left(b_i^w - f_{iw}a_w^{w-1} - \bar{f}_{iw}a_i^{w-1}\right). \tag{2.16}$$

3. $(2.14d) \iff (2.8f), (2.8g), (2.8h)$: Consider that $(\mathbf{a}, \mathbf{b}, \mathbf{v})$ satisfies $(2.14d)$. For each $w \in \mathscr{W}$ and $d \in \mathscr{A}_N^w$, choose $(R,P) \in \Xi$ as follows: put $i \in R$ if $a_i^w - \mathbb{I}_{d_i>d}a_i^{w-d}$ is greater than $\max\{0, b_i^w - g_{iw}^d a_i^{w-d} - \bar{f}_{iw}^d b_i^{w-d}\}$, and put $i \in P$ if $b_i^w - g_{iw}^d a_i^{w-d} - \bar{f}_{iw}^d b_i^{w-d}$ is greater than $\max\{0, a_i^w - \mathbb{I}_{d_i>d}a_i^{w-d}\}$.

Then, for each (i,w,d) define $u_{iw}^d = \max\left\{0, a_i^w - \mathbb{I}_{d_i>d}a_i^{w-d}, b_i^w - g_{iw}^d a_i^{w-d} - \bar{g}_{iw}^d b_i^{w-d}\right\}$. By (2.14d), we get

$$\alpha d + \sum_{k=w-d+1}^{w} v_k \geq \sum_{i\in R} \left(a_i^w - \mathbb{I}_{d_i>d}a_i^{w-d}\right) + \sum_{i\in P} \left(b_i^w - g_{iw}^d a_i^{w-d} - \bar{g}_{iw}^d b_i^{w-d}\right) \tag{2.17a}$$

$$= \sum_{i\in N} \max\left\{0, a_i^w - \mathbb{I}_{d_i>d}a_i^{w-d}, b_i^w - g_{iw}^d a_i^{w-d} - \bar{g}_{iw}^d b_i^{w-d}\right\} = \sum_{i\in N} u_{iw}^d. \tag{2.17b}$$

Now, suppose that $(\mathbf{a}, \mathbf{b}, \mathbf{v}, \mathbf{u})$ satisfies $(2.8f), (2.8g), (2.8h)$, select any pair $(R,P) \in \Xi$ and

get

$$\alpha d + \sum_{k=w-d+1}^{w} v_k \geq \sum_{i \in N} u_{iw}^d \tag{2.18a}$$

$$\geq \sum_{i \in R} u_{iw}^d + \sum_{i \in P} u_{iw}^d \tag{2.18b}$$

$$\geq \sum_{i \in R} \left(a_i^w - \mathbb{I}_{d_i > d} a_i^{w-d} \right) + \sum_{i \in P} \left(b_i^w - g_{iw}^d a_i^{w-d} - \bar{g}_{iw}^d b_i^{w-d} \right) \tag{2.18c}$$

$$= \sum_{i \in R_d} a_i^w + \sum_{i \in \bar{R}_d} \left(a_i^w - a_i^{w-d} \right) + \sum_{i \in P} \left(b_i^w - g_{iw}^d a_i^{w-d} - \bar{g}_{iw}^d b_i^{w-d} \right). \tag{2.18d}$$

□

2.7.2 Proof of Property 2.4.2

Proof. We start proving that there exists at least one optimal solution for (2.8) satisfying $a_i^w \leq p_i$ and $b_i^w \leq g_{iw}^w p_i$ for all $i \in N$ and $w \in \mathscr{W}_0$. Choose any $i \in N$ and do forward induction on w.

- $w = 0$ is given by constraints (2.8b).
- Inductive step:

  Assume that $a_i^k \leq p_i$ and that $b_i^k \leq g_{ik}^k p_i$ for all $k < w$. We prove the statement for step w.

  Suppose that $a_i^w = p_i + \delta_a$ and $b_i^w = g_{iw}^w p_i + \delta_b$, with $\varepsilon = \max\{\delta_a, \delta_b\} > 0$. By the inductive hypothesis, (2.8c),(2.8d), (2.8f) and (2.8g) it implies that $s_{iw} \geq \varepsilon$, $u_{iw}^d \geq \varepsilon$, for all $d \in \mathscr{A}_N^w$ and by (2.8e) we have $v_w \geq \varepsilon$.

  So, update the variables for time w as follows: $a_i^w \leftarrow p_i$; $b_i^w \leftarrow b_i^w - \varepsilon$; $s_{iw} \leftarrow s_{iw} - \varepsilon$; $u_{iw}^d \leftarrow u_{iw}^d - \varepsilon$, $\forall d \in \mathscr{A}_N^w$ and $v_w \leftarrow v_w - \varepsilon$. Also, update the variables for time $v > w$: $a_i^v \leftarrow a_i^v - \varepsilon$ and $b_i^v \leftarrow b_i^v - \varepsilon$. These changes keep (2.8) feasible and the objective value does not changes (the reduction in v_w increases the objective by ε, but the change in $\mathbb{P}(\tau_i = W) a_i^W + \mathbb{P}(\tau_i < W) b_i^W$ reduces it by ε).

Now, let us show that there exists at least one optimal solution for (2.8) satisfying $a_i^w \geq 0$, $b_i^w \geq 0$ for all $i \in N$ and $w \in \mathscr{W}_0$. Choose any $i \in N$ and do forward induction on w.

- $w = 0$ is given by constraints (2.8b).
- Inductive step:

  Assume that $a_i^k \geq 0, b_i^k \geq 0$ for all $k < w$. We prove the statement for step w.

  Suppose that: $a_i^w < 0$ and/or $b_i^w < 0$. We can set these variables equal to 0 without losing feasibility. If $w < W$, then the objective remains unaltered. Else, it improves when $w = W$.

□

2.7.3 Proof of Property 2.4.3

Proof. Choose any $i \in N$. We prove by induction on w that there exists an optimal solution satisfying $a_i^w = p_i$ and $b_i^w = g_{iw}^w p_i$ for all $i \in N, w \in \mathscr{W}_0 : d_i > w$.

- $w = 0$ is given by (2.8b).
- Inductive step:

  Assume that $a_i^k = p_i$, $b_i^k = g_{ik}^k p_i, \forall k \in \mathscr{W}_0 : d_i > k$ with $k < t$ and suppose that the optimal solution is such that $a_i^w = p_i - \delta_a, b_i^w = g_{iw}^w p_i - \delta_b$, where $\max\{\delta_a, \delta_b\} > 0$. We can reassign these two variables, *i*.e., $a_i^w \leftarrow p_i$ and $b_i^k \leftarrow g_{ik}^k p_i$, keeping feasibility and without reducing the objective value. Just note that for constraints (2.8f) we have $d_i > d$ (given by $d_i > w$ and

$d \in \mathscr{A}_N^w$). Thus, all constraints involving the reassigned variables are

$$
\begin{aligned}
s_{iw} &\geq a_i^w - a_i^{w-1} = -\delta_a \\
s_{iw} &\geq b_i^w - f_{iw} a_i^{w-1} - \bar{f}_{iw} b_i^{w-1} = -\delta_b \\
s_{iw+1} &\geq a_i^{w+1} - a_i^w = a_i^{w+1} - p_i + \delta_a \\
s_{iw+1} &\geq b_i^{w+1} - f_{iw+1} a_i^w - \bar{f}_{iw+1} b_i^w = b_i^{w+1} - g_{iw+1}^{w+1} p_i + f_{iw+1}\delta_a + \bar{f}_{iw+1}\delta_b \\
u_{iw}^d &\geq a_i^w - a_i^{w-d} = -\delta_a \\
u_{iw}^d &\geq b_i^w - g_{iw}^d a_i^{w-d} - \bar{g}_{iw}^d b_i^{w-d} = -\delta_b \\
u_{iw+d}^d &\geq a_i^{w+d} - a_i^w = a_i^{w+d} - p_i + \delta_a \\
u_{iw+d}^d &\geq b_i^{w+d} - g_{iw+d}^d a_i^w - \bar{g}_{iw+d}^d b_i^w = b_i^{w+d} - g_{iw+d}^{w+d} + g_{iw+d}^d \delta_a + \bar{g}_{iw+d}^d \delta_b,
\end{aligned}
$$

and when $\delta_a, \delta_b \to 0$ the lower bounds for $\mathbf{u}$ and $\mathbf{s}$ do not increase, since $\mathbf{u}$ and $\mathbf{s}$ are nonnegative. The missing case, i.e. $b_i^w = g_{iw}^w p_i$ when $d_i = w$ follows a similar proof.

□

2.7.4 Proof of Theorem 2.4.4

For this proof we simplify our formulation to keep the intuition as simple as possible. The action set $\mathscr{A}_R^w$ in state (w, R, P) will be $\{d \in \mathbb{Z}_+ : d \leq w\}$, and so, will include possibly suboptimal actions. So, consider the stochastic DDWP

$$
\begin{aligned}
C^* = &\max_C \ \mathbb{E}_{\hat{R}} \left[C_W \left(\hat{R}, N \backslash \hat{R} \right) \right] \\
\text{s.t. } & C_0(R,P) \leq \textstyle\sum_{i \in R} p_i, && (R,P) \in \Xi \\
& C_w(R,P) \leq \mathbb{E}_{F_1^w} \left[C_{w-1} \left(R \cup F_1^w, P \backslash F_1^w \right) \right], && w \in \mathscr{W},\ (R,P) \in \Xi \\
& C_w(R,P) \leq \alpha d + \mathbb{E}_{F_d^w} \left[C_{w-d} \left(R_d \cup F_d^w, P \backslash F_d^w \right) \right], && w \in \mathscr{W},\ d \in \mathbb{Z}_+ : d \leq w,\ (R,P) \in \Xi,
\end{aligned}
$$

and its ALP bound

$$
\begin{aligned}
C^{'} = \max_{a,v,s,u} & \sum_{i\in N} \left(\mathbb{P}(\tau_i = W) a_i^W + \mathbb{P}(\tau_i < W) b_i^W\right) - \sum_{k=1}^{W} v_k \\
\text{s.t. } & a_i^0 \le p_i,\ b_i^0 \le 0, && i \in N \\
& a_i^w - a_w^{w-1} - s_{iw} \le 0, && i \in N, w \in \mathscr{W} \\
& b_i^w - f_{iw} a_i^{w-1} - \bar{f}_{iw} b_i^{w-1} - s_{iw} \le 0, && i \in N, w \in \mathscr{W} \\
& \textstyle\sum_{i\in N} s_{iw} - v_w \le 0, && w \in \mathscr{W} \\
& a_i^w - u_{iw}^d \le 0, && i \in N, w \in \mathscr{W},, d \in \{d_i, \ldots, w\} \\
& a_i^w - a_i^{w-d} - u_{iw}^d \le 0, && i \in N, w \in \mathscr{W},, d \in \{1, \ldots, \min(d_i - 1, w)\} \\
& b_i^w - g_{iw}^d a_i^{w-d} - \bar{g}_{iw}^d b_i^{w-d} - u_{iw}^d \le 0, && i \in N, w \in \mathscr{W}, d \in \{1, \ldots, w\} \\
& \textstyle\sum_{i\in N} u_{iw}^d - \sum_{k=w-d+1}^{w} v_k \le \alpha d, && w \in \mathscr{W},, d \in \{1, \ldots, w\} \\
& \mathbf{s}, \mathbf{u} \ge 0.
\end{aligned}
$$

For the deterministic case we get $\mathbb{P}(\tau_i = W) = \mathbb{I}_{(\tau_i = W)}$, $f_{iw} = \mathbb{I}_{(\tau_i = w-1)}$ and $g_{iw}^d = \mathbb{I}_{(w-d \le \tau_i < w)}$.

The ALP collapses to

$$
\begin{aligned}
C^{'} = \max_{a,v,s,u} & \sum_{i\in N} \left(a_i^W \mathbb{I}_{(\tau_i=W)} + b_i^W \mathbb{I}_{(\tau_i<W)}\right) - \sum_{k=1}^{W} v_k \\
\text{s.t. } & a_i^0 \le 0,\ b_i^0 \le 0, && i \in N \\
& a_i^w - a_w^{w-1} - s_{iw} \le 0, && i \in N, w \in \mathscr{W} \\
& b_i^w - b_i^{w-1} - \mathbb{I}_{(w-1\le\tau_i<w)}(a_i^{w-1} - b_i^{w-1}) - s_{iw} \le 0, && i \in N, w \in \mathscr{W} \\
& \textstyle\sum_{i\in N} s_{iw} - v_w \le 0, && w \in \mathscr{W} \\
& a_i^w - u_{iw}^d \le 0, && i \in N, w \in \mathscr{W}, d \in \{d_i, \ldots, w\} \\
& a_i^w - a_i^{w-d} - u_{iw}^d \le 0, && i \in N, w \in \mathscr{W},, d \in \{1, \ldots, \min(d_i - 1, w)\} \\
& b_i^w - b_i^{w-d} - \mathbb{I}_{(w-d\le\tau_i<w)}(a_i^{w-d} - b_i^{w-d}) - u_{iw}^d \le 0, && i \in N, w \in \mathscr{W}, d \in \{1, \ldots, w\} \\
& \textstyle\sum_{i\in N} u_{iw}^d - \sum_{k=w-d+1}^{w} v_k \le \alpha d, && w \in \mathscr{W}, d \in \{1, \ldots, w\} \\
& \mathbf{s}, \mathbf{u} \ge 0.
\end{aligned}
$$

From this point we assume without loss of generality that $d_i \le \tau_i$. Otherwise, we can transform the model to an equivalent one satisfying this requirement. If request i does not arrive ($\tau_i < 0$), the optimal ALP value does not get altered by removing it, since at optimality $a_i^w = b_i^w = 0,\ \forall w \in \mathscr{W}_0$. In case that $0 < \tau_i < d_i$, i.e. the order arrives but cannot be served, one optimal solution is $a_i^w = p_i,\ \forall w \in \mathscr{W}_0$ and may be removed from the analysis by adding a constant p_i to the objective.

Now, let us preset some variables in the ALP:

- $a_i^w = 0$, for all $i \in N, w > \tau_i$; the open order cost before arrival is zero.
- $b_i^w = a_i^{\tau_i}$, for all $i \in N, w > \tau_i$; the potential order cost before arrival is equal to the open order cost upon arrival.
- $b_i^w = 0$, for all $i \in N, w \le \tau_i$, i.e., the potential order cost after arrival is zero.

We have restricted the feasible space, and thus the remaining model is still an underestimate of

C^* given by

$$
\begin{aligned}
C'' = \max_{a,v,s,u} \; & \sum_{i\in N} a_i^{\tau_i} - \sum_{k=1}^{W} v_k \\
\text{s.t. } (x) \;\; & a_i^w - a_w^{w-1} - s_{iw} \le 0, && i \in N, w \in \{1,\dots,\tau_i\} \\
(m) \;\; & a_i^w - u_{iw}^d \le 0, && i \in N, w \in \{d_i,\dots,\tau_i\}, d \in \{d_i,\dots,w\} \\
(\alpha) \;\; & a_i^w - a_i^{w-d} - u_{iw}^d \le 0, && i \in N, d \in \{1,\dots,d_i-1\}, w \in \{d,..,\tau_i\}, \\
(\beta) \;\; & a_i^{\tau_i} - a_i^{w-d} - u_{iw}^d \le 0, && i \in N, w \in \{\tau_i+1,\dots,W\}, d \in \{t-\tau_i,\dots,w\} \\
(\gamma) \;\; & a_i^0 \le p_i, && i \in N \\
(Z) \;\; & \sum_{i\in N} s_{iw} - v_w \le 0, && w \in \mathscr{W} \\
(Y) \;\; & \sum_{i\in N} u_{iw}^d - \sum_{k=w-d+1}^{w} v_k \le d, && w \in \mathscr{W}, d \in \{1,\dots,w\} \\
& \mathbf{s}, \mathbf{u} \ge 0,
\end{aligned}
$$

and its dual problem is

$$C'' = \min_{Z,Y,\alpha,\beta,\gamma,m,x\geq 0} \sum_{i\in N} p_i\gamma_i + \sum_{w=1}^{W}\sum_{d=1}^{w} dY_{w,d} \tag{2.23a}$$

$$\text{s.t. } (v)\ Z_w + \sum_{w'=w}^{W}\sum_{d=w'-w+1}^{w'} Y_{w',d} = 1, \qquad w \in \mathscr{W} \tag{2.23b}$$

$$(s)\ x_i^w \leq Z_w, \qquad i \in N, w \in \{1,\ldots,\tau_i\} \tag{2.23c}$$

$$(u)\ m_{i,w}^d \leq Y_{w,d}, \qquad i \in N, w \in \{d_i,\ldots,\tau_i\}, \quad d \in \{d_i,\ldots,w\} \tag{2.23d}$$

$$\alpha_{i,w}^d \leq Y_{w,d}, \qquad i \in N, d \in \{1,..,d_i-1\}, \quad w \in \{d,\ldots,\tau_i\} \tag{2.23e}$$

$$\beta_{i,w}^d \leq Y_{w,d}, \qquad i \in N, w \in \{\tau_i+1,\ldots,W\}, \quad d \in \{w-\tau_i,\ldots,w\} \tag{2.23f}$$

$$(a)\ \gamma_i = \left(x_i^1 + \textstyle\sum_{k=1+\tau_i}^{W} \beta_{i,k}^k + \sum_{k=1}^{d_i-1} \alpha_{i,k}^k\right), \qquad i \in N, \quad w = 0 \tag{2.23g}$$

$$\left(x_i^w + \textstyle\sum_{d=1}^{w} \alpha_{i,w}^d\right) = \left(x_i^{w+1} + \sum_{k=1+\tau_i}^{W} \beta_{i,k}^{k-w} + \sum_{k=1}^{d_i-1} \mathbb{I}_{(k+w\leq\tau_i)}\alpha_{i,k+w}^k\right), \qquad i \in N, \quad w \in \{1,\ldots,d_i-1\} \tag{2.23h}$$

$$\left(x_i^w + \textstyle\sum_{d=d_i}^{w} m_{i,w}^d + \sum_{d=1}^{d_i-1} \alpha_{i,w}^d\right) = \left(x_i^{w+1} + \sum_{k=1+\tau_i}^{W} \beta_{i,k}^{k-w} + \sum_{k=1}^{d_i-1} \mathbb{I}_{(k+w\leq\tau_i)}\alpha_{i,k+w}^k\right), \qquad i \in N, \quad w \in \{d_i,\ldots,\tau_i-1\} \tag{2.23i}$$

$$\left(x_i^{\tau_i} + \textstyle\sum_{d=d_i}^{\tau_i} m_{i,\tau_i}^d + \sum_{d=1}^{d_i-1} \alpha_{i,w}^d\right) + \left(\sum_{k=\tau_i+1}^{W}\sum_{d=k-\tau_i+1}^{k} \beta_{i,k}^d\right) = 1, \qquad i \in N, \quad w = \tau_i. \tag{2.23j}$$

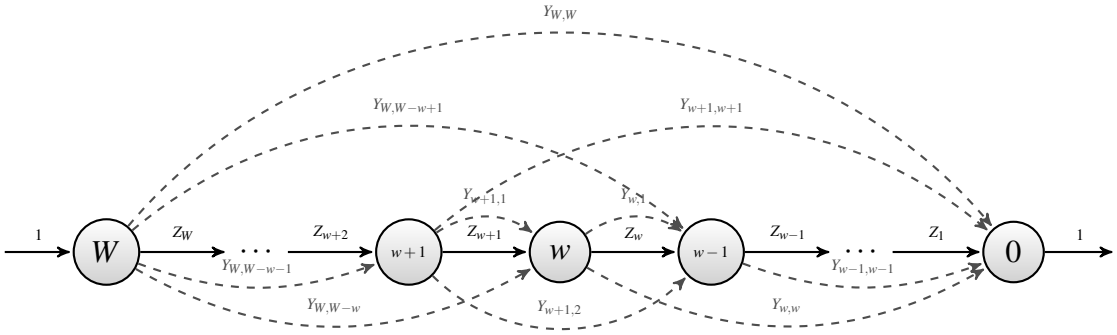


Figure 2.8: Network Structure in (Z,Y)-domain

Consider (2.23):

1. First note that (2.23b) are equivalent to the following network flow balance constraints

$$1 = Z_W + \sum_{d=1}^{W} Y_{W,d} \tag{2.24a}$$

$$Z_{w+1} + \sum_{w'=w+1}^{W} Y_{w',w'-w} = Z_w + \sum_{d=1}^{w} Y_{w,d}, \qquad w \in \{1,..,W-1\} \tag{2.24b}$$

$$Z_1 + \sum_{w'=2}^{W} Y_{w',w'-1} = 1 \tag{2.24c}$$

$$Z, Y \geq 0, \tag{2.24d}$$

represented in Figure 2.8.

Proof. Equivalence is obtained by subtracting constraint w from constraint $w+1$ in (2.23b) for all $w \in \{W-1,\ldots,2\}$. The flow balance constraint at node $w = W$ comes explicitly, and the flow balance constraint at node $w = 1$ is obtained by adding the previously derived equations. □

Therefore, substructure (2.24) has integral extreme points.

2. Now, let us study the remaining constraints. Note that for a given (Z,Y) the resulting problem in variables $(\alpha,\beta,\gamma,x,m)$ collapses to n independent capacitated minimum cost network flow

problems (CMCNF) for each order $i \in N$ defined in (2.25)

$$\gamma_i(Z,Y) = \min_{\alpha_i,\beta_i,\gamma_i,m_i,x_i \geq 0} \gamma_i \tag{2.25a}$$

$$\text{s.t. } (2.23c),(2.23d),(2.23e),(2.23f),(2.23g),(2.23h),(2.23i),(2.23j). \tag{2.25b}$$

In this network there is a set of nodes given by $\{0,\ldots,\tau_i\}$ and a sink node S^i defined by the (redundant) flow balance constraint $\gamma_i + \sum_{w=d_i}^{\tau_i} \sum_{x=d_i}^{w} m_{i,w}^d = 1$ obtained when adding (2.23g),(2.23h),(2.23i) and (2.23j). We would like to minimize the cost of moving one unit of flow from node τ_i to the sink node. There are five arc types available in (2.25) given by

- Type 1 arc (γ_i) going from node 0 to S^i. Our objective is to minimize the value of this flow, since it is the only one with non-zero cost.
- Type 2 arcs ($m_{i,w}^d$) going from node $w \in \{d_i,\ldots,\tau_i\}$ to S^i. We want to maximize these flows, but these arc flows are bounded by $Y_{w,d}$.
- Type 3 arcs (x_i^w) going from w to $w-1$ for each $w \in \{1,\ldots,\tau_i\}$. These flows are bounded by Z_w.
- Type 4 arcs ($\alpha_{i,w}^d$) going from a node $w \in \{1,\ldots,\tau_i\}$ to any node $w-d$ for each $d < d_i$ and $d \leq w$; also bounded by $Y_{w,d}$.
- Type 5 arcs ($\beta_{i,k}^d$) going from node τ_i to any node $w \in \{0,\ldots,\tau_i - 1\}$ for each $k \in \mathbb{Z}_+$ and $d \in \mathbb{Z}_+$ satisfying $\tau_i < k \leq W$ and $k - d = w$; also bounded by $Y_{k,d}$.

Note that problem (2.25) is feasible for any value $(Z,Y) \in (2.24)$. Its network is graphically represented in Figure 2.9.

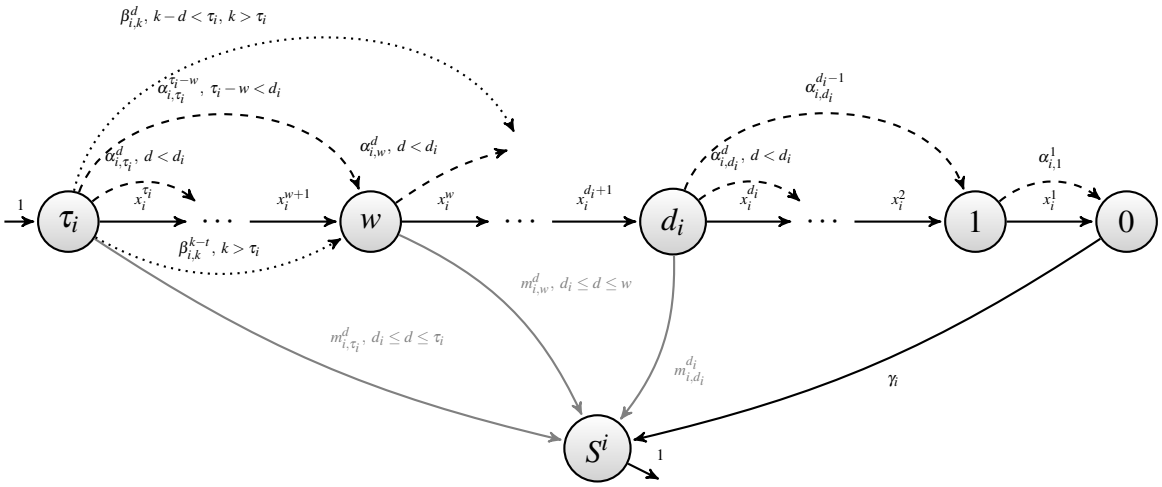


Figure 2.9: Network for i^{th} order subproblem.

If we put these two comments together, the dual ALP in (2.23) is equal to

$$\min_{(Z,Y)\in(2.24)} C(Z,Y) := \underbrace{\sum_{i\in N} p_i\gamma_i(Z,Y)}_{P(Z,Y)} + \underbrace{\sum_{w=1}^{W}\sum_{d=1}^{w} dY_{w,d}}_{C_{OP}(Y)}. \tag{2.26}$$

We show in two parts that (2.26) has an optimal value equal to the optimal cost of the deterministic DDWP in (2.4). First, we prove that any feasible dispatch for the deterministic DDWP has a one-to-one mapping with integer feasible solutions (Z,Y) to (2.26). Then, we show that without loss of optimality a solution of (2.26) can be assumed integral.

Part 1: Consider any feasible dispatch with lengths $\{d^1,...,d^K\}$ and dispatch times $\{w^1,..,w^K\}$. Then, there is a unique integer solution of (Z,Y) representing this operation. Just set to zero all components of Y except for $Y_{w^k,d^k}=1, \forall k\in\{1,\ldots,K\}$ and set Z to satisfy (2.23b). Thus, $Y_{w,d}$ represents a dispatch at w with distance length d and Z_w represents waiting at the depot between w and $w-1$. Its corresponding operational dispatch cost matches the second term in the objective of

(2.26), i.e. $C_{OP}(Y) := \sum_{w=1}^{W} \sum_{d=1}^{w} dY_{w,d} = \sum_{k=1}^{K} d^k$. Also, let

$$\eta_i = \begin{cases} 1 & \text{if } d^k \geq d_i \text{ and } w^k \leq \tau_i, \text{ for some } k \in \{1,\ldots,K\} \\ 0 & \text{otherwise} \end{cases}$$

indicate whether order i is covered by any dispatch or not. If $\eta_i = 0$, then all type 2 arcs for subproblem (2.25) cannot be used, i.e. $m_{i,w}^d \leq 0$ for $d_i \leq w \leq \tau_i$ and $d_i \leq d \leq w$, so there is a unique path from τ_i to S^i with $\gamma_i = 1$. If $\eta_i = 1$, then a new $(\tau_i - S^i)$-path arises with capacity one. The idea is to move the unit flow horizontally using type 3 arcs ($x_i^w = 1$) at each node $w : 1 \leq w \leq \tau_i$ when $Z_w = 1$. Otherwise, if $Z_w = 0$ there are three potential scenarios:

- A dispatch at w covers i, i.e., $d \geq d_i$. Then we can use the corresponding type 2 arc $m_{i,w}^d = Y_{w,d} = 1$ and reach the sink node S^i at zero cost ($\gamma_i = 0$).

- A dispatch at w does not cover i, i.e. $d < d_i$. Then we can use the corresponding type 4 arc $\alpha_{i,w}^d = Y_{w,d} = 1$ and reach node $w - d$ at zero cost. Since $\eta_i = 1$, we proceed until we find the type 2 arc associated with the earliest dispatch that covers i.

- We have $w = \tau_i$ and there is a dispatch at time $k > \tau_i$ with distance d such that $k - d < \tau_i$. Then we can send one unit of flow in a type 5 arc to node $k - d$, i.e., $b_{i,k}^d = Y_{k,d} = 1$. Again, we proceed until we find the earliest type 2 arc.

The first cost term in (2.26) will be exactly equal to the penalties paid for orders left unattended: $P(Z,Y) := \sum_{i \in N} p_i \gamma_i = \sum_{i \in N:\ n_i = 0} p_i$.

Part 2: Now we prove that without loss of optimality Z, Y is binary, and hence an optimal solution is an optimal dispatch for the deterministic DDWP. Assume by contradiction that Y has fractional components and that $C(Z,Y) < C(\bar{Z},\bar{Y})$ for any integral solution $(\bar{Z},\bar{Y}) \in (2.24)$. We can express (Z,Y) as a convex combination of the extreme points $(Z^1,Y^1),\ldots,(Z^p,Y^p)$ of (2.24)

which are binary. Thus, we have $(Z,Y) = \sum_{l=1}^{p} \lambda_l (Z^l, Y^l)$ for a given nonnegative vector $\lambda \geq 0$ such that $\sum_{l=1}^{p} \lambda_l = 1$. The operational cost term $C_{OP}(Y)$ in (2.26) is additive in Y, since

$$C_{OP}(Y) = \sum_{w=1}^{W} \sum_{d=1}^{w} dY_{w,d} = \sum_{w=1}^{W} \sum_{d=1}^{w} d \left(\sum_{l=1}^{p} \lambda_l Y^l_{w,d} \right) = \sum_{l=1}^{p} \lambda_l \left(\sum_{w=1}^{W} \sum_{d=1}^{w} dY^l_{w,d} \right) = \sum_{l=1}^{p} \lambda_l C_{OP}(Y^l).$$

So, if (Z,Y) satisfies for each $i \in N$ that

$$\gamma_i(Z,Y) = \sum_{l=1}^{p} \lambda_l \gamma_i(Z^l, Y^l), \tag{2.27}$$

then the additive relation follows for the penalty cost term $P(Z,Y)$ in (2.26), because

$$P(Z,Y) = \sum_{i \in N} \gamma_i(Z,Y) p_i = \sum_{i \in N} \left(\sum_{l=1}^{p} \lambda_l \gamma_i(Z^l, Y^l) p_i \right) = \sum_{l=1}^{p} \lambda_l \left(\sum_{i \in N} \gamma_i(Z^l, Y^l) p_i \right) = \sum_{l=1}^{p} \lambda_l P(Z^l, Y^l),$$

and the total cost is additive in (Z,Y), i.e. $C(Z,Y) = \sum_{l=1}^{p} \lambda_l C(Z^l, Y^l)$. So, if condition (2.27) is true, the optimal cost is a convex combination of binary extreme point costs and it directly implies that there should be an integer extreme point l^* satisfying $C(Z^{l^*}, Y^{l^*}) \leq C(Z,Y)$. This is our desired contradiction.

Proof of condition (2.27): Note that the condition $\gamma_i(Z,Y) \leq \sum_{l=1}^{p} \lambda_l \gamma_i(Z^l, Y^l)$ is trivial, since the optimal value of (2.25) is a convex function of the right-hand-side argument (Z,Y). Also, we have that $\gamma_i(Z^l, Y^l) = 1$ when the operation encoded in Y^l covers order i, else it is equal to 0. So, the right-hand-side of (2.27) yields $\sum_{l=1}^{p} \lambda_l \gamma_i(Z^l, Y^l) = 1 - \sum_{l:Y^l \text{ covers } i} \lambda_l$. We need to show that the left-hand-side of (2.27) is also equal to the above value. There is two cases:

1. Suppose that for each $l \in \{1, \ldots, p\}$ with $0 < \lambda_l < 1$, the operation encoded in Y^l covers order $i \in N$ at most in one dispatch. In case that Y^l covers i exactly once, then $(\lambda_l Y^l, \lambda_l Z^l)$ will add in (2.25) exactly one type 2 arc $m^d_{i,w}$ with capacity $\lambda_l > 0$, where $Y^l_{w,d}$ is such that $d_i \leq d$ and $\tau_i \geq w$. Also, $(\lambda_l Y^l, \lambda_l Z^l)$ will produce a zero cost path from τ_i to S^i with capacity

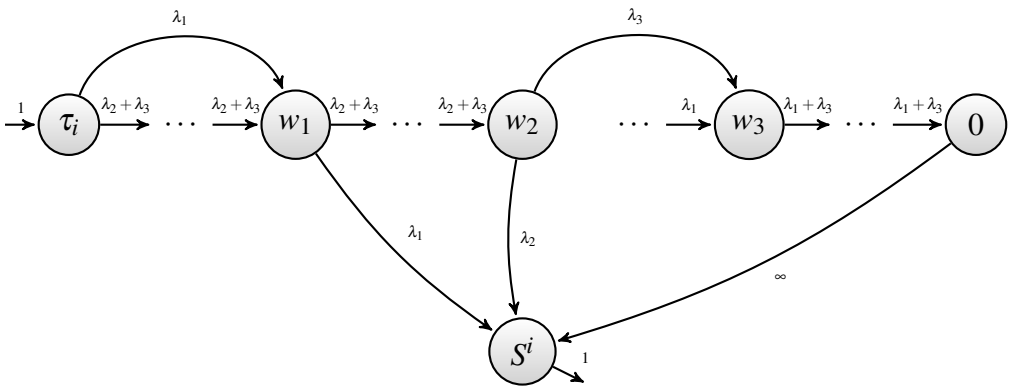


Figure 2.10: Example of a convex combination of three operations in i^{th} order subproblem.

λ_l that uses arc $m^d_{i,w}$. On the other hand, if Y^l does not cover i there will be no additional paths to S^i. If we put all these solutions Y^l together for each $l \in \{1,\ldots,p\}$ with $0 < \lambda_l < 1$ and form $Y = \sum_{l=1}^{p} \lambda_l Y^l$, the binding cut between τ_i and S^i with zero-cost flows will be defined by $U = \{1,\ldots,\tau_i\}$ with capacity $\sum_{l:Y^l \text{ covers } i} \lambda_l$. So, given that the cut is always binding, if we put these paths together in one single network it does not affect the output and $\gamma_i(Z,Y) = 1 - \sum_{l:Y^l \text{ covers } i} \lambda_l$. Figure 2.10 provides an example of this network showing the arc capacities of subproblem (2.25) for order i. This case has three integer extreme points Y: Y^1, Y^2 and Y^3 defining $Y = \lambda_1 Y^1 + \lambda_2 Y^2 + \lambda_2 Y^3$ and $1 = \lambda_1 + \lambda_2 + \lambda_2$ for $\lambda \geq 0$. Y^1 and Y^2 cover order i, but Y^3 does not. It is clear that the maximum zero-cost flow from τ_i to S^i is equal to the capacity of the cut U equal to $\lambda_1 + \lambda_2 < 1$. So, $\gamma_i = 1 - \lambda_1 - \lambda_2$.

2. A potential problem could occur if an operation covers an order more than once in multiple dispatches. For example, suppose that there exists an operation l^1 with $0 < \lambda^{l^1} < 1$ such that Y^{l^1} covers order i twice and that there exists another operation l^2 not covering i such that the vehicle is at the depot when operation l^1 dispatches the latest dispatch covering i. Then, an "artificial" coverage is created for order i. Figure 2.11 illustrates this problem. In this example, operation $l = 1$ with weight $\lambda_1 = 0.5$ waits at the depot until t_1, covers order i at t_1,

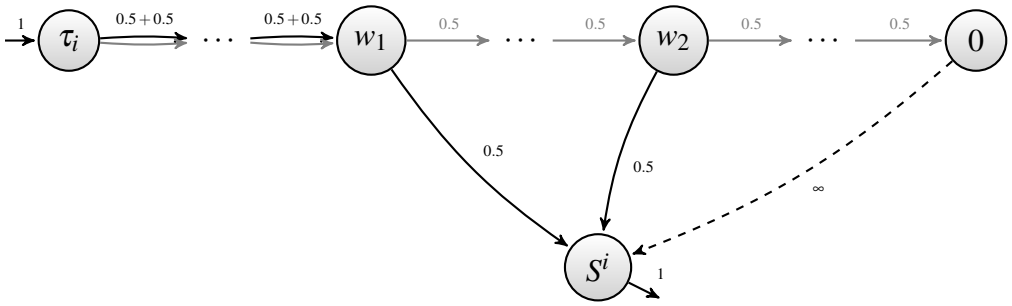


Figure 2.11: Example of a convex combination of two operations where subproblem for order i is not additive in the argument (Z,Y).

returns at t_2 and covers order i again at t_2. Operation $l = 2$ with weight $\lambda_2 = 0.5$ waits at the depot all the time (between τ_i and 0). We have that $0.5\gamma_i(Z^1,Y^1) + 0.5\gamma_i(Z^2,Y^2) = 0.5$, but $\gamma_i\left(0.5(Z^1,Y^1) + 0.5(Z^2,Y^2)\right) = 1 - 0.5 - \min\{0.5,0.5\} = 0$. So condition (2.27) does not hold. Fortunately, we can prove that there exists an alternative set of operations $l \in E$ such that Y can also be written as $Y = \sum_{l\in E} \lambda_l Y^l$ and such that condition (2.27) holds.

Let us solve this problem for the example in Figure 2.11 first. Define Y^3 and Y^4 as follows. Let

$$Y^3_{w,d} := \begin{cases} Y^1_{w,d} & w > w_2, 1 \le d \le w \\ Y^2_{w,d} & w \le w_2, 1 \le d \le w \end{cases} \text{ and } Y^4_{w,d} := \begin{cases} Y^2_{w,d} & w > w_2, 1 \le d \le w \\ Y^1_{w,d} & w \le w_2, 1 \le d \le w \end{cases}.$$

Note that $Y = 0.5Y^3 + 0.5Y^4$ and, thus, this new decomposition does not affect operational costs. Also, it covers the same amount of orders plus the "artificial" coverage which is now valid. So $0.5\gamma_i(Z^3,Y^3) + 0.5\gamma_i(Z^4,Y^4) = \gamma_i\left(0.5(Z^3,Y^3) + 0.5(Z^4,Y^4)\right) = 0$. Figure 2.12 presents this solution.

The general proof can be constructed by induction on $r_1 + r_2$, where r_1 is the total number of additional dispatches covering $i \in N$ in operations inside S, and r_2 is the number of operations

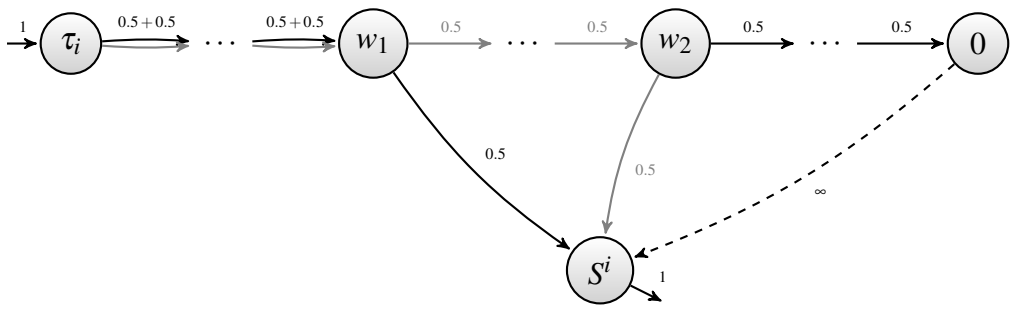


Figure 2.12: Same example with two operations where subproblem for order i is additive in the argument (Z,Y).

not covering i in S with the vehicle at the depot at a time w^* where another operation $l' \in S$ executes a dispatch covering i which is not the earliest such dispatch.

- Case $r_1 = 0, r_2 = 0$: This case is trivial, since the set $S : Y = \sum_{l \in S} \lambda_l Y^l$ satisfies (2.27).
- Case $r_1 > 0, r_2 = 0$: This case is also trivial, since the multiple dispatches cannot be used to generate "artificial coverages" and any S such that $Y = \sum_{l \in S} \lambda_l Y^l$ satisfies (2.27).
- Case $r_1 = 0, r_2 > 0$: This case is impossible, by the definition of r_2 ($r_1 = 0 \implies r_2 = 0$).
- Case $r_1 > 0, r_2 > 0$: Let $l^1 \in S$ be the operation with a repeated dispatch to i at time t^* such that there exists another operation $l^2 \in S$ not covering i and with the vehicle available at the depot at time w^*. Construct two new operations l^3 and l^4 as follows:

$$Y^{l^3}_{w,d} := \begin{cases} Y^{l^1}_{w,d} & w > w^*, 1 \le d \le w \\ Y^{l^2}_{w,d} & w \le w^*, 1 \le d \le w \end{cases} \text{ and } Y^{l^4}_{w,d} := \begin{cases} Y^{l^2}_{w,d} & w > w^*, 1 \le d \le w \\ Y^{l^1}_{w,d} & w \le w^*, 1 \le d \le w \end{cases}.$$

  We have three cases:

  - If $\lambda_{l^1} < \lambda_{l^2}$, we have $Y = \sum_{l \in S \setminus \{l^1, l^2\}} \lambda_{l^1} Y^l + \lambda_{l^1}(Y^{l^3} + Y^{l^4}) + (\lambda_{l^2} - \lambda_{l^1}) Y^{l^2}$ and r_1 decreases by one. Use induction with $S' = S \setminus \{l^1\} \cup \{l^3, l^4\}$.
  - If $\lambda_{l^2} < \lambda_{l^1}$, we have $Y = \sum_{l \in S \setminus \{l^1, l^2\}} \lambda_{l^1} Y^l + \lambda_{l^2}(Y^{l^3} + Y^{l^4}) + (\lambda_{l^1} - \lambda_{l^2}) Y^{l^1}$ and r_2

decreases by one. Use induction with $S' = S \setminus \{l^2\} \cup \{l^3, l^4\}$.

– If $\lambda_{l^2} = \lambda_{l^1}$, set $Y = \sum_{l \in S \setminus \{l^1, l^2\}} \lambda_{l^1} Y^l + \lambda_{l^1}(Y^{l^3} + Y^{l^4})$ and r_1 and r_2 each decrease by one. Use induction with $S' = S \setminus \{l^1, l^2\} \cup \{l^3, l^4\}$.

□

2.7.5 ALP solution pruning

We can reduce the computational effort involved in getting the ALP optimal policy defined by (2.10) with the following proposition:

Proposition 2.7.1 (ALP solution pruning)**.** *Suppose $\delta \in \mathscr{A}_R^w$ is a feasible dispatch distance at state (w, R, P) and its related ALP solution to* (2.10) *is $\{\mathbf{a}(\delta), \mathbf{b}(\delta), \mathbf{v}(\delta)\}$. Let $\mu \in \mathscr{A}_R^w$ be a different feasible dispatch distance. If*

$$\begin{aligned} &\alpha\delta + \textstyle\sum_{i \in R_\delta} a_i(\delta)^{w-\delta} + \sum_{i \in P} \left(g_{iw}^{\delta} a_i(\delta)^{w-\delta} + \bar{g}_{iw}^{\delta} b_i(\delta)^{w-\delta} \right) - \sum_{k=1}^{w-\delta} v_k(\delta) \\ &< \alpha\mu + \textstyle\sum_{i \in R_\mu} a_i(\delta)^{w-\mu} + \sum_{i \in P} \left(g_{iw}^{\mu} a_i(\delta)^{w-\mu} + \bar{g}_{iw}^{\mu} b_i(\delta)^{w-\mu} \right) - \sum_{k=1}^{w-\mu} v_k(\delta), \end{aligned}$$

then μ is suboptimal for (2.10) *and can be discarded before solving its related ALP.*

Proof. The proof is based on the fact that $\{\mathbf{a}(\delta), \mathbf{b}(\delta), \mathbf{v}(\delta)\}$ is also a feasible solution for the ALP problem related to μ. By proposition (2.7.1) and the feasibility of $\mathbf{a}(\delta), \mathbf{b}(\delta), \mathbf{v}(\delta)$ in any ALP problem we get

$$\begin{aligned} &\alpha\delta + \sum_{i \in R_\delta} a_i(\delta)^{w-\delta} + \sum_{i \in P} \left(g_{iw}^{\delta} a_i(\delta)^{w-\delta} + \bar{g}_{iw}^{\delta} b_i(\delta)^{w-\delta} \right) - \sum_{k=1}^{w-\delta} v_k(\delta) \\ &< \alpha\mu + \sum_{i \in R_\mu} a_i(\delta)^{w-\mu} + \sum_{i \in P} \left(g_{iw}^{\mu} a_i(\delta)^{w-\mu} + \bar{g}_{iw}^{\mu} b_i(\delta)^{w-\mu} \right) - \sum_{k=1}^{w-\mu} v_k(\delta) \\ &\leq \alpha\mu + \max_{\{(\mathbf{a},\mathbf{b},\mathbf{v}) \in (2.8b)-(2.8i)\}} \sum_{i \in R_\mu} a_i^{w-\mu} + \sum_{i \in P} \left(g_{iw}^{d} a_i^{w-\mu} + \bar{g}_{iw}^{d} b_i^{w-\mu} \right) - \sum_{k=1}^{w-\mu} v_k, \end{aligned}$$

and this proves that the dispatch distance δ yields a lower approximate expected cost than μ for the ALP policy. □

CHAPTER 3

THE DYNAMIC DISPATCH WAVES PROBLEM FOR SAME-DAY DELIVERY

3.1 Introduction

In this chapter, we study the general Dynamic Dispatch Waves Problem (DDWP) that captures the fundamental tradeoffs in dynamic dispatch decision-making and extends the one-dimensional variant proposed in [44] to a general network topology. The DDWP is an order delivery problem with dynamic dispatch and routing decisions for a single vehicle operating over a fixed-duration operating period (*i*.e., a day) partitioned in W dispatch *waves*. Each dispatch wave can be thought of as a point in time when picking and packing of a set of orders is completed, and a vehicle (or vehicles) can be loaded for dispatch. In this research, a dispatch wave represents a decision epoch where a single vehicle (if available at the depot) can wait for the next wave, or alternatively be loaded and dispatched from the depot to serve a subset of *open* customer delivery orders. Open orders are defined as those ready to be dispatched and not previously served. At each wave decision epoch, complete information is known for all open orders and probabilistic information is available describing potential future orders. After the vehicle completes a dispatch route, it returns to the depot and is ready to be dispatched again. The objective is to minimize vehicle operating costs and penalties for open orders that remain unserved at the end of the operating period. In this paper, we extend the model in [44] to a general network topology to make it compatible with same-day delivery operations using a typical road network.

The DDWP presents interesting challenges because of two fundamental tradeoffs discussed in [44]. First, there is a tradeoff between waiting and dispatching a vehicle. When a vehicle is dispatched, the set of open orders waiting to be served is reduced, but the opportunity to observe and serve future orders arriving geographically nearby to ones in the current route is lost. Conversely,

when the vehicle is not dispatched, the time remaining to serve the set of open and future orders is reduced. Second, there is a tradeoff between dispatching longer routes serving many orders versus shorter ones with fewer visits. The former consume more time and keep the vehicle away from the depot longer, but require less time per customer visited due to density economies. Shorter routes require more time per customer, but enable the vehicle to be reused sooner.

We consider the following to be our primary contributions.

1. We formulate a natural model for the deterministic variant of our problem, which we leverage to provide **lower bounds** for the stochastic-dynamic case via information relaxation and simulation.

2. We use the deterministic model to find an **optimal *a priori* solution** to the stochastic variant, by showing that the *a priori* optimization problem is equivalent to a deterministic instance with known customer order arrival times and adjusted penalties. We design construction and local search heuristics to complement commercial MIP solvers to speed up the identification of solutions to problem instances.

3. We then provide three approaches to obtain **dynamic policies** using the *a priori* model. The first uses a rollout scheme to dispatch according to the *a priori* solution and iteratively update it with new information, and the latter approaches are based on fast heuristic modifications to the initial *a priori* solution.

4. We empirically show the benefits of dynamic policies with computational experiments that suggest that the marginal improvement provided by dynamic policies both in cost reduction and in optimality gap improvements of our dynamic policies over an *a priori* one in terms of cost reduction and duality gap are important for instances with greater order arrival variability and less information disclosed before the start of the operation. We also see that the quality of a dynamic solution has less variability over all instances tested.

5. Finally, we empirically analyze the tradeoff between two common objectives in SDD: minimizing total costs (including vehicle travel time) versus maximizing order coverage. One might think that these two objectives deliver similar results, since well-sequenced routes leave more vehicle time to cover additional orders. However, we find that there are fundamental differences in the solution's structure for both cases in terms of number of vehicles dispatched, route length and initial wait at the depot, that one should expect significant sacrifices in vehicle routing efficiency in order to maximize the order fill rate, and that the distance cost of an additional customer covered becomes more expensive as order coverage increases.

The remainder of the chapter is organized in the following manner. Section 3.2 defines the notation and formulates the DDWP, Section 3.3 covers the deterministic problem, and Sections 3.4 and 3.5 respectively cover *a priori* and dynamic policies. Finally, Section 3.6 outlines the results of a computational study, and we conclude with Section 3.7.

3.2 DDWP problem formulation

We formally define the DDWP as a Markov Decision Process (MDP); see the text [57] for a reference on MDPs. We start by describing the notation and elements of our model:

1. **Operating period.** Let $\mathscr{W} := \{1,\ldots,W\}$ be the set of *waves* (decision epochs), each with equal time duration ℓ. The number $w \in \mathscr{W}$ represents the "waves-to-go" before $w = 0$, the deadline for the vehicle to finish all deliveries and return to the depot.

2. **Customer orders and geography.** Let $N := \{1,\ldots,n\}$ be the set of all *potential* customer orders. Each $i \in N$ and the depot ($i = 0$) define known locations represented by a complete undirected graph $\mathscr{G} = (N \cup \{0\}, E)$, where E is the set of edges. We assume that the vehicle takes t_e time and spends c_e to traverse an edge $e \in E$; we assume for simplicity that time and cost values are proportional to each other ($c_e = \gamma t_e$), non-negative, and satisfy the triangle inequality.

3. **Order ready times.** Each order i is ready for dispatch at a random wave $\tau_i \in \mathscr{W} \cup \{-1\}$ drawn from an order dependent distribution; if $\tau_i = -1$ the orders i does not instantiate. We assume that these ready times are independent between different orders, and that the set $\{i \in N : \tau_i \geq w\}$ is known at wave w.

4. **Penalties.** Each $i \in N$ has a non-negative penalty β_i to be paid if order i realizes and is left unattended by $w = 0$.

We now are ready to formulate an MDP model for the DDWP. If the vehicle is available at the depot at wave w, the system state is (w, R, P), where $R \subseteq N$ is the set of open orders and $P \subseteq N$ is the set of remaining *potential* orders with an unknown arrival time ($\tau_i < w$). Orders in $N \setminus \{R \cup P\}$ are "closed", meaning they were ready and served before wave w. The pair (R, P) includes two disjoint subsets of N, *i*.e., $(R, P) \in \Xi := \{(R, P) : R, P \subseteq N, R \cap P = \emptyset\}$. The maximum number of waves W and three possible states for each order (open-closed-potential) define an $\mathscr{O}(3^n W)$ bound on the cardinality of the state space.

An action at any non-terminal state $(w, R, P) : w \geq 1$ is defined as a vehicle dispatch that serves a subset of open orders $S \subseteq R$; $S = \emptyset$ represents waiting at the depot. The set S completely defines the action and takes $t(S)$ time, *i*.e., the minimum time required by any tour to visit S and return to the depot (a Traveling Salesman Problem (TSP) tour over $S \cup \{0\}$). Once dispatched, the vehicle cannot serve any other order until it returns for reloading at wave $q_w(S) := w - \max\{1, \lceil t(S)/\ell \rceil\}$, and S is constrained such that the vehicle returns before the end of the day, $q_w(S) \geq 0$. The subsets $S \subset R$ imply $\mathscr{O}(2^n)$ possible actions. Selecting an action S in state (w, R, P) produces a transition to state $(q_w(S), (R \setminus S) \cup F_w(S), P \setminus F_w(S))$ where $F_w(S) := \{i \in N : w > \tau_i \geq q_w(S)\}$ is the random set of newly arriving orders in waves $w - 1, \ldots, q_w(S)$, and $R \setminus S$ is the set of orders in R left open by action S.

Let $C_w(R, P)$ be a set function representing the minimum expected cost-to-go at state (w, R, P) and let $C^*(\hat{R}) := C_W\left(\hat{R}, N \setminus \hat{R}\right)$ be the minimum expected cost depending on $\hat{R}$, a given set of orders ready at $w = W$ (the order information known at the start of the operation). The dynamic program

defined in (3.1) computes $C^*(\hat{R})$ recursively over w. Any terminal cost $C_0(R,P)$ is equal to the sum of penalties of unserved open orders R, and subsequently, for each $w \in \mathscr{W}$ the cost-to-go at state (w,R,P) is equal to the minimum cost between no dispatch and any feasible dispatch. Define an optimal action as a subset $S_w(R,P) \subseteq R$ that attains $C_w(R,P)$ at a given state (w,R,P) and an optimal policy as a vector of optimal actions for each possible state of the system.

$$C_0(R,P) = \textstyle\sum_{i\in R} \beta_i, \qquad \forall (R,P) \in \Xi \quad (3.1a)$$

$$C_w(R,P) = \min_{S\subseteq R: q_w(S)\geq 0} \left\{ t(S) + \mathbb{E}_{F_w(S)} \left[C_{q_w(S)} \left(R \setminus S \cup F_w(S), P \setminus F_w(S) \right) \right] \right\}, \quad \forall w \in \mathscr{W}, \forall (R,P) \in \Xi. \quad (3.1b)$$

The model (3.1) shows the difficulty in finding an optimal policy; it has exponentially many states ($\mathscr{O}(3^n W)$), exponentially many actions per state ($\mathscr{O}(2^n)$), and exponentially many terms in the expectations ($\mathscr{O}(2^n)$). Moreover, it is NP-Hard to evaluate the cost-to-go at any state (w,R,P) given an action S, because we need to compute $t(S)$ which requires the solution of a TSP over $S \cup \{0\}$. Given all these levels of difficulty, we will focus on finding good heuristics policies for the DDWP.

3.3 The Deterministic DDWP

Suppose the wave τ_i at which order $i \in N$ is ready is known at the beginning of the operating horizon. Then, the set of orders ready at any wave w for each $w \in \mathscr{W}$ is known beforehand, but it remains infeasible to serve an order $i \in N$ with a vehicle dispatch prior to τ_i. Let $N_w := \{i \in N : a_i \leq w \leq \tau_i\}$ be the set of orders ready and feasible to serve at wave $w \in \mathscr{W}$; where a_i defines the latest wave to feasibly serve order i, *i*.e., $a_i := \left\lceil 2t_{\{0,i\}}/\ell \right\rceil$. Problem 3.3.1 states the deterministic DDWP, which is NP-Hard since it generalizes the Prize-Collecting Traveling Salesman Problem (PC-TSP); just set all $\tau_i = 1$, $W = 1$, and $\ell = \sum_{i\in N} 2t_{0i}$.

Problem 3.3.1 (Deterministic DDWP)**.** *Find a collection of mutually disjoint subsets* $\{S_w \subset N_w : w \in \mathscr{W}\}$ *that minimize* $\sum_{w\in\mathscr{W}} \left\{ t(S_w) - \sum_{i\in S_w} \beta_i \right\}$ *subject to:*

1. $q_w(S_w) \geq 0$ *for each* $w \in \mathscr{W}$, *and*

2. *if* $S_w \neq \emptyset$, *then* $S_v = \emptyset$ *for all* $v \in \{w-1, \ldots, q_w(S)\}$, *for each* $w \in \mathscr{W}$.

Property 3.3.2 is taken from [44] and extended to a general network topology.

Property 3.3.2 (No wait after a dispatch)**.** *There exists an optimal solution to Problem 3.3.1 in which the vehicle does not wait after the first dispatch has occurred.*

In [44], we show for the one-dimensional case that an optimal solution contains consecutive vehicle dispatches with decreasing dispatch duration. This property does not hold for a general network topology. Consider a line segment with a centered depot and two orders located at each end of the line. Let $W = 6$, $\ell = 1$, $\alpha = 1$, $\beta_1 = \beta_2 = 7$, $\tau_1 = 4$, $\tau_2 = 6$, $t_{\{0,1\}} = 2$, $t_{\{0,2\}} = 1$, $t_{\{1,2\}} = 3$. Leaving any order unattended costs at least 7 and the unique solution serving both orders has cost 6. This solution dispatches a vehicle round-trip to serve order 2 at wave $w = 6$, and again at wave 4 to serve order 1.

We next formulate the deterministic DDWP as an Integer Program (IP). Define $E_w := \{e \in E, a_e \leq w \leq b_e\} \subset E$ for each $w \in \mathscr{W}$ as the set of feasible edges for a vehicle dispatch at wave w, where $a_{\{i,j\}} = \left\lceil \left(t_{\{0,i\}} + t_{\{i,j\}} + t_{\{0,j\}}\right)/\ell \right\rceil$ and $b_{\{i,j\}} = \min\{\tau_i, \tau_j\}$, for each $\{i,j\} \in E$. Also, define the cut set $E_w(S) = \{\{i,j\} \in E_w : i \in S, j \notin S\}$, for any subset $S \subseteq N_w$. Problem 3.3.1 is equivalent

to the IP

$$C^*(\tau) = \min_{\{\mathbf{x},\mathbf{y},\mathbf{v},\mathbf{z}\}} \sum_{i\in N:\tau_i>0} \beta_i\left(1 - \sum_{w=a_i}^{\tau_i} y_i^w\right) + \sum_{w\in\mathscr{W}} \sum_{e\in E_w} t_e x_e^w \tag{3.2a}$$

$$\text{s.t.} \sum_{w=a_i}^{\tau_i} y_i^w \le 1, \qquad \forall i \in N \tag{3.2b}$$

$$\sum_{e\in E_w(0)} x_e^w \le 2, \qquad \forall w \in \mathscr{W} \tag{3.2c}$$

$$\sum_{e\in E_w(S)} x_e^w \ge 2y_i^w, \qquad \forall w \in \mathscr{W}, \forall S \subseteq N_w, \forall i \in S \tag{3.2d}$$

$$\sum_{e\in E_w} t_e x_e^w \le \ell \sum_{k<w} (w-k) v_k^w, \qquad \forall w \in \mathscr{W} \tag{3.2e}$$

$$\sum_{k<W} z_k + \sum_{k<W} v_k^W = 1 \tag{3.2f}$$

$$\sum_{k<w} v_k^w = \sum_{k>w} v_w^k + z_w, \qquad \forall w \in \mathscr{W} \setminus \{W\} \tag{3.2g}$$

$$v_k^w \in \{0,1\}, \qquad \forall w \in \mathscr{W}, \forall k \in \mathscr{W} \cup \{0\} : k < w \tag{3.2h}$$

$$z_k \in \{0,1\}, \qquad \forall k \in \mathscr{W} \cup \{0\} : k < W \tag{3.2i}$$

$$y_i^w \in \{0,1\}, \forall i \in N_w, \text{ and } x_e^w \in \{0,1,2\}, \forall e \in E_w. \qquad \forall w \in \mathscr{W} \tag{3.2j}$$

Variable y_i^w is equal to 1 if a dispatch at wave w serves order i, and 0 otherwise; x_e^w is equal to $m \in \{0,1,2\}$ if the vehicle traverses edge e m times at a dispatch at wave w, and 0 otherwise; v_k^w is equal to 1 if a dispatch at w returns at wave k, and 0 otherwise; and z_k is equal to 1 if the vehicle waits at the depot until wave k, and 0 otherwise ($z_0 = 1$ implies no dispatch throughout the horizon).

Constraints (3.2b) guarantee serving each order i exactly once at wave w ($y_i^w = 1$) or, alternatively, leaving it unserved ($y_i^w = 0,\ w = a_i, \ldots, \tau_i$). Constraints (3.2c) - (3.2d) guarantee that vector $\mathbf{x}^w$ defines a feasible tour only visiting orders selected by the vector $\mathbf{y}^w$. Constraints (3.2e) force routes to satisfy durations limits determined by v_k^w. Finally, wave flow constraints (3.2f)-(3.2g) enforce vehicle conservation throughout time.

We can solve instances of (3.2) using a standard Branch & Cut approach with dynamic generation for subtour elimination cuts based on approaches for the TSP; see [8]. We start with singleton constraints (3.2d), *i*.e., $S = \{i\}$. If we find an integer $\mathbf{x}^w$ for wave w at any given node in the Branch & Bound tree, we check if it has a subtour in $\mathcal{O}(n)$ running time and add the corresponding cut. Moreover, if $\mathbf{x}^w$ is fractional we can check if it violates 2-connectivity by solving a Minimum Cut problem efficiently. If the answer is yes, we add the corresponding cut from (3.2d) and repeat.

Consider again the stochastic DDWP defined in Section 3.2. We can estimate a lower bound on the optimal expected cost of (3.1) with a Perfect Information Relaxation (PIR) of cost C_{PIR} that disregards the "non-anticipative" dynamics of the problem and computes one solution for each possible realization of the random variables [19, 61]. To estimate the PIR, we simulate a set of $m \in \{1, \ldots, M\}$ realizations τ^m for the random vector of ready times τ, find the deterministic optimal cost $C^*(\tau^m)$ for each realization m, and take the sample average $C_{PIR} \approx \sum_{m=1}^{M} C^*(\tau^m)/M$.

3.4 *A priori* policies

In this section, we develop a procedure to compute an *a priori* policy in which a schedule specifying the waves at which to dispatch the vehicle and the subsets of orders to be covered at each

dispatch is determined only with information disclosed at wave W.

We start by planning an optimal *a priori* policy in which no recourse actions are allowed. The operating cost of such a policy is known beforehand, and the penalties paid for unserved orders depend on future order arrivals. In this case each order has a probability $\mathbb{P}(\tau_i = w)$ to be ready at wave w. To simplify notation, the probability values include all information regarding the initial set of orders $\hat{R}$, *i*.e., $\mathbb{P}(\tau_i = W) = 1$ if $i \in \hat{R}$ and 0 otherwise. This *a priori* problem is equivalent to solving a deterministic DDWP instance in which each order $i \in N$ is duplicated at most W times and each copy is assumed to be ready at each wave $w \in \mathscr{W}$ in the support of the random variable τ_i, with an adjusted penalty for not serving the order equal to $\beta_i P(\tau_i = w)$.

Under this "extended" deterministic model, there exists an optimal solution visiting the customers within the same location at most once. If this is not the case, we could simply delete all but the latest visit and reduce the vehicle dispatch cost without a loss on customer coverage. This observation allows us to define an *a priori* IP problem where planning to serve order i at wave w indicates that order i is indeed serviced if it arrives during any wave $v \in \{w, w+1, \ldots, W\}$; this action thus reduces the expected penalty by $\beta_i \mathbb{P}(\tau_i \geq w)$. Define the expected penalty to be paid if no vehicle dispatches are planned as $\beta^0 := \sum_{i \in N} \beta_i \mathbb{P}(\tau_i \geq 1)$, the earliest wave that a vehicle can serve order i as $b_i := \max\{w : \mathbb{P}(\tau_i = w) > 0\}$, and redefine the sets N_w and E_w accordingly. Then, the *a priori* DDWP is defined by

$$C^*_{AP}(\hat{R}) = \min_{\{\mathbf{x},\mathbf{y},\mathbf{v},\mathbf{z}\}} \beta^0 - \sum_{i \in N} \sum_{w=a_i}^{b_i} \beta_i \mathbb{P}(\tau_i \geq w) y_i^w + \sum_{w \in \mathscr{W}} \sum_{e \in E_w} t_e x_e^w \tag{3.3a}$$

$$\text{s.t. } (3.2\text{c}) - (3.2\text{j}),$$

$$\sum_{w=a_i}^{b_i} y_i^w \leq 1, \qquad \forall i \in N, \tag{3.3b}$$

which shares its feasible region with (3.2), but now has a stochastic objective. In case of not serving an order, we pay a penalty discounted by the arrival probability $\mathbb{P}(\tau_i \geq 1)$, and in case of

planning to serve i at wave w, we pay the penalty discounted by the probability of a later arrival, $\mathbb{P}(1 \leq \tau_i < w)$.

We can improve the performance of the *a priori* policy at execution by skipping planned orders that aren't ready on time from a route dispatched at wave w; the triangle inequality guarantees that the solution cannot become more costly by skipping. We define by $\mathscr{Q}_w(\mathbf{x}^{\mathbf{w}})$ the expected duration of a dispatch at wave w given by $\mathbf{x}^w$; [40] provides a closed form to compute this expected cost for a given route in $\mathscr{O}(n^2)$ time. The expected cost of such a policy for a given *a priori* feasible solution is equal to

$$\beta^0 - \sum_{i \in N} \sum_{w=a_i}^{b_i} \beta_i \mathbb{P}(\tau_i \geq w) y_i^w + \sum_{w \in \mathscr{W}} \mathscr{Q}_w(\mathbf{x}^{\mathbf{w}}). \tag{3.4}$$

We could also design an *a priori* solution that considers the order-skipping recourse proactively when planning a solution with an extension of the L-shaped method described for the Probabilistic Traveling Salesman Problem (PTSP) in [45]. We implemented this approach for our problem and were only able to solve small instances to optimality ($n = 25, W = 3$). Moreover, the benefits in cost savings over the value of (3.3) were small (under 2%). The intuition is the following: If the arrival probabilities are high, the probabilistic routing cost in the objective collapses to a deterministic routing cost, just as the PTSP. However, this is a prize-collecting problem, which is fundamentally different from the PTSP, and if the arrival probabilities are small, the probabilistic routing cost reduces its comparative importance with respect to the penalty cost in the objective and it becomes more important to make good dispatch selections and less important to route. Nonetheless, it would be interesting for future research efforts to design efficient heuristic procedures over the a-priori cost with order-skipping recourse starting from an optimal solution of (3.3) to gain these marginal improvements over the *a priori* solution; see *e*.g., [18, 62].

3.4.1 Practical considerations when solving the *a priori* model

To solve larger instances of the *a priori* model effectively, we propose a solution improvement local search heuristic (LS) that exploits problem structure and complements a MIP solver. Specifically, we use this heuristic in two phases when solving problem (3.3). First, we run the heuristic for any new feasible solution s identified during the branch-and-bound tree search, and update the incumbent if the local search produces a solution with lower cost. Second, we use the heuristic during a solution construction phase as described in Algorithm 8 to generate a good initial feasible solution for the MIP solver.

Let $\{r_w^s, w \in \mathscr{W}_s\}$ be the set of routes of a feasible solution s to (3.3) indexed over the wave subset $\mathscr{W}_s \subseteq \mathscr{W}$ where these dispatches occur; we refer to $\mathscr{W}_s$ as the dispatch profile of s. Each route r_w^s represents an elementary sequence of order visits starting and ending at the depot. The *a priori* cost c_s is defined by (3.3a), its first dispatch wave by $ini_s := \max\{w \in \mathscr{W}_s\}$, the duration of each route by d_w^s, and its unserved customer set by $N_s := N \setminus \bigcup_{w \in \mathscr{W}_s} r_w^s$. An example of a feasible solution is given in Table 3.1.

Table 3.1: Example of a feasible solution s for an instance with 25 probabilistic orders

$\mathscr{W}_s$	route (r_w^s)	duration (d_w^s)
1	$\{0,22,16,17,2,9,18,21,0\}$	1
3	$\{0,15,3,12,6,8,7,14,20,4,24,25,19,5,10,23,0\}$	2
4	$\{0,1,0\}$	1
N_s	{11,13}	

Our LS procedure uses three separate neighborhood searches given solution s: (1) intra-route local search (IntraLS), *i*.e., single route node selection and re-sequencing; (2) inter-route local search (InterLS), *i*.e., node exchanges between routes and re-sequencing; and (3) dispatch profile local search. Pseudocode for LS is given by Algorithm 1. We execute the three-level search until no improving solution is found; each separate search procedure is described below.

Algorithm 1 Local Search Procedure

1: **procedure** LOCALSEARCH(Solution s)
2: **loop**
3: **if** ($\neg$INTRALS(s) **and** $\neg$INTERLS(s) **and** $\neg$DPLS(s)) **then return** s.

IntraLS, defined in Algorithm 2, exploits the relation between the DDWP and a prize-collecting TSP

$$PCTSP(m,Q,\beta) := \min_{\substack{S\subseteq Q:\\ t(S)\leq m\ell}} \left\{ \alpha t(S) - \sum_{i\in S} \beta_i \right\}, \tag{3.5}$$

with a set $Q \subseteq N$ of potential customer orders, prizes $\beta_i, i \in Q$, and a maximum route duration of m waves. IntraLS is a best move procedure, where a move is described by re-optimizing one route from s leaving all remaining routes unaltered. The procedure chooses a single route r_w^s from s, and solves a PCTSP over a set of nodes $Q := N_s \cup r_w^s$ defined by all orders left unattended if route r_w^s where removed, a maximum route duration $m = d_w^s$ predefined by the waves left available, and prizes β_i discounted by $\mathbb{P}(\tau_i \geq w)$, for $i \in Q$. Any solution s processed by IntraLS contains only routes w that are optimally sequenced and that cannot be improved by selecting a different subset of orders to service from $N_s \cup r_w^s$.

Algorithm 2 Intra-route LS procedure

1: **procedure** INTRALS(Solution s)
2: $improved \leftarrow false$ and $s^* \leftarrow s$
3: **repeat**
4: **for** $w \in W_s$ **do**
5: Let s' be a copy of s without route r_w^s
6: Solve PCTSP($d_w^s, N_{s'}, \{\beta_i \mathbb{P}(\tau_i \geq w)\}$) and add optimal route found to s' at wave w
7: **if** ($c_{s'} < c_{s^*}$) **then** $s^* \leftarrow s'$ **and** $improved \leftarrow true$
8: **if** ($c_{s^*} < c_s$) **then** $s \leftarrow s^*$
9: **until** $\neg improved$

InterLS uses best move searches over pairs of routes using neighborhoods inspired by those in [66] for the CVRP: two-edge exchanges between routes, removal and reinsertion of a k-order sequence from one route to another, and order swaps between routes. To implement these ideas, we

account for two differences between the CVRP and the DDWP. First, we model the prize-collecting component; a move changes penalty savings (due to the different dispatch time). Second, we check the durations of the new routes to ensure that they remain compatible with the fixed dispatch times of the unchanged routes.

The third neighborhood search that we implement is a Dispatch Profile Local Search (DPLS), described in Algorithm 3. The DPLS search perturbs the structure of the dispatch profile W_s for solution s using five operators: Cut, Merge, Exchange, Start, and Reorder. The first four operators find potential new solutions by solving a PCTSP, while the fifth uses a job scheduling approach.

Algorithm 3 Dispatch Profile Local Search (DPLS)

```
1: procedure DPLS(Solution s)
2:     improved ← true
3:     repeat
4:         if (¬CUT(s) and ¬MERGE(s) and ¬EXCHANGE(s) and ¬START(s) and
   ¬REORDER(s)) then
5:             improved ← false.
6:     until ¬improved
```

The Cut operator, described in Algorithm 4, searches over all possible dispatch profiles that result when splitting a single dispatch w with duration $d_w^s \geq 2$ into two dispatches with shorter duration; this operator always add an extra return trip to the depot, as depicted in Figure 3.1. The Merge operator, described in Algorithm 5, works in reverse and searches over all dispatch profiles that arise when merging two consecutive dispatches into a single longer duration dispatch, as shown in Figure 3.2. The Exchange operator, described in Algorithm 6, changes the dispatch durations of two consecutive dispatches (thus changing the dispatch wave w of the latter); see Figure 3.3. The Start operator, described in Algorithm 7, searches for a better solution among the (at most) two new dispatch profiles induced by moving the initial dispatch wave of solution s backward or forward one wave, when feasible without altering subsequent dispatches; when the move is backward, the initial dispatch is extended by one wave and when the move is forward, it

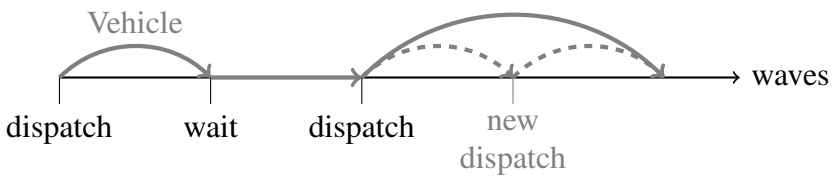


Figure 3.1: Example of a cut operation where a new dispatch profile is created (dashed flow) from an existing one (continuous flow) by cutting a route into two.

is reduced by one wave, as depicted in Figure 3.4.

The Reorder operator reassigns the routes in s to the best possible dispatch waves, without altering the customer visit sequences or the route durations. This assignment problem is equivalent to a single machine job scheduling problem of type $1||\sum_j f_j(w_j)$ (see *e*.g., [54]): Consider a set of jobs, where each job j corresponds to a route with processing time p_j equal to the route duration (measured in waves). The cost of assigning job j to start wave w is $f_j(w) := \sum_{i \in r^j} \mathbb{P}(\tau_i < w)$. This job scheduling problem is *NP*-hard, since the single-machine scheduling problem minimizing the weighted sum of tardy jobs can be reduced to this problem. Small instances with fewer than 10 routes can be solved effectively with dynamic programming.

Algorithm 4 Cut operation over solution s

procedure CUT(Solution s)
  $improved \leftarrow false$ and $s^* \leftarrow s$
  for $w \in W_s$ **do**
    for $v : (w-1) \rightarrow (w - d^s_w + 1)$ **do**
      Let s' a copy of s without route r^s_w
      Solve PCTSP($w - v, N_{s'}, \{\beta_i \mathbb{P}(\tau_i \geq w)\}$) and add optimal route to s' at wave w.
      Solve PCTSP($v - w + d^s_w, N_{s'}, \{\beta_i \mathbb{P}(\tau_i \geq v)\}$) and add optimal route to s' at wave v
      if ($c_{s'} < c_{s^*}$) **then** $s^* \leftarrow s'$ and $improved \leftarrow true$
  if ($c_s > c_{s^*}$) **then** $s \leftarrow s^*$
  return *improved*

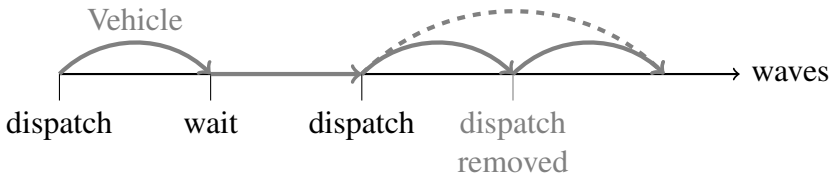


Figure 3.2: Example of a merge operation where a new dispatch profile is created (dashed flow) from an existing one (continuous flow) by merging two dispatches into one.

Algorithm 5 Merge operation over solution s

procedure MERGE(Solution s)
  $improved \leftarrow false$ and $s^* \leftarrow s$
  for $w \in W_s$ such that $w - d^s_w > 0$ **do**
    Let s' a copy of s without routes r^s_w and $r^s_{w-d^s_w}$
    Solve PCTSP($d^s_w + d^s_{w-d^s_w}, N_{s'}, \{\beta_i \mathbb{P}(\tau_i \geq w)\}$) and add optimal route to s' at wave w
    if $(c_{s'} < c_{s^*})$ **then** $s^* \leftarrow s'$ and $improved \leftarrow true$
  if $(c_s > c_{s^*})$ **then** $s \leftarrow s^*$
  return *improved*

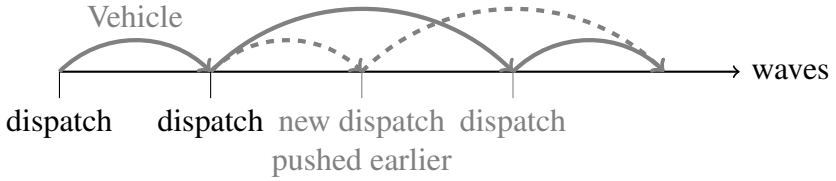


Figure 3.3: Example of an exchange operation where a dispatch gives one wave to its successor (dashed flow).

Algorithm 6 Exchange operation over solution s

procedure EXCHANGE(Solution s)
  $improved \leftarrow false$ and $s^* \leftarrow s$
  for pair $w_1, w_2 \in W_s$ such that $d^s_{w_1} > 1$ and $(w_1 = w_2 - d^s_{w_2}$ or $w_1 = w_2 + d^s_{w_1})$ **do**
    if $(w_1 > w_2)$ **then** Let $w'_1 = w_1$, $w'_2 = w_2 + 1$, $d'_1 = d^s_{w_1} - 1$, and $d'_2 = d^s_{w_2} + 1$.
    else Let $w'_1 = w_2$, $w'_2 = w_1 - 1$, $d'_1 = d^s_{w_2} + 1$, and $d'_2 = d^s_{w_1} - 1$
    Let s' a copy of solution s without routes $r^s_{w_1}$ and $r^s_{w_2}$
    Solve PCTSP($d'_1, N_{s'}, \{\beta_i \mathbb{P}(\tau_i \geq w'_1)\}$) and add optimal route to s' at wave w'_1
    Solve PCTSP($d'_2, N_{s'}, \{\beta_i \mathbb{P}(\tau_i \geq w'_2)\}$) and add optimal route to s' at wave w'_2
    if $(c_{s'} < c_{s^*})$ **then** $s^* \leftarrow s'$ and $improved \leftarrow true$
  if $(c_s > c_{s^*})$ **then** $s \leftarrow s^*$
  return *improved*

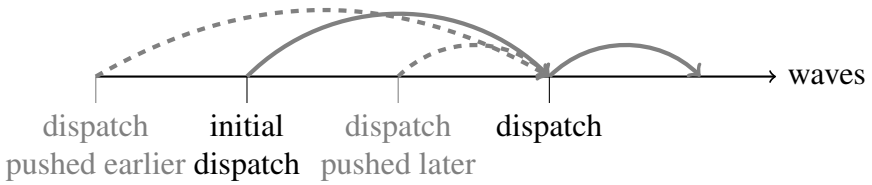


Figure 3.4: Example of a Start operation where the first dispatch is enlarged/reduced (dashed flow).

Algorithm 7 Start operation over solution s

procedure START(Solution s)
  $improved \leftarrow false$ and $s^* \leftarrow s$
  Let s_1 and s_2 be two copies of solution s without route $r^s_{ini_s}$
  if $(ini_s < W)$ **then**
    Solve PCTSP($d^s_{ini_s}+1, N_{s_1}, \{\beta_i \mathbb{P}(\tau_i \geq ini_s+1)\}$) and add optimal route to s_1 at wave ini_s+1
    if $(c_{s_1} < c_{s^*})$ **then** $s^* \leftarrow s_1$ and $improved \leftarrow true$
  if $(d^s_{ini_s} > 1)$ **then**
    Solve PCTSP($d^s_{ini_s}-1, N_{s_2}, \{\beta_i \mathbb{P}(\tau_i \geq ini_s-1)\}$) and add optimal route to s_2 at wave ini_s-1
    if $(c_{s_2} < c_{s^*})$ **then** $s^* \leftarrow s_2$ and $improved \leftarrow true$
  if $(c_s > c_{s^*})$ **then** $s \leftarrow s^*$
  return $improved$

We can in addition enhance the search by calling LS recursively, *i*.e., we run LS within the Cut, Merge, Exchange, and Start operators on the temporary solution s', after the move gets implemented but before comparing to the best solution available s^*. In our experiments, we implemented a two-level local search to keep solution times manageable.

In addition to using the LS procedure during the branch-and-bound tree search, we also embed it in a heuristic for building a good initial feasible solution s_0 for the DDWP. This constructive heuristic is given a set of m dispatch profiles, and for each $W_k, k = 1, \ldots, m$ solves a series of sequential prize-collecting TSPs over time to build a solution which is then improved by the LS procedure. In our experiments, the set of dispatch profiles provided to the construction heuristic was $\{\{1,2,\ldots,q\}\}$ for all q in $1 \leq q \leq W$, *i*.e., all possible profiles that include consecutive, single-

wave dispatches up to the final wave.

Algorithm 8 Constructive Heuristic

procedure CONSTRUCTIVE(Set of dispatch profiles $\{\mathscr{W}_1, \mathscr{W}_2, \ldots, \mathscr{W}_{s_m}\}$)
  Initialize s^0 as the empty solution
  for $k : 1 \to m$ **do**
    Build $\{d_w, w \in \mathscr{W}_k\}$, the dispatch durations of $\mathscr{W}_k$
    Let s_k be a copy of s_0.
    for $w \in \mathscr{W}_k$ in decreasing order **do**
      Update N_{s_k}
      Solve a PC-TSP over N_{s_k}, with max duration d_w and prizes $\beta_i \mathbb{P}(\tau_i \geq w)$
      Add optimal route to s_k.
    LOCALSEARCH(s_k)
    if $(c_{s_k} < c_{s_0})$ **then** $s_0 \leftarrow s_k$
  return s_0

3.5 Dynamic Policies

A priori policies, particularly when adjusted via recourse actions, may yield reasonable solutions to many problems. However, [44] provides an instance sequence for which an optimal *a priori* policy with recourse is arbitrarily worse than an optimal dynamic policy, *i*.e., $C^*_{AP}(\hat{R})/C^*(\hat{R}) \to \infty$. Therefore, we next develop dynamic policies.

3.5.1 *A Priori*-Based Rollout Policy

A natural, but possibly computationally-expensive dynamic policy is to roll out the *a priori* policy. At each wave $w \in \mathscr{W}$ when the vehicle is available at the depot, we recompute an optimal *a priori* policy beginning at wave w using the current system state (w, R, P) to define a new, reduced problem over the remaining operating period. If the policy decides at wave w to dispatch a route serving customer set $S \subset R$, then this decision is implemented and a new *a priori* policy is computed again at wave $q_w(S)$; otherwise, a new *a priori* policy is computed at $w-1$ after waiting for one wave.

Computing such a rollout policy requires the solution of $\mathcal{O}(W)$ *a priori* problems and may be difficult to solve exactly for larger instances. However, there are multiple ways to improve the computational performance. First, one can warmstart the solution of the *a priori* problem at wave w using the most recently computed solution (from wave $q > w$), adjusted to skip all planned orders that are not ready yet and then improved via the LS procedure. It is also not necessary within this heuristic procedure to solve each *a priori* problem to optimality. A more substantial simplification is to begin the rollout process only at the first wave where a dispatch is planned in the initial *a priori* solution (computed at wave W); we call this approach the *restricted* rollout policy.

Improvement guarantees for the a priori-based rollout policy

Now we show that rolling out the *a priori* solution has no bigger expected cost than the *a priori* policy without recourse actions. We base this proof in the result provided by [38] stating that any *sequentially improving* heuristic weakly improves its performance when rolled out. The definition of sequentially improving heuristic is stated in 3.5.1. In Proposition 3.5.2, we prove that the optimal *a priori* solution is a sequentially improving heuristic.

Definition 3.5.1. *(Sequentially Improving Policy). Let s be the state of the system, let $\pi^{H(s)}$ be a policy induced by a Heuristic $H(s)$ computed at state s, and let $C^{\pi}(s)$ be the expected cost-to-go paid if a policy π is implemented from state s onwards. Let s' be any state such that it is on a sample path induced by implementing $\pi^{H(s)}$ at state s. Then, $H(\cdot)$ is sequential improving if*

$$\mathbb{E}\left[C^{\pi^{H(s)}}(s')|s'\right] \geq \mathbb{E}\left[C^{\pi^{H(s')}}(s')|s'\right] \tag{3.6}$$

Proposition 3.5.2. *The optimal a priori solution is sequentially improving*

Proof. (Proposition 3.5.2) Let $\pi^{AP(s)}$ be the resulting policy that arises from the optimal *a priori* policy computed in state s and let s' a state of the system in the sample path induced by $\pi^{AP(s)}$. As-

sume by contradiction that we have $\mathbb{E}\left[C^{\pi^{AP(s)}}(s')|s'\right] < \mathbb{E}\left[C^{\pi^{AP(s')}}(s')|s'\right]$, then $\pi^{AP(s)}$ has expected cost lower than the expected cost of the optimal *a priori* solution policy computed at state s'. This is clearly a contradiction. □

3.5.2 Greedy *a priori*-based prize-collecting TSP Heuristic

We also test simpler rollout strategies that do not rely on repeatedly solving the *a priori* problem. One such approach takes advantage of the relationship between the DDWP and the PC-TSP. To initiate the approach, we solve the *a priori* problem at initial wave W. We then use this solution to guide a rollout procedure that only solves deterministic PC-TSP problems as follows. At each wave $w < W$ for which the *a priori* solution dictates a vehicle dispatch, we determine the maximum duration of the route (by ensuring the vehicle is back at the depot for the next *a priori* dispatch wave). Then, we determine a vehicle route by solving a PC-TSP using only open orders that are not postponed in the *a priori* solution for dispatch at a later wave. Let $\mathscr{W}^{AP}$ be the set of waves where vehicle dispatches take place in the *a priori* solution, and let $Q_w \subset N_w$ be its set of planned orders to be dispatched at each $w \in \mathscr{W}^{AP}$. We implement the heuristic dynamic policy outlined in Algorithm 9.

Algorithm 9 Greedy *a priori*-based policy

1: Set $w \leftarrow \max\{v \in \mathscr{W}^{AP}\}$, $w^+ \leftarrow \max\{v \in \mathscr{W}^{AP} : v < w\}$
2: Wait at the depot until wave w
3: **while** $w > 0$ **do**
4: Read system state (w, R, P)
5: Compute $\bar{R} := R \setminus \bigcup_{v=1}^{w^+} Q_v$, the set of open orders not included in future dispatches of the *a priori* solution
6: Solve PCTSP$(w - w^+, \bar{R}, \beta)$ and let $S \subset \bar{R}$ be the optimal set of sequenced orders selected
7: **if** $q_w(S) = w^+$, **then** dispatch a vehicle to S, set $w \leftarrow w^+$, $w^+ \leftarrow \max\{v \in \mathscr{W}^{AP} : v < w\}$,
8: **else** set $w \leftarrow w - 1$.

The greedy policy ignores information about orders dynamically realized throughout the operation, and only considers the available probabilistic information at the start of the horizon. However,

it is dynamic enough to accommodate newly arriving orders, and to re-optimize routing decisions. This last feature makes it a better candidate than a simple *a priori* policy with order-skipping recourse.

3.6 Computational Experiments

We now present a set of computational experiments designed over a family of randomly generated instances with the objective of testing the quality of our heuristic policies and to get qualitative insights regarding the management of vehicle dispatches in a same-day delivery context. Table 3.2 summarizes the heuristic policies computed for each instance. All heuristics were programmed in Java and computed using one thread of a Xeon E5620 processor with up to 12Gb RAM, using CPLEX 12.6 when necessary as a MIP solver.

Table 3.2: Heuristic policies computed in our experiments

symbol	strategy	section
AP	*a priori* policy + order-skipping	3.4
GP	Greedy PCTSP-based policy	3.5.2
RP	Rollout of *a priori* policy	3.5.1
RRP	Restricted rollout of *a priori* policy	3.5.1

3.6.1 Design of data sets

We generated 240 data sets to evaluate our policies over different performance indicators. Each data set has a specific geography of 50 orders, a known subset $\hat{R} \subseteq \{1,\ldots,50\}$ of orders ready at the start of the operating period, and a vector of ready wave probabilities for orders with unknown arrival wave.

The geography for each data set is defined by a random seed $g \in \{0,\ldots,4\}$ used to assign 50 different locations over a 51×51 square subset of $\mathbb{R}^2$ following a discrete uniform distribution $U(0,50)$ for each component of the location's coordinate and with the depot located at coordinate

$(25,25)$. We ruled out repeated coordinates to have a more interesting geography. Travel times between locations are given by the ℓ_1-norm (Manhattan distance) between two locations, chosen to model urban travel times. All data sets share a common horizon with $W = 6$ possible dispatch waves, each with duration $\ell = 100$ time units. The duration of a round-trip visiting any single order is less than or equal to 100 time units, and thus can be completed in a single dispatch wave.

For each order $i \in \{1,..,50\}$, its ready wave τ_i is a discrete random variable, independently distributed with probability $\mathbb{P}(\tau_i = W) = p_{start}$ of being ready at the start of the operating period (the order's degree of dynamism); a conditional probability $\mathbb{P}(\tau_i = -1|\tau_i < W) = p_{out}$ of not arriving at all during the operation period; and a conditional discrete uniform distribution with probability $\mathbb{P}(\tau_i = w|1 \leq \tau_i < W) = (\min(W-1, \mu_i+\sigma) - \max(1, \mu_i-\sigma)+1)^{-1}$ of arriving during the operation at any wave $w \in \{\max(1, \mu_i-\sigma), \ldots, \min(W-1, \mu_i+\sigma)\}$. The parameter μ_i represents the mean ready wave and is drawn for each i with equal probability between 1 and $W-1$, and σ represents variability. Each data set uses a triple $(\sigma, p_{start}, p_{out}) : \sigma \in \{\text{Lo} = 1, \text{Hi} = 6\}$, $p_{start} \in \{10\%, 15\%, 25\%, 50\%\}$, $p_{out} \in \{20\%, 40\%\}$, and a setting of the seed $h \in \{0,1,2\}$ to draw the set $\hat{R}$ of orders ready at start.

We created $M = 50$ realizations of the ready time vector τ for each data set using the probabilistic model above. These scenarios are used as a common sample to estimate lower bounds based on a perfect information relaxation and the expected cost of all policies. Each data set has a unique value of $(g, \sigma, p_{start}, p_{out}, h)$, and all sets, including their simulated realizations, are published online at

`sites.google.com/site/maklappor/ddwp-data-sets`.

3.6.2 Set 1: Base experiments

For our first set of experiments, we build 3 instances by considering the first 25, 35 and 50 orders for each one of the 240 data sets. This makes a set of 720 instances with 3 different problem sizes. The penalty for leaving order i unattended if it appears is set as the duration of a round-trip to

order i from the depot, $\beta_i := 2t_{\{0,i\}}$; this setting models a system that serves all realized orders by outsourcing the service of each order that remains open at the end of the day to a direct delivery service. Under these penalties, there is always a potential service profitability; in particular, there is an economic incentive to dispatch the vehicle in the last wave if any order is open.

We computed the following metrics for each policy over each realization from each instance:

- Total cost (*cost*), travel time ($duration$) and penalties paid ($penalty$),
- gap: the percentage increase of the policy's cost over the perfect information bound,
- fill rate (fr): the percentage of orders served by the vehicle over all realized orders,
- $duration/order$: the routes' duration over the number of served orders,
- *nRoutes*: number of vehicle dispatches,
- *nWaves*: average dispatch length in waves used by each route,
- *iWait* and *pWait*: number of waves spent waiting at the depot before/after the initial dispatch,
- $time_{off}$: average "off-line" solution time, *i*.e., before the solution is implemented,
- $totaltime_{on}$: total "on-line" solution time over the operating period,
- $time_{on}$: total "on-line" solution time divided by the number of active decision epochs, *i*.e., the number of waves in which the policy makes a dispatch decision (which excludes all predetermined waits established by the initial *a priori* policy and dispatch waves jumped by the vehicle routes).

These metrics are averaged for each instance over all $M = 50$ realizations. Table 3.3 presents average results for each heuristic policy over all instances.

Table 3.3: Average results of heuristic policies

metric policy	**AP**	**GrAP**	**RRP**	**RP**
cost (decrease from AP %)	555	523 (5.8%)	505 (9.1%)	505 (9.1%)
duration	298	305 (-2.5%)	305 (-2.4%)	311 (-4.4%)
penalty	258	218 (15.2%)	200 (22.3%)	194 (24.7%)
gap	23.1%	16.1%	12.1%	12.1%
fr	81.6%	85.0%	86.2%	86.6%
$duration/order$	11.0	11.2	11.2	11.4
nRoutes	2.5	2.5	2.6	2.7
nWaves (std)	1.4	1.4	1.4	1.4
iWait	2.6	2.6	2.6	2.5
pWait	0.003	0.005	0.002	0.003
$time_{off}$	836.2s.			
$totaltime_{on}$	0.0001s.	0.08s.	252.0s.	607.5s.
$time_{on}$	0.0002s.	0.18s.	85.8s.	120.0s.

On average, the AP policy's cost is 23.1% over the perfect information bound and, as expected based on each heuristic's recourse possibility, rolling it out improves upon AP and cuts the average gap by 47.7%, with an average cost reduction of 9.1%. The dynamic heuristic GP achieves 63% of that cost reduction, suggesting that a simple but dynamic policy can capture a significant amount of the benefits of dynamism. The RRP policy achieves similar average cost reduction and gap as RP, indicating that the AP policy is adequately choosing an initial dispatch. We find this to be a useful insight for managers: Assuming average behavior in the future appears to be sufficient when making an initial dispatch decision. Conversely, once the AP policy recommends an initial dispatch, RP and RRP perform better by incorporating newly arrived information.

Also, RP increases the order fill rate over AP by 6.2%, by redesigning the solution at each decision moment, which is crucial for logistics service providers interested in providing a better customer service; GP and RRP respectively achieve 68.1% and 90.6% of this fill rate increase. The dynamic policies' benefits mostly stem from reducing penalty costs, while on average they produce a slight increase in the routes' duration. This is a tradeoff wherein dynamic policies significantly improve order service, while incurring small increases in duration per order. We also see that

dynamic policies slightly increase the average number of dispatches and reduce the initial waiting time. In terms of waiting at the depot after the first dispatch (*pWave*), our results suggest it does not occur often, and it may be better to keep the vehicle busy serving more orders.

All policies share the $time_{off}$ value, but radically differ in on-line solution time: AP and GP are almost instantaneous online policies, while each rollout policy requires more computational power. RRP's total online time is significantly less than RP, even though the former closely approximates the latter in terms of cost and the other metrics.

In Figure 3.5, we observe the distribution of the gap over all instances for each heuristic policy. We observe that RP not only outperforms AP on average, but is also less variable in solution quality, with 3.7% deviation versus 7.4%; RRP has a similar distribution to RP's, while GP has an intermediate 4.9% deviation.

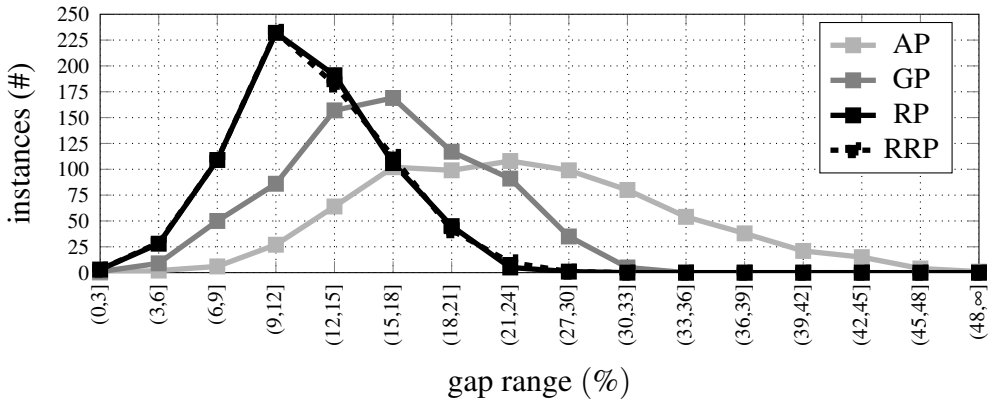


Figure 3.5: Distribution of gap over all instances

In Figure 3.6 we compare the average gap of our heuristic policies between instances sharing parameters of size n, degree of dynamism p_{start}, probability of not showing up p_{out}, and arrival variability σ. In the first graph on the left we see how the average gap increases with the number of potential orders n; this increase may be related to an increase in the problem's size and complexity, but also to the lower bound's tightness. Moreover we see that RPs gap difference over AP increases

as n grows, with GP in between and RRP indistinguishable from RP. We also observe a significant gap reduction for the *a priori* policy (AP) as p_{start} increases. As expected, the more information available at the initial wave, the closer we can get to a deterministic problem and the better we can optimize exactly. For dynamic policies the gap is significantly smaller and tends to be stable over different values of p_{start}, showing the benefit and importance of complex recourse actions when dealing with higher degrees of dynamism. Regarding the orders' conditional probability of not showing up and the variability of the ready wave, Figure 3.6 suggests that it may be harder to optimize instances with higher value of p_{out} and σ due to an increase in the problem's uncertainty and/or possibly due to a deterioration of our lower bound. Again, we see that the average gap reduction of dynamic policies over the *a priori* policy is particularly valuable for instances with higher ready time variability.

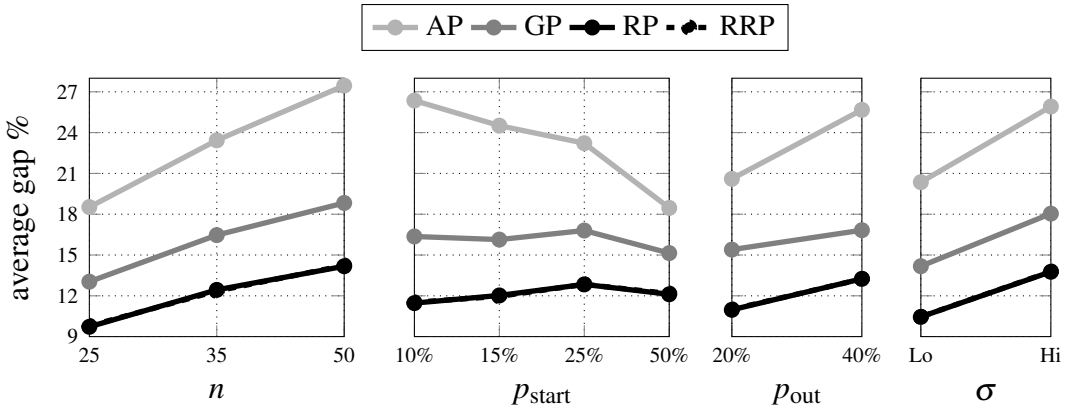


Figure 3.6: Average gap versus n, p_{start}, p_{out}, and σ

Figure 3.7 presents the average fill rate as a function of the instance parameters; for reference, maximizing fill rate is the objective function of the model presented in [71]. We see that fr decreases as n increases over all policies, which may indicate congestion related to available vehicle time. Interestingly, this fill rate reduction is marginally decreasing with n, presumably through order consolidation; *i*.e., at $N = 25$, an increase of 10 customers results in a larger fill rate decrease

than an increase of 15 customers at $N = 35$. Also, RP's fill rate is consistently over AP by more than 4.6% regardless of n. The second graph from the left shows how the average fill rate increases by a 16.1% amount for AP and 10.6% for RP when p_{start} increases from 10% to 50%, meaning that more information available at the start can significantly increase the number of orders served, especially for the AP policy. The relative difference between AP and RP is higher for instances with higher dynamism showing again the additional value of dynamic policies. The third graph from the left shows how RP (and dynamic policies more generally) can be useful when opportunities to increase fill rate are presented. While the *a priori* policy's rate remains stable over the probability of not showing up (p_{out}), all dynamic policies increase order fill rate, *e*.g., RP's increases by 1.6%. This may be explained by the fact that RP uses the time gained when initially planned orders do not realize to serve unexpected and initially unplanned orders that do show up. This effect is also observed when the average fill rate is compared versus σ. The fill rate of AP decreases with σ, but RP's fill rate is more stable and the difference between RP and AP increases as the variability parameter increases.

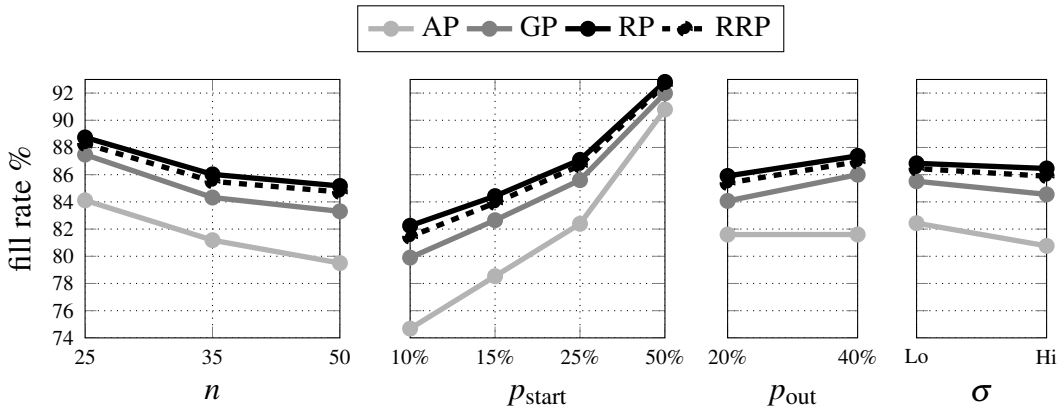


Figure 3.7: Average fill rate versus n, p_{start}, p_{out}, and σ

The graphs presented in Figures 3.8,3.9, and 3.10 show for each policy how the average number of dispatches, duration in waves per dispatch, and initial number of waves spent waiting at the

depot evolve over the different instance parameters. From the first graphs on the left, we see that all policies dispatch more vehicles as n grows and wait a smaller amount of initial waves at the depot, meaning that a relatively dense geography justifies a higher number of dispatches during the day and more "active" dispatch waves during the operating period. We also see that the difference in routes dispatched between RP and AP increases with n, and that dynamic policies slightly reduce the waves used per dispatch as n grows. This shows empirically how dynamic policies increase recourse opportunities, *i*.e., more returns to the depot, as n increases. The second set of graphs from left to right show how our policies have fewer and longer vehicle dispatches with shorter initial waiting periods as off-line information increases (higher p_{start}). This means that as deterministic information increases, fewer recourse opportunities are needed and routing efficiency becomes the focus. The shorter initial wait periods can be explained because there may be relatively less need to "wait and see" versus instances where less information is given at start. We also see how the RP policy comparatively increases the number of dispatches and reduces route duration (gaining recourse opportunities) as p_{start} gets smaller. Similar effects can also be observed from the graphs on the right. It is interesting to note that RP slightly increases the average number of dispatches and the route length over AP as p_{out} and *sigma* increase. Again, it shows how this policy can recover from uncertainty, *e*.g., orders realizing later than expected or not showing up, by dynamically inserting unplanned orders as substitutes.

The average solution times over all instances disaggregated by number of customers n and the probability p_{start} are presented Table 3.4. The results on the left table show the off-line solution times shared by all policies. As expected due to the nature of exact MIP models, this time increases exponentially with the number of orders, but should not be problematic since this procedure is intended to run before the start of the operating horizon. In addition, average times increase with p_{start}, probably because the initial model has a bigger feasible region. The right-hand table shows

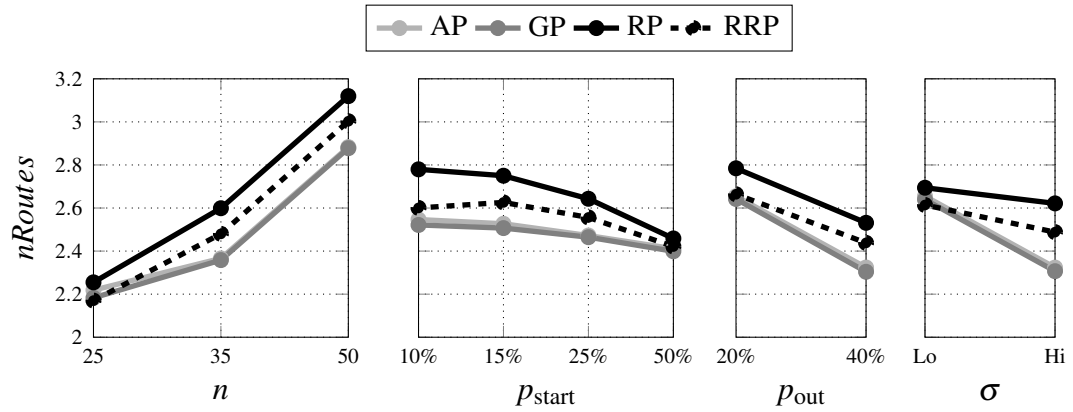


Figure 3.8: Average number of vehicles dispatched versus n, p_{start}, p_{out}, and σ

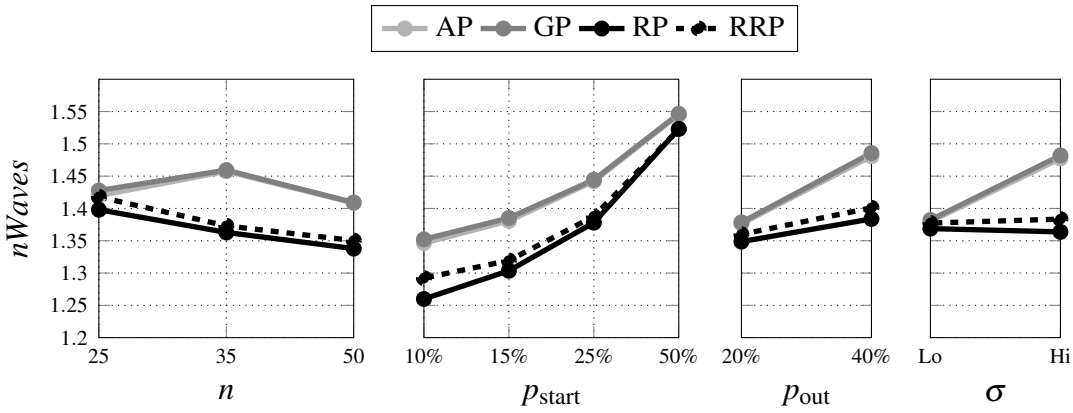


Figure 3.9: Average number of waves per vehicles dispatch versus n, p_{start}, p_{out}, and σ

the average solution times per dispatch of GP, RRP and RP policies. We observer a similar increasing effect over the number of customers, which depending on available computing resources could be an issue for the full rollout policy if n is large. Our results suggest substituting RRP for RP, since both policies showed almost equivalent cost reductions over AP. For even larger n, the GP policy is a simpler alternative to consider if RRP becomes inefficient; GP still takes less than 1 second per decision and generates better solutions than AP. Another possibility would be to further exploit local search and meta-heuristic procedures; we leave this question for future work.

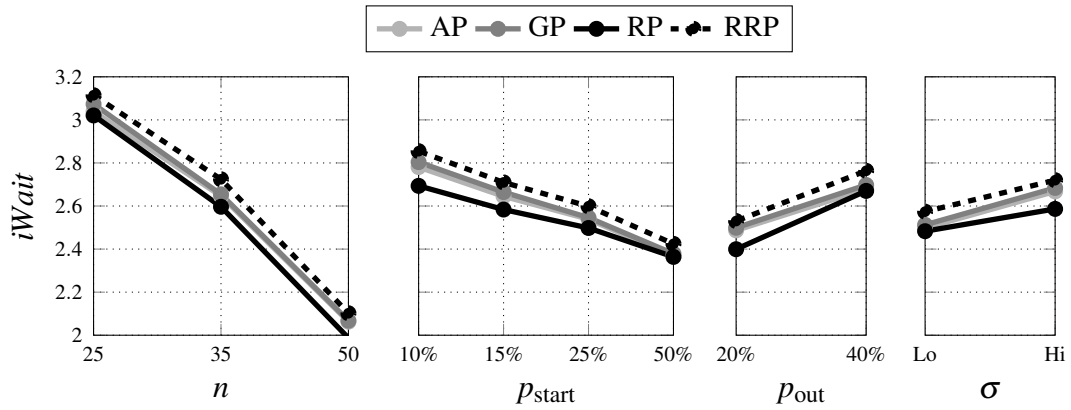


Figure 3.10: Average number waves waited before initial dispatch versus n, p_{start}, p_{out}, and σ

Table 3.4: Average $time_{on}$ and $time_{off}$ versus p_{start} and n

Average time offline in secs.

$p_{start} \backslash n$	25	35	50
10%	121	338	1689
15%	130	375	1870
25%	224	494	1886
50%	245	579	2084

Average time per decision in secs.

	GP			**RRP**			**RP**		
$p_{start} \backslash n$	25	35	50	25	35	50	25	35	50
0.10	0.06	0.10	0.16	3.4	16.5	147	5.5	44.0	273
0.15	0.05	0.12	0.14	4.1	16.1	186	6.0	41.4	284
0.25	0.04	0.05	0.07	4.3	32.8	256	7.1	57.8	324
0.50	0.03	0.03	0.06	4.7	51.9	305	7.8	62.8	326

3.6.3 Set 2: Second set of experiments

We now present a second set of computational experiments to study the tradeoff between two plausible objective functions in same-day delivery: minimizing total cost and maximizing (weighted) order coverage; in the DDWP setting this is equivalent to minimizing penalty costs. This coverage goal also matches the objective in the models in [71]. We look for basic tradeoffs, performance of our heuristic policies, and structural differences in our solutions to provide qualitative insights.

We build a new set of 720 instances by making 3 instances for each one of the 240 data sets (with 35 orders each). Each instance has a different value of $\alpha \in \{1, 2, 100\}$ for the penalty setting $\beta_i = 2\alpha d_{0i}$. While the first set ($\alpha = 1$) balances both vehicle traveling costs and penalty costs, the last one ($\alpha = 100$) hierarchically focuses on covering orders first, travel time minimization as a

Table 3.5: Average cost and reduction percentage over AP for each policy under different settings of α

policy\objective	$\alpha = 1$	$\alpha = 2$	$\alpha = 100$
LB	441	546	9395
AP	541	760	18958
GP	511 (5.5%)	711(6.4%)	16402 (13.5%)
RRP	493 (8.8%)	670 (11.8%)	13625 (28.1%)
RP	494 (8.7%)	669 (11.9%)	13614 (28.2%)

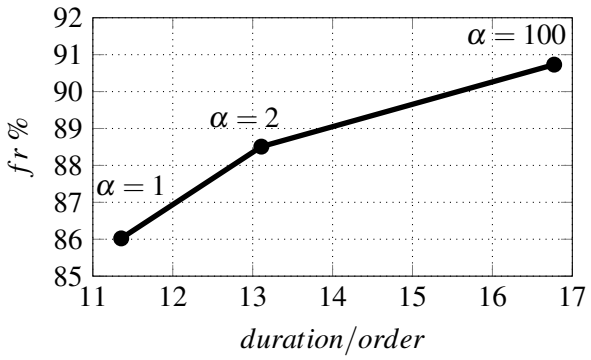
secondary objective; the second set ($\alpha = 2$) is an intermediate case.

Table 3.5 presents average costs of all heuristic policies and perfect information bound over all instances under different settings of α. It also shows for each dynamic policy the average cost reduction percentage over AP. We first observe that cost reduction percentages of dynamic policies substantially increase with α; this is explained by the order coverage improvement that dynamic policies enjoy, discussed in the first set of experiments. It may also occur because the cost savings from delivering an additional order become more significant as the weight on penalty costs is bigger. For example, the cost reduction from serving one more order when there are two left unattended is 50%. Second, we observe that RP becomes more attractive over GP as the relative importance of penalty costs increases, suggesting that sophisticated dynamic policies that get better fill rate improvements become more valuable as the focus shifts towards coverage.

The left table in Figure 3.11 presents results for RP averaged over all instances sharing the same settings of α. The first two rows present the tradeoff between coverage cost ($penalty$) and vehicle traveled time ($duration$) over the three cases of α. The third and fourth rows present these results in comparable metrics: $duration/order$ representing routing efficiency and fr representing order satisfaction. These results are plotted in the graph on the right as a Pareto chart. As expected, we observe that fr increases as order coverage becomes more relevant in the objective, but this fill rate improvement requires a sacrifice in $distance/order$, *i*.e., the higher the order fill rates, the more inefficient the routes. Moreover, the marginal rate of substitution between fr and $distance/order$

metric	$\alpha = 1$	$\alpha = 2$	$\alpha = 100$
$duration$	305	352	448
$penalty/\alpha$	189	159	132
$duration/order$	11.4	13.1	16.8
fr	86.0%	88.5%	90.7%
$nRoutes$	2.60	3.40	4.80
$nWaves$	1.36	1.22	1.13
$iWait$	2.60	1.97	0.58
$pWait$	0.003	0.013	0.047

(a) Metrics for RP under different settings of α



(b) Pareto chart for RP with fr versus $duration/order$

Figure 3.11: Average results for RP under different settings of α

decreases with α. From $\alpha = 1$ to $\alpha = 2$ the average gain in fr per $distance/order$ sacrificed is 1.4%. The same number from $\alpha = 2$ to $\alpha = 100$ is 0.61%, a 57% reduction. For decision makers, this suggests that even in the efficient frontier, the distance cost of an additional customer covered becomes increasingly more expensive; a cost-focused manager may be willing to sacrifice coverage for routing efficiency at a sufficiently high fill rate.

We further explain the decreasing marginal substitution rate between coverage and routing efficiency by looking at the last four rows of the table in Figure 3.11. RP has to create more recourse opportunities (more returns to the depot), increases by 85% the average number of dispatches and reduces the average dispatch length in waves by 17% to improve coverage; by the triangle inequality, this reduces routing efficiency. Moreover, it drastically reduces the initial wait time by 78%, implying a longer usage of the vehicle throughout the day. This is also an interesting insight for managers: When coverage is the objective, vehicles operate longer periods of time with higher maintenance costs and longer workdays for drivers (a higher cost in human resources). Conversely, order consolidation increases when total cost is the objective, routes become longer and efficient, and dispatches are pushed forward in time. This reduces the operation's length, but increases the time between an order's arrival and its dispatch; logistics providers may want to minimize this

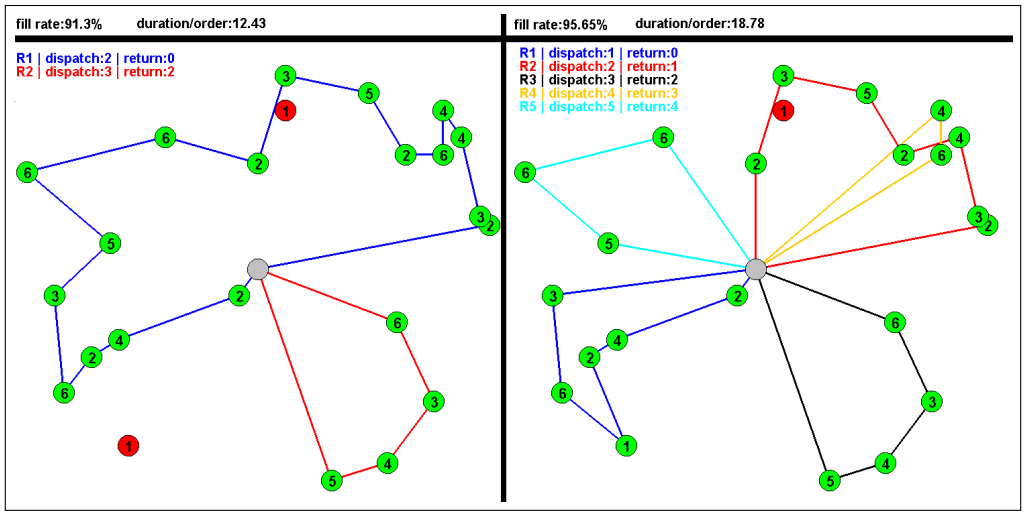


Figure 3.12: Example of two different solutions to the same instance realization. The left one minimizes total cost and the right one minimizes lost penalties first. Green orders are served and red ones are left unattended. Each order's ready wave is labeled at its node and route dispatch/return waves are shown in the upper-left corner.

value to guarantee good service. In Figure 3.12 we show this tradeoff graphically, presenting two solutions for the same instance realization. The left solution minimizes total cost while the right one minimizes lost penalties. We clearly see how the right solution has to sacrifice roughly 50% in routing efficiency to increase its order coverage by one order. It also makes 3 more vehicle dispatches and actively uses the vehicle for 2 more waves.

We next analyze the performance of all heuristics under different values of α and the data set parameters p_{start}, p_{out} and σ. In Figure 3.13 we present the previously shown Pareto charts for all heuristic policies under different settings of p_{start}. We observe that RP can approximately cut the gap between the *a priori* heuristic and the perfect information bound in half. Approximately two thirds of that improvement can be achieved by GP, while RPP appears equivalent to RP; this is consistent with the first round of experiments. When p_{start} is small (10%) the value of dynamic policies is higher and the marginal rate of substitution between fr and $duration/order$ is smaller. When p_{start} in higher (50%) there is less additional value in implementing dynamic policies and

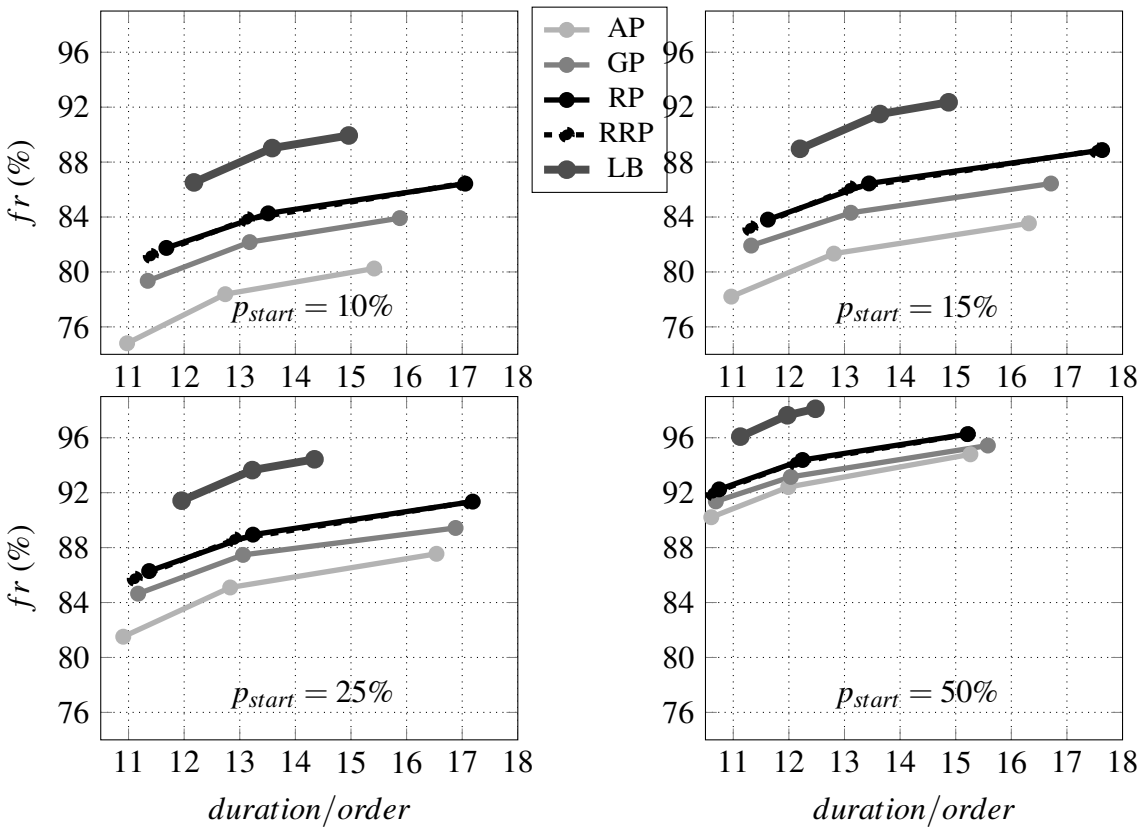


Figure 3.13: Pareto charts showing each policy's average fr and $duration/order$ for different settings of p_{start}

the marginal rate of substitution becomes higher, implying that one has to sacrifice less routing efficiency for a marginal increase in order coverage. We also see that as p_{start} grows the system moves to the upper left, better for both fr and $duration/order$.

In Figure 3.14 we present the Pareto charts of all heuristics for each value of the conditional probability of a order not showing up, p_{out}. We observe that as p_{out} increases all heuristics move to the right, a loss in routing efficiency explained by the increased amount of uncertainty in the order arrival process that makes *a priori* plans less reliable; this is especially true for problems maximizing fill rate, making the marginal rate of substitution smaller. Dynamic policies provide more value as p_{out} increases and the curves move up in relation to AP, because they can better

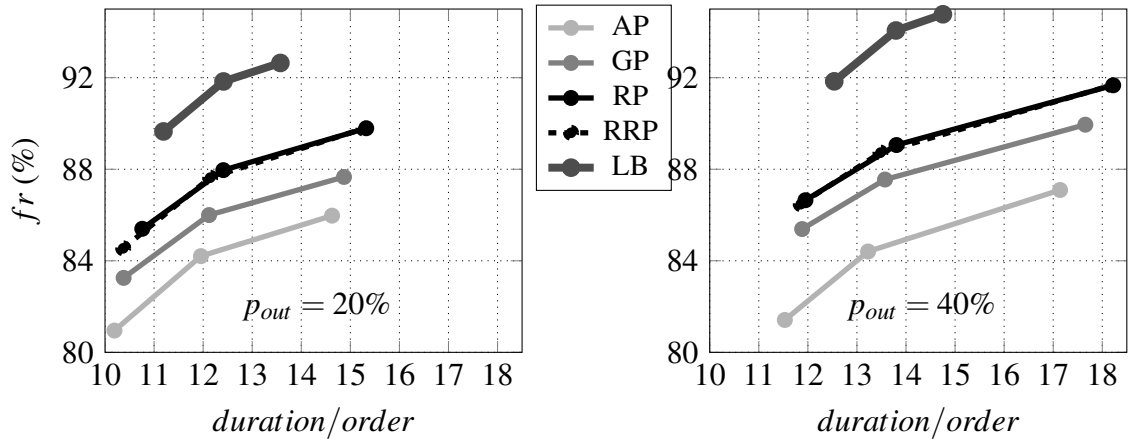


Figure 3.14: Pareto charts showing each policy's average fr and $duration/order$ for different settings of p_{out}

recover from unexpected order late arrivals and corresponding changes in the route plans.

In Figure 3.15 we analyze the Pareto charts of all heuristics under the two settings of order ready wave variability (σ). We observe two effects as variability increases. First, all curves get wider, with a smaller marginal rate of substitution. This implies that as arrival variability increases it is more expensive to gain coverage and more sacrifices in efficiency have to be made. Also, we observe that AP and GP move down as σ increases, meaning that our static policy and simple dynamic policy lose order coverage. This is not the case of RP and RRP; again we see the importance of sophisticated dynamic policies as variability increases.

3.7 Conclusions

We have formulated the Dynamic Dispatch Waves Problem (DDWP) on general network topologies to investigate the fundamental tradeoffs in same-day delivery distribution systems. We have formulated an integer program to solve the deterministic version of the problem, and used this model to derive an optimal solution for the stochastic *a priori* problem by converting it into a deterministic equivalent.

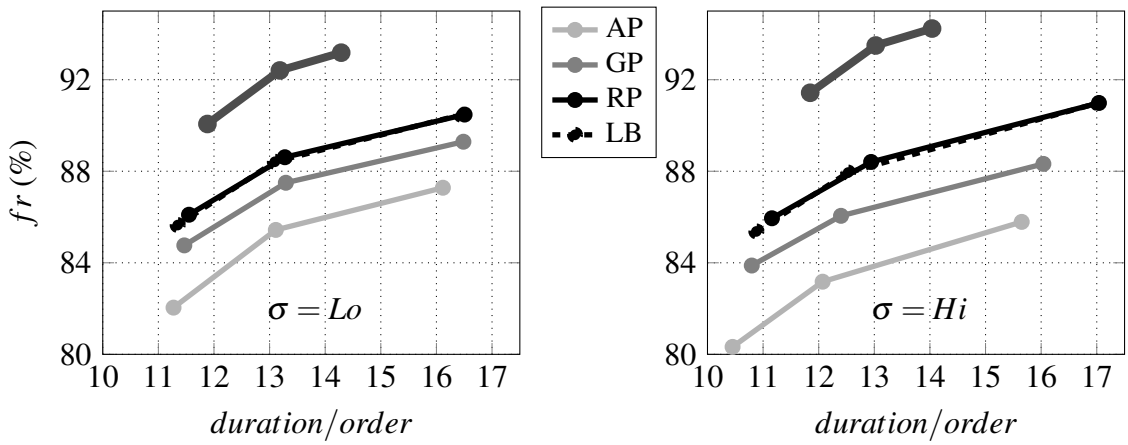


Figure 3.15: Pareto charts showing each policy's average fr and $duration/order$ for different settings of σ

From the *a priori* solution we derive four heuristic policies that differ in their dynamism, from an *a priori* policy with simple recourse to a fully dynamic rollout policy. The first policy, AP, is an *a priori* solution with a order-skipping recourse; the second policy is a direct rollout of the *a priori* solution, RP; the third, RRP, is a restricted version of RP that waits until the first dispatch wave given by AP to roll out; and the fourth one is a computationally cheaper alternative to RP that rolls out a prize-collecting TSP guided by the initial *a priori* solution.

We designed a first set of computational instances to test these policies under different settings of geography, problem size, level of information disclosed before the operation starts (degree of dynamism), and variability of the arrival process. Our computational experiments indicate that the performance of the *a priori* policy has costs that are 23.1% over our perfect information bound, even if we include heuristic order-skipping improvements. Its order fill rate is 81.6%. The benefit of a fully dynamic policy over an *a priori* one can be significant; in our experiments, RP is able to cut AP's cost by 9.1% on average, yielding an average gap of 12.1% over the lower bound. It also improves the order fill rate to 86.6%, which is highly desirable for SDD services. Part of this benefit is achieved by increasing recourse opportunities to catch newly arrived orders, and by sacrificing a small amount of routing efficiency. This also shows that the marginal benefit of RP

is concentrated in order coverage and not in vehicle routing costs. A more surprising benefit is that dynamic solutions also reduce the gap standard deviation by 3.6%, giving consistently good solutions compared to AP. The cost reduction and fill rate increase are especially significant when the instance dynamism is high (smaller p_{start}), and the order arrival variability is high (bigger p_{out} and σ); this can be commonplace in SDD systems. We also found little benefit and few occurrences of the vehicle waiting at the depot during the planning horizon once dispatches begin.

RP's computational effort may prevent its deployment (at least using exact optimization) for bigger problem sizes, *e*.g., $n \geq 50$. In this case, we provide a simpler and computationally efficient dynamic policy (GP) that attains on average two thirds of the cost reduction benefits and fill rate increase over AP. We also show that a practically equivalent result can be achieved by the restricted rollout RRP; there appears to be almost no value in changing the initial dispatch wave established by AP. In general, the DDWP proved quite challenging to solve and future work may consider improvements in heuristic solution of the *a priori* problem using local search and meta-heuristics. This could allow us to solve bigger problems and to make the RP policy practical for on-line deployment in larger instances. Another interesting question is whether we can exploit probabilistic routing methods to solve the *a priori* problem with recourse.

We also empirically studied the tradeoff between two possible SDD objectives, minimizing total cost and maximizing the weighted order fill rate. One might think that these two objectives deliver similar results, since well-sequenced routes leave more vehicle time available to cover more orders. However, we found that one should expect significant sacrifices in vehicle routing efficiency in order to maximize fill rate, and that the distance cost of an additional customer covered becomes more expensive as order coverage increases. In our results, an RP solution that maximizes order coverage and one that minimizes costs have a 50% difference in traveled distance per order and a 5% difference in number of orders covered; also, a solution focusing on coverage makes more dispatches (85% more), has 17% shorter routes on average, and starts dispatching earlier, having on average a 50% longer vehicle utilization throughout the day. This has direct implications on driver

salaries and vehicle maintenance costs. We also observed that, as fill rate becomes more important, the average improvement of dynamic policies over *a priori* policies increases, and sophisticated dynamic policies such as RP improve in relative terms over simpler dynamic policies like GP.

Possible extensions of the DDWP include incorporating vehicle service times at each location or including customer service time windows instead of a deadline at the end of the day. The extension of this model to multiple vehicles also seems natural; having a fleet of vehicles could pool the risk associated with leaving orders unattended and therefore reduce costs, *e*.g., [2]. In general, same-day delivery offers many new challenges to the logistics research community.

CHAPTER 4

ORDER ACCEPTANCE MECHANISMS FOR SAME-DAY DELIVERY

4.1 Introduction

The Dynamic Dispatch Waves Problem (DDWP), defined in [43, 44], seeks to determine a vehicle dispatch plan in a model where requests at potential delivery locations arrive according to some stochastic process, and must be served by a single vehicle operating from a single fulfillment depot over the course of a service day partitioned into a discrete set of feasible dispatch epochs (waves). At any wave when the vehicle is available to dispatch at the fulfillment location, the vehicle can wait for one wave to potentially accumulate and load more delivery requests, or can be dispatched to serve a subset of ready requests. The objective of the DDWP is to minimize vehicle travel cost, plus penalties for delivery requests that are not fulfilled by the end of the operating day.

An implicit assumption in the DDWP is that orders that will not be served via same-day delivery are not formally rejected until the end of the operating day, after the final vehicle dispatch. Such a setting gives the service provider a degree of flexibility that is likely not available in some practical settings. In this chapter, we study a similar but more realistic problem setting in which customers are offered a same-day delivery option when placing an order, and if a customer selects the option then same-day delivery is guaranteed. To do so, we use an *accept or reject* framework: a request is accepted (and thus delivered in the same day), or rejected immediately when received. Such a framework, of course, reduces flexibility and leads to higher costs; we study the magnitude of this cost increase. The DDWP also assumes that order processing times (from picking and packing) at the distribution facility are negligible; thus any request is ready for delivery immediately once it is known. In this research, we include a non-negligible order processing time that delays the dispatch of each accepted order; see Figure 4.1.

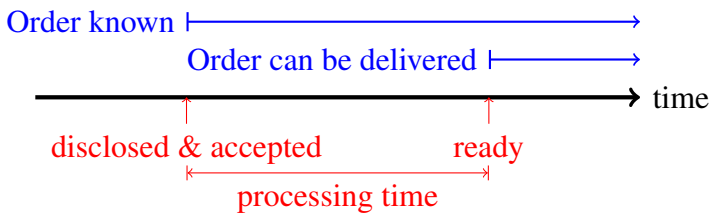


Figure 4.1: Order disclosure time and order ready time for an accepted order.

In this chapter, we formulate and study the Dynamic Dispatch Waves Problem with Immediate Acceptance (DDWP-IA), which integrates order request acceptance with vehicle dispatch and routing decisions for SDD systems with a single vehicle. The DDWP-IA assumes customers place **online order requests** dynamically until a cutoff time during the operating day; each order is not known until its disclosure time. However, we assume that probabilistic information describing potential future order requests is available. Immediately upon receipt of an order request, an **order acceptance mechanism** must choose whether to accept or reject it. Each accepted order must be delivered to its customer location no later than the end of the operating day, after it has been processed by the distribution center and dispatched. An **order dispatch system** decides at each of a number of dispatch *waves* whether to dispatch a vehicle from the distribution center (if available), loaded with a subset of accepted and ready orders. The routing of the dispatched vehicle incurs travel cost, and also determines when the vehicle returns to the distribution center to be loaded again; the order dispatch system is identical to that proposed in [43]. The objective of the DDWP-IA is to minimize vehicle travel costs plus penalties for order rejections.

The order acceptance mechanism and dispatch system are clearly interrelated. For instance, we avoid penalty cost if more orders are accepted earlier in the operating day, but doing so may incur additional travel cost and reduces the flexibility of the dispatch system to accommodate orders that appear later. Conversely, rejecting too many orders early in the day may lead to a vehicle that is under-utilized. Routing economies created by geographic consolidation are also important; accepting an order with a delivery location close to another order awaiting its dispatch may only

lead to a small marginal increase in travel cost and vehicle travel time.

We explore two types of solution policies for the DDWP-IA; *myopic policies* that make acceptance decisions using only information about previously accepted orders, and proactive policies that additionally incorporate probabilistic information regarding potential future order requests. We expect proactive policies to outperform myopic policies, but the magnitude of the improvement is of interest. Moreover, SDD systems operate under a high degree of information dynamism, and there is a need for dynamic decision support tools able to continuously re-plan operations [44]. These *dynamic policies* determine decisions based on the information state at each decision epoch. In contrast, simpler *a priori* policies specify most decisions in advance, and allow only simple changes over time via pre-established recourse rules.

The primary contribution of this paper is to formulate the DDWP-IA, extending the DDWP model from [43], and to develop a solution methodology for this new problem. The solution approach we develop uses a lower bounding procedure that solves deterministic instances of the problem to optimality, and a proactive and dynamic policy that rolls-out an optimal *a priori* solution. In addition to this *a priori* roll-out policy, we also design a fast meta-heuristic for order acceptance as an alternative that requires far less computational effort. We compare our approaches with two simpler benchmark approaches: a myopic re-optimization policy that assumes no information regarding future arrivals, and a myopic policy that fixes dispatch waves according to an initial *a priori* solution, but dynamically assigns order to dispatches and routes. A computational study provides results that are used to estimate the cost of imposing immediate order acceptance into SDD systems, and to compare our best approaches with the performance of the simpler benchmark approaches. A second contribution is a study of the impact of distribution center order processing time on the performance of SDD systems.

The remainder of the chapter is organized as follows. Section 4.2 defines notation and formulates the DDWP-IA. Section 4.3 focuses on the deterministic problem, and proposes an approach for it that will be used in the *a priori* model. Section 4.4 then proposes policies for solving the

dynamic and stochastic problem, and Section 4.5 provides a heuristic to speed up the order acceptance step. Finally, Section 4.6 presents the results of a computational study, and Section 4.7 provides conclusions.

4.2 Problem formulation

The DDWP-IA works within a service area defined by a finite set of nodes $I := \{1,\dots,|I|\}$ representing geographic customer locations, *e*.g., neighborhoods or city blocks; let $I \cup \{0\}$ be the extended set that includes the depot ($i = 0$). Moreover, let $E := I \cup \{0\} \times I \cup \{0\}$ be the set of edges between all pairs of nodes so that $\mathscr{G} := (I \cup \{0\}, E)$ defines a complete undirected graph representing direct routes between locations. Traversing an edge $e \in E$ takes d_e time and costs γd_e; we assume for simplicity that time and cost values are proportional to each other, non-negative, and that they satisfy the triangle inequality.

The problem's **operating period** is represented as the continuous set $\mathscr{T} = [T,0]$; time is counted backwards as a resource being consumed so that $t = T$ represents the start of the operating day and $t = 0$ when it ends. Let $\mathscr{W} := \{W,\dots,1\}$ be a discrete set of waves, *i*.e., feasible dispatch times, placed as a layer on top of $\mathscr{T}$ such that each wave w occurs at time $t_w \in \mathscr{T}$, with $t_W = T$; the value w is also counted backwards and represents the waves-to-go before the terminal wave $w = 0$ at $t_0 = 0$. Define the extended wave set $\mathscr{W}_0 := W \cup \{0\}$ and the set of upcoming waves at time t as $\mathscr{W}(t) := \{w \in \mathscr{W} : t_w \leq t\}$; similarly let $\mathscr{W}_0(t) := \{w \in \mathscr{W}_0 : t_w \leq t\}$.

Customer requests arrive over time according to a counting process for each node $i \in I$ defined by the random variables $N_i(t) \in \mathbb{Z}_+, t \in \mathscr{T}$. We assume node-independent counting processes stopping after a cut-off time t^{ct}, and satisfying the memoryless property, meaning that the probability distribution of the k^{th} arrival time $\tau_i^k \in \mathscr{T}$ at node i given that it has not occurred by time t is completely independent of the history $\{\tau_i^j\}_{j=0}^{k-1}$. Mathematically, we assume for each $i \in I$

$$\mathbb{P}(\tau_i^k \geq t' | \tau_i^k < t, \tau_i^j = t^j, j = 0..k-1) = \mathbb{P}(\tau_i^k \geq t' | \tau_i^k < t) = \mathbb{P}(\tau_i \geq t' | \tau_i < t). \tag{4.1}$$

An example of (4.1) is a Poisson process truncated after the cutoff time t^{ct} with rate λ_i orders per time unit, where

$$\mathbb{P}(\tau_i \geq t' | \tau_i < t) = \begin{cases} 1 - e^{-\lambda_i (t - t')} & t' \in [t^{ct}, t] \\ 1 - e^{-\lambda_i (t - t^{ct})} & t' \in [0, t^{ct}] \end{cases}. \tag{4.2}$$

Order acceptance decisions occur immediately after each delivery request realization at a time $t \geq t^{ct}$ demanding service to a node $i \in I$. If accepted, the order must be processed and dispatched to the customer between its release time from the depot $t - p$ and the end of the day; the parameter $p \in \mathbb{R}_+$ is the request processing time. We study the problem with a constant p, but our model can handle any node-dependent $p_i, i \in I$. This implies that the order must be dispatched at a wave $w \in \mathscr{W}(t - p)$. A rejected request is lost, incurring a node-dependent penalty cost β_i.

Vehicle dispatch decisions may occur at times t_w for some wave $w \in \mathscr{W}$. If the system dispatches a vehicle to execute a delivery route $r = \{0, i^r_1, \ldots, i^r_{m_r}, 0\}$ with m_r customer visits, its transportation cost is $\gamma d(r) := \sum_{j=1}^{m_r+1} \{\gamma d_{(i^r_{j-1}, i^r_j)}\}$ and spends $t(r) := \sum_{j=1}^{m_r+1} \{u_{i^r_{j-1}} + d_{(i^r_{j-1}, i^r_j)}\} = d(r) + \sum_{j=0}^{m_r} u_{i^r_j}$ time units, where $u_i, i \in I$ is a node service time assumed to be independent of the number of customer requests served per node. The vehicle returns to the depot at time $t_w - t(r)$ and becomes available for dispatch again at the earliest wave after the return time defined as

$$q(w, r) := \begin{cases} \max\{k \in \mathscr{W}_0(t_w - t(r))\} & \text{if } m_r \geq 1 \\ w - 1 & \text{else} \end{cases}. \tag{4.3}$$

The depot's service time u_0 represents vehicle set-up and load times. For notational purposes we say that a route visits node i if $i \in r$. By the triangle inequality, we do not increase travel time if we consolidate all requests at node i into one visit, so we consider only elementary routes. Also, we assume no time difference between servicing one or multiple requests per node, so without loss of optimality a route serves all open requests at a visited node.

4.2.1 System states

The system's state is described by a vector $s = (t, \mathbf{a}, w) \in \mathscr{S}$ at any time $t \in \mathscr{T}$, where $\mathscr{S}$ is the set of all feasible states. The parameter $\mathbf{a} \in \mathscr{W}^{|I|}$ is the vector of open commitments, and each component indicates the earliest possible wave $a_i \in \mathscr{W}$ a visit to node i at which the vehicle can cover all its pending accepted orders; if $a_i = \infty$, no order is waiting for service at i. Given $\mathbf{a}$, let $I_{\mathbf{a}} := \{i \in I : a_i < \infty\}$ be the subset of nodes with open orders. The parameter $w \in \mathscr{W}_0(t)$ represents the earliest upcoming wave when the vehicle is available at the depot, and determines the next dispatch decision. The state s does not carry disaggregated information regarding specific orders at i; without loss of optimality, all such orders can be covered in one visit occurring no earlier than $\min(w, a_i)$.

Any state s is certified as feasible by a dispatch plan π capable of feasibly serving all commitments in $\mathbf{a}$, and defined by a set of elementary routes $\{r_k^\pi\}_{k \in \mathscr{W}^\pi}$ indexed by its set of dispatch waves $\mathscr{W}^\pi \subset \mathscr{W}$. Formally, a state $(t, \mathbf{a}, w) \in \mathscr{S}$ is feasible if and only if $t \in \mathscr{T}, w \in \mathscr{W}_0(t)$ and there exists a dispatch plan π satisfying three conditions:

1. Plan π starts after wave w, *i*.e., if $k \in \mathscr{W}^\pi$, then $k \leq w$.

2. Plan π covers all commitments, *i*.e., if $i \in I_{\mathbf{a}}$, then $\exists k \in \mathscr{W}^\pi$ such that $k \leq a_i$ and $i \in r_k^\pi$.

3. Routes in π don't overlap in time, *i*.e., for $k_1, k_2 \in \mathscr{W}^r : k_1 > k_2$ we have $k_2 \leq q(k_1, r_{k_1}^\pi)$.

Let $\mathscr{P}(a, w)$ be the set of feasible dispatch plans for state (t, a, w). The state space is completely defined by

$$\mathscr{S} = \{(t, a, w) : t \in \mathscr{T}, w \in \mathscr{W}_0(t), a \in \mathscr{W}^{|V|}, \exists \pi \in \mathscr{P}(a, w)\}. \tag{4.4}$$

We also define the extended state space

$$\mathscr{S}' = \{(t, a, w, \pi) : t \in \mathscr{T}, w \in \mathscr{W}_0(t), a \in \mathscr{W}^{|V|}, \pi \in \mathscr{P}(a, w)\}, \tag{4.5}$$

where all feasible states are combined with all feasible dispatch plans. A plan's feasibility condition only depends on $\mathbf{a}$ and w; if π is feasible for state $(t,\mathbf{a},w)$, then it is also feasible for any state $(t',\mathbf{a},w) : t' \in (t,t_w]$. Therefore, a plan's feasibility is preserved over time until the next dispatch wave, and is useful in heuristic design. If p is negligible, then all previously accepted and pending orders are open for dispatch at the current time t and, therefore, any feasible state $(t,\mathbf{a},w)$ has a plan π consisting of one single vehicle dispatch after w covering all commitments in $\mathbf{a}$. This is not the case when $p > 0$ and a dispatch plan is a solution the VRP with release dates [9, 24].

4.2.2 Actions, transitions and costs

An accept/reject decision is immediately taken after a request realizes at node $i \in I$ in state $s = (t,\mathbf{a},w)$. Denote the order's earliest release wave from the depot as $b := \max\{k \in \mathscr{W}(t-p)\}$. Rejecting an order costs β_i, but keeps the system's state unaltered. Accepting it is free of charge, but can only be performed if the post-decision state $(t,\mathbf{a}^i,w)$ remains feasible, where $\mathbf{a}^i$ is the updated vector of commitments with $\mathbf{a}^i_j = a_j, \forall j \neq i$ and $a^i_i = \min\{a_i, b\}$. Acceptance is always feasible when the new order satisfies $b \geq \max\{a_i, w\}$.

A dispatch decision is taken at any state $(t_w,\mathbf{a},w)$ where time matches the next decision wave. If we restrict ourselves to optimal Traveling Salesman Problem (TSP) routes, the vehicle dispatch is fully determined by a subset $Q \in I$ of node visits representing an optimal tour over $Q \cup \{0\}$, defined by $r^*(Q) := \text{argmin}_{\{r:Q\in r\}} t(r)$, minimizing cost $\gamma d^*(Q) := \gamma d(r^*(Q))$ and maximizing the return wave $q^*(w,Q) := q(w,r^*(Q))$. A feasible dispatch Q at wave w keeps the system in a feasible post-decision state $(t,\mathbf{a}(w,Q),q^*(w,Q)) \in \mathscr{S}$, where $\mathbf{a}(w,Q)$ is the updated vector of commitments defined as

$$a(w,Q)_i := \begin{cases} \infty & i \in Q \text{ and } a_i \geq w \\ a_i & \text{otherwise} \end{cases}. \tag{4.6}$$

$Q = \emptyset$ is the special action that represents waiting at the depot at zero cost and sets $q^*(w,\emptyset) = w-1$. Figure 4.2 depicts a flowchart of all system transitions and actions connected to a state $(t,\mathbf{a},w) \in$

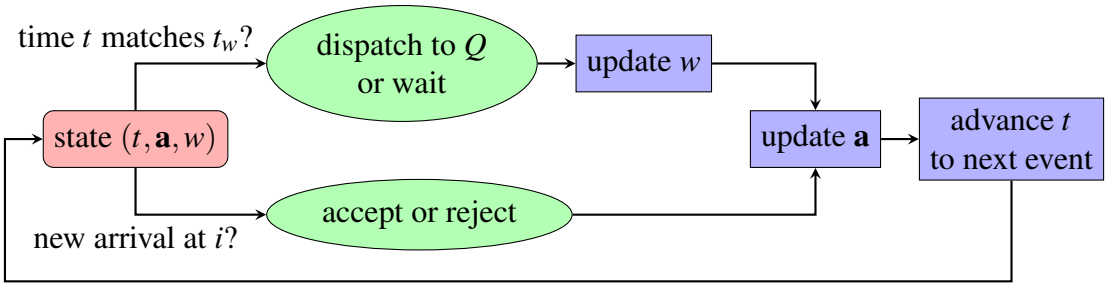


Figure 4.2: Flowchart of system transitions and actions in state (t,a,w) for the DDWP-IA

$\mathscr{S}$.

4.2.3 Dynamic programming model

We model the DDWP-IA as a semi-Markov decision process (SMDP) that generalizes a Markov decision process (MDP) by modeling time as a continuous variable. Specifically, we assume that the time spent between consecutive decision epochs follows a probability distribution that depends on the previous state and action taken [57]. Given a time $t \in T$, let $\phi(i,t',t)$ be the probability density of the next order arriving at time $t' < t$ at node $i \in I$, and let $\psi(t',t)$ be the probability that no order arrives at all between t and t'. We have

$$\phi(i,t',t) = \mathbb{P}(\{\tau_i = t'\} \bigcap \{\tau_i > \tau_j, \forall j \in I : j \neq i\} | \tau_i < t), \tag{4.7}$$

$$\psi(t',t) = \prod_{i \in I} \mathbb{P}(\tau_i < t' | \tau_i < t); \tag{4.8}$$

and in the particular case of the Poisson process we have

$$\phi(i,t',t) = \begin{cases} \lambda_i e^{(\sum_{i' \in I} \lambda_{i'})(t-t')} & \text{if } t' \in [t^{ct}, t] \\ 0 & \text{else} \end{cases} \tag{4.9a}$$

$$\psi(t,t') = e^{\sum_{i \in I} \lambda_i (t - \max(t', t^{ct}))}. \tag{4.9b}$$

Let $C(t,\mathbf{a},w)$ be a function representing the optimal expected cost-to-go at state $(t,\mathbf{a},w)$, and let $C^*(I^0)$ be the optimal expected cost-to-go at time T with a set $I^0 \subseteq I$ of previously accepted visits ready at wave W. Let $\mathbf{a}^0$ satisfy $a_i^0 = W, i \in I^0$ and $a_i^0 = \infty$ otherwise. The SMDP defined in (4.10) computes $C^*(I^0)$ recursively over time with

$$C^*(I^0) = C(T,\mathbf{a}^{I^0},W), \tag{4.10a}$$

$$C(0,\infty,0) = 0, \tag{4.10b}$$

$$C(t_w,\mathbf{a},w) = \min_{\substack{Q\subseteq I:\\ (t,\mathbf{a}(w,Q),q^*(w,Q))\in\mathscr{S}}} \{\gamma d^*(Q) + C(t_w,\mathbf{a}(w,Q),q^*(w,Q))\}, \qquad \forall (t_w,\mathbf{a},w) \in \mathscr{S} \tag{4.10c}$$

$$C(t,\mathbf{a},w) = \psi(t,t_w)C(t_w,\mathbf{a},w) + \sum_{i\in I}\int_{t'=t_w}^{t} \phi(i,t',t)\tilde{C}(t',\mathbf{a},w,i)dt', \qquad \forall (t,\mathbf{a},w) \in \mathscr{S} : t > t_w, \tag{4.10d}$$

$$\tilde{C}(t,\mathbf{a},w,i) = \min\{\beta_i + C(t,\mathbf{a},w), C(t,\mathbf{a}^i,w) : (t,\mathbf{a}^i,w) \in \mathscr{S}\}, \qquad \forall i \in I, (t,\mathbf{a},w) \in \mathscr{S} : t > t_w, \tag{4.10e}$$

where Equation (4.10a) defines the optimal cost and Equation (4.10b) sets the terminal cost equal to zero. Equation (4.10c) models the state transition during a dispatch decision and states that the cost-to-go at a state $(t_w,\mathbf{a},w)$ is equal to the minimum sum of dispatch cost $\gamma d^*(Q)$ plus the post-decision cost-to-go $C(t_w,\mathbf{a}(w,Q),q^*(w,Q))$ over all feasible dispatched subsets $Q \subseteq I$: $(t_w,\mathbf{a}(w,Q),q^*(w,Q)) \in \mathscr{S}$. Equation (4.10d) models the evolution of the system over time and states that any cost-to-go $C(t,\mathbf{a},w) : t > t_w$ is equal to the cost-to-go in the next dispatch decision $C(t_w,\mathbf{a},w)$ if no orders arrive between t and t_w (with probability $\psi(t,t_w)$), and equal to $Q(t',\mathbf{a},w,i)$ if the next order realizes at node i and time $t' \in [t_w,t]$ (with probability $\phi(i,t',t)$); $\tilde{C}(t,\mathbf{a},w,i)$ represents the cost-to-go immediately before the acceptance decision defined in Equation (4.10e), equal to the minimum cost-to-go between rejecting the order $C(t,\mathbf{a},w) + \beta_i$ and accepting it $C(t,\mathbf{a}^i,w)$ (if feasible).

Model (4.10) clearly shows its intractability; it has an uncountable number of states, expo-

nentially many dispatch decisions for each state $(t_w, \mathbf{a}, w) \in \mathscr{S}$, and an uncountable number of terms in the expectations that model transitions in time. Also, it is NP-Hard to evaluate $d^*(Q)$ and $q^*(w,Q)$, which involve solving a TSP over $Q \cup \{0\}$. We next develop approximate solutions to the DDWP-IA, where each action's computing time does not grow with the continuous time interval $\mathscr{T}$.

4.3 The deterministic DDWP and lower bounds

We first study the simplified problem when the number of orders and their arrival times at customer nodes are disclosed before the operation starts. Under this setting all relevant information is available beforehand and all request acceptance and vehicle dispatch decisions can be made beforehand. So, the DDWP-IA model collapses to the deterministic variant of the DDWP discussed in [43].

Suppose all order arrival times $\{\tau_i^1, \tau_i^2, \ldots\}$ are known at $t = T$ for each node $i \in I$; then each variable $N_i(t)$ is a deterministic quantity and the number of requests ready at any wave $w \in \mathscr{W}$ $n_{i,w} := N_i(t_w + p)$ is known beforehand. Figure 4.3 provides an example for a particular node i where functions $N_i(t)$ and $n_{i,w}$ are depicted in a seven-wave horizon, a cutoff time $t^{ct} = t_2$, and a processing time p.

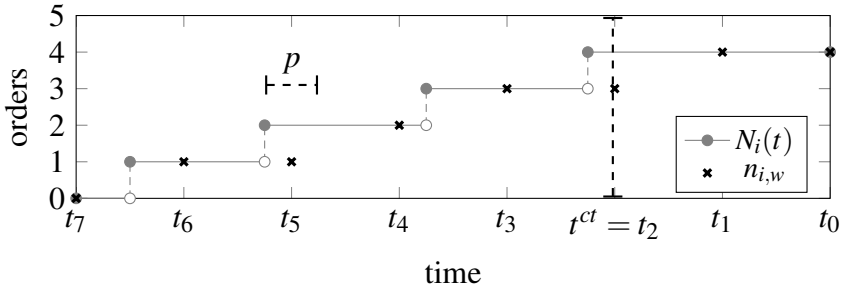


Figure 4.3: Evolution of the orders disclosed over time ($N_i(t)$) and orders ready for each dispatch wave ($n_{i,w}$) for a particular realization (ω) of arrivals at node $i \in I$.

In the deterministic DDWP it is still infeasible to serve a request before its ready wave, meaning

the number of orders that can be accepted and dispatched to node $i \in I$ by wave $w \in \mathscr{W}$ is at most $n_{i,w}$. We cite some properties from [43] that any feasible deterministic solution satisfies without loss of optimality:

1. *Any node $i \in I$ is visited by a vehicle at most once.* Otherwise, there exist at least two vehicle dispatches at waves w_1 and $w_2 < w_1$ visiting i and one could skip i from the dispatch at wave w_1, covering these requests in the dispatch at w_2. By the triangle inequality, the change does not increase travel cost or travel time.

2. *The vehicle does not wait at the depot after the first dispatch.* If the vehicle waits for $k > 1$ waves after returning from a dispatch at a wave w, one can delay each dispatch occurring prior to wave w by exactly k waves, serving the same orders at a later point.

In the deterministic case, a dispatch plan π completely defines the number of orders accepted at each node. Consider all plan routes r_k^π dispatched at waves $k \in \mathscr{W}^\pi$; the number of accepted orders at node $i \in I$ is equal to $n_{i,w^*_{\pi,i}}$, where $w^*_{\pi,i} = \min\{k \in \mathscr{W}^\pi : i \in r_k^\pi\}$ is the latest visit to i in π, or 0 if π does not visit i.

Define $C^D(I^0, \mathbf{N}(t))$ as the minimum cost of the deterministic DDWP, where I^0 is the set of initially accepted visits and $\mathbf{N}(t) = \{N_1(t), \ldots, N_{|I|}(t)\}$ is the vector function of all order realizations over time $t \in \mathscr{T}$; define $I_w := \{i \in I : w \geq h(0,i)\} \subset I$ and $E_w := \{e \in E : w \geq h(e)\} \subset E$ for each $w \in \mathscr{W}$ as the sets of feasible nodes and edges for a vehicle dispatch at wave w, where $h(i,j) = \min\{w \in W : t_w \geq \mathbb{I}_{(i>0,j>0)}(s_0) + \mathbb{I}_{(i>0)}(t_{\{0,i\}}) + \mathbb{I}_{(j>0)}(t_{\{0,j\}}) + t_{\{i,j\}} + s_i + s_j\}$ is edge's $\{i,j\} \in E$ latest possible wave. Also, define the cut set $E_w(S) = \{\{i,j\} \in E_w : i \in S, j \notin S\}$, for any subset $S \subseteq I_w$; and define $\bar{t}_e = t_e + 0.5s_i + 0.5s_j$ as the adjusted time spent at edge $e = \{i,j\} \in E$ considering service times. The Integer Program (IP) in (4.11) formulates the deterministic DDWP

that computes the optimal dispatch plan $\pi(I^0, \mathbf{N}(t))$,

$$C^D(I^0,\mathbf{N}(t)) = \min_{\{\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{v},\mathbf{s}\}} \sum_{i\in I}\beta_i\{N_i(0)z_i + \sum_{w=h(0,i)}^{W}(N_i(0)-n_{i,w})y_i^w\}$$

$$+ \sum_{w\in\mathscr{W}}\sum_{e\in E_w}\gamma d_e x_e^w \tag{4.11a}$$

$$\text{s.t.} \sum_{w=h(i,0)}^{W} y_i^w = 1, \qquad \forall i \in I^0 \tag{4.11b}$$

$$z_i + \sum_{w=h(i,0)}^{W} y_i^w = 1, \qquad \forall i \in I \tag{4.11c}$$

$$\sum_{e\in E_w(0)} x_e^w \le 2, \qquad \forall w \in \mathscr{W} \tag{4.11d}$$

$$\sum_{e\in E_w(S)} x_e^w \ge 2y_i^w, \qquad \forall w \in \mathscr{W}, \forall S \subseteq I_w, \forall i \in S \tag{4.11e}$$

$$\sum_{e\in E_w} \bar{t}_e x_e^w \le \sum_{k<w}(t_w - t_k)v_k^w, \qquad \forall w \in \mathscr{W} \tag{4.11f}$$

$$\sum_{k<W} s_k + \sum_{k<W} v_k^W = 1, \tag{4.11g}$$

$$\sum_{k<w} v_k^w = \sum_{k>w} v_w^k + s_w, \qquad \forall w \in \mathscr{W}\setminus\{W\} \tag{4.11h}$$

$$v_k^w \in \{0,1\}, \qquad \forall w,k \in \mathscr{W}_0 : k < w \tag{4.11i}$$

$$s_k \in \{0,1\}, \qquad \forall k \in \mathscr{W}_0\setminus\{W\} \tag{4.11j}$$

$$z_i \in \{0,1\}, \qquad \forall i \in I \tag{4.11k}$$

$$y_i^w \in \{0,1\}, \forall i \in I_w, \text{ and } x_e^w \in \{0,1,2\}, \forall e \in E_w, \qquad \forall w \in \mathscr{W} \tag{4.11l}$$

where variable z_i is equal to 1 if node i isn't visited, and 0 otherwise; y_i^w is equal to 1 if a dispatch at wave w visits node i, and 0 otherwise; x_e^w is equal to $m \in \{0,1,2\}$ if the vehicle traverses edge e m times at a dispatch in wave w; v_k^w is equal to 1 if a dispatch at w returns at wave k, and 0 otherwise; and s_k is equal to 1 if the vehicle waits at the depot until wave k, and 0 otherwise ($s_0 = 1$ implies an empty plan with no dispatch throughout the horizon). Constraints (4.11b) force all initially accepted visits in I^0 and (4.11c) guarantee visiting each node i at most once at wave w.

Constraints (4.11d) - (4.11e) guarantee that vector $\mathbf{x}^w$ defines a feasible tour only visiting nodes selected by the vector $\mathbf{y}^w$. Constraints (4.11f) force routes to satisfy durations limits determined by v_k^w. Finally, wave flow constraints (4.11g)-(4.11h) enforce vehicle conservation throughout time. To simplify upcoming notation in this article, we say that any vector of variables $(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{v}, \mathbf{s})$ satisfying constraints (4.11c) through (4.11l) belongs to domain $\mathscr{D}(\mathscr{W})$.

Problem (4.11) generalizes the Prize-Collecting TSP (PC-TSP) and is NP-Hard. However, the problem's size is independent of the time horizon $\mathscr{T}$ and only depends on the number of waves W. In [43] the authors show how to optimize over (4.11) using standard branch and cut approaches for a routing problem based on arc variable formulations and dynamic addition of the subtour elimination constraints (4.11e); they also estimate a lower bound on the optimal expected cost $C^*(I^0)$ with a Perfect Information Relaxation (PIR) $\mathbb{E}_{\omega}[C^D(I^0, \mathbf{N}(t,\omega))] \leq C^*(I^0)$ that disregards the "non-anticipative" dynamics of any feasible policy for the stochastic DDWP-IA problem and computes a different solution for each possible scenario realization ω of the random variables [19, 61].

4.4 Solution policies for the stochastic case

Now, we develop a framework to construct heuristic policies for the DDWP-IA, where a system state is maintained that includes a feasible dispatch plan (with potentially multiple planned trips) serving all accepted delivery requests along with a set of potential future delivery requests that have not yet realized. This dispatch plan is used to guide order acceptance and vehicle dispatch decisions, and it is updated dynamically when new information is available. Each policy P constructs an initial dispatch plan π that is feasible for the initial state s^0, defined by routes r_k^{π} dispatched at waves $k \in \mathscr{W}^r$, each visiting a subset $Q_k^{\pi} \subseteq I$ of nodes. After the operation starts, policy P dynamically updates π to keep it feasible throughout all states $s^j \in \mathscr{S}$, $j = \{0,1,2,3,4,\ldots\}$ visited by the system, so that $(s^j, \pi) \in \mathscr{S}'$.

Define the list of online request arrivals for a given realization of the random variables ω as

$\mathscr{L}(\omega) = \{1, \ldots, L(\omega)\}$, ordered by arrival time so that element k represents the k-th request arrival time $\tau(\omega,k) \in \mathscr{T}$ at node $i(\omega,k) \in I$. Algorithm 10 provides a high-level pseudo-code description to compute the cost $C^P(I^0,\omega)$ of a policy P given ω and the set of previously accepted visits I^0. P is determined by three functions used to update the dispatch plan: **IniPlan**, **ArrivalUpdate** and **DispatchUpdate**.

Algorithm 10 Implementation of a generic policy P

1: **procedure** EXECUTEPOLICY(P, I^0,ω)
2: Initialize parameters: $k \leftarrow 1$, $C \leftarrow 0$, $s := (t, \mathbf{a}, w) \leftarrow (T, a^0, W)$.
3: Initialize plan: $\pi \leftarrow$**INIPLAN**(P,I^0)
4: **do**
5: $t \leftarrow \max\{\tau(\omega,k), t_w\}$
6: **if** ($\tau(\omega,k) \geq t$) **then** an order arrives at time t at node $i \leftarrow i(\omega,k)$.
7: Get release wave: $b \leftarrow \max\{x \in \mathscr{W}_0(t-p)\}$
8: Update plan: $\pi \leftarrow$**ARRIVALUPDATE**(P, π, s, i, b)
9: **if** (π covers i after b) **then** accept: $a_i \leftarrow \min\{a_i, b\}$
10: **else** reject: $C \leftarrow C + \beta_i$
11: k++;
12: **else**
13: Update plan: $\pi \leftarrow$**DISPATCHUPDATE**(P, π, s)
14: **if** $w \in \mathscr{W}^\pi$ **then**
15: dispatch route r_w^π visiting nodes Q_w^π: $C \leftarrow C + \gamma t(r_w)$, $\mathbf{a} \leftarrow \mathbf{a}(w, Q_w^\pi), w \leftarrow q(w, r_w^\pi)$,
16: delete route r_w^π from plan π.
17: **else** wait at the depot: $w \leftarrow w - 1$
18: **while** ($k \leq L(\omega)$ or $w > 0$)
19: **return** C

Algorithm 10 initializes all variables in line 2 and calls function **IniPlan**, which returns an initial plan π in line 3. Then, it runs an event-based simulation that advances to the time of the next event in line 5. If the event is a request arrival, it gets its information and updates the plan π, calling **ArrivalUpdate** in line 8; if the updated plan covers the new request it accepts it in line 9 and updates the state s; otherwise, it rejects it and pays the penalty cost in line 10. On the other hand, if the event is a dispatch decision, it updates the plan, calling **DispatchUpdate** in line 13 and,

if w belongs to the set of planned dispatches $\mathscr{W}^{\pi}$, it dispatches route r_w^{π}, making all cost, state, and plan updates in lines 15 and 16; otherwise, the vehicle waits at the depot one wave by subtracting one wave to state s in line 17. The simulation returns the cost C for realization ω after there are no more orders to process in the list of arrivals and there are no more waves remaining.

The online computing time of Algorithm 10 depends directly on functions **ArrivalUpdate** and **DispatchUpdate**. In particular, the speed of the former is critical to allow fast accept/reject decisions. We next describe multiple policies that differ in how they implement functions **IniPlan**, **ArrivalUpdate** and **DispatchUpdate**.

4.4.1 Myopic policy

The first policy disregards all available probabilistic information regarding future request arrivals, but makes optimal decisions with respect to the information disclosed so far. Assume that a new request realizes at time t and node i when the system is in state $(t, \mathbf{a}, w) \in \mathscr{S}$; the myopic policy (MP) solves the IP defined in (4.12) to get its optimal dispatch plan $\pi^M(\mathbf{a}, w, i, b)$ and, therefore, accepts the new request if it is covered by this plan before $b = \max\{k \in \mathscr{W}_0(t-p)\}$.

$$\min_{\{\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{v},\mathbf{s}\}\in\mathscr{D}(\{1,\ldots,w\})} \beta_i\{z_i + \sum_{k=\max(b+1,h(0,i))}^{w} y_i^k\} + \sum_{k=1}^{w}\sum_{e\in E_k} \gamma d_e x_e^k \tag{4.12a}$$

$$\text{s.t.} \sum_{k=h(0,i)}^{\min\{a_i,w\}} y_i^k = 1, \qquad \forall i \in I_{\mathbf{a}}. \tag{4.12b}$$

The objective (4.12a) minimizes vehicle travel cost plus a penalty paid if the solution does not dispatch to node i after wave b. The plan is forced to belong to all feasible plans starting from wave w, *i*.e., $\{\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{v},\mathbf{s}\} \in \mathscr{D}(\{1,\ldots,w\})$ and constraints (4.12b) guarantee that we cover all previous commitments in vector $\mathbf{a}$. The function **ArrivalUpdate**(MP, π, s, i, b) defined in Algorithm 11 formally defines MP and optimizes problem (4.12) to update the feasible dispatch plan starting from the previous plan; **DispatchUpdate**(MP, π, s) returns the same plan π entered as argument.

Finally, function **IniPlan**(MP, I^0) is defined in Algorithm 12 and determines an initial plan π with a single route equal to an optimal TSP route over $I^0 \cup \{0\}$ dispatched at the latest possible wave $k^* = \min\{k \in \mathscr{W}_0 : t_k \geq t^*(I^0)\}$.

Algorithm 11 Dispatch plan update upon arrival for MP

1: **procedure** ARRIVALUPDATE(MP, π, s, i, b)
2: Solve problem (4.12) with initial feasible solution π.
3: **return** $\pi^M(\mathbf{a}, w, i, b)$, the optimal to (4.12) dispatch plan.

Algorithm 12 Initial dispatch plan for MP

1: **procedure** INIPLAN(MP,I^0)
2: Solve a TSP over $I^0 \cup \{0\}$ and get r^{TSP}, the optimal TSP route
3: **return** single route r^{TSP} dispatched at $k^* = \min\{k \in \mathscr{W}_0 : t_k \geq t^*(I^0)\}$.

A myopic plan suffers from a significant drawback; it tends to build a plan consisting of one single and long dispatch route, leaving few recourse possibilities (or none). It is consistent with objective (4.12a), focused on consolidating all accepted dispatches, and does not consider the costs for rejecting potential future request arrivals. We are interested in a solution with multiple returns to the depot to create better recourse opportunities, and we can thus heuristically set a maximum route duration d^{max} to enforce this in (4.12)

$$v_k^w = 0, \qquad \forall w, k \in \{0, 1, \ldots, w'\} : k < w \text{ and } t_w - t_k > d^{max}. \tag{4.13}$$

The value of parameter d^{max} must be calibrated beforehand.

4.4.2 *A priori* policy

Now we present an optimal *a priori* policy (AP) in which a dispatch plan π is determined beforehand, using all probabilistic information available at time $t = T$. We accept each request released at a wave and node covered by the *a priori* plan. As in [43], we plan an optimal *a priori* policy in

which no recourse actions are allowed. The dispatch cost of such a policy is known beforehand, while the penalties paid for unserved requests depend on future request realizations; however, one can compute expected penalty costs based on the planned route at each dispatch wave. Define the expected number of requests realized at node $i \in I$ and released by wave $w \in \mathscr{W}_0$ as the discrete function $\bar{n}_{i,w} := \mathbb{E}(N_i(t_w + p))$; for a Poisson process with a rate of λ_i orders per time and cutoff time t^{ct} we have $\bar{n}_{i,w} = \lambda_i \max(0, T - \max(t^{ct}, t_w + p))$; an example with four expected orders over the horizon and $t^{ct} = t_2$ is depicted in Figure 4.4.

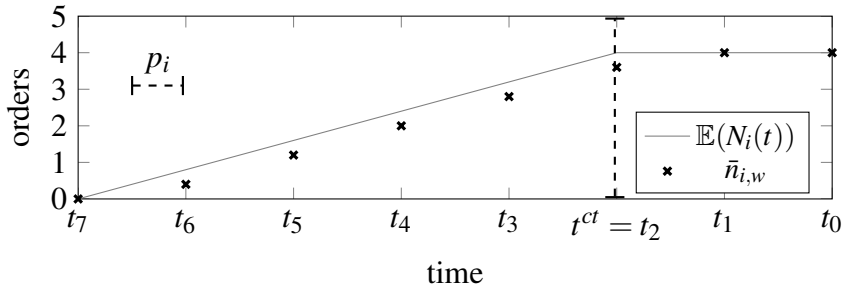


Figure 4.4: Expected orders realized at time t ($\mathbb{E}(N_i(t))$) and expected orders ready for dispatch at wave w ($\bar{n}_{i,w}$) at a particular node $i \in I$ for a particular Poisson arrival process.

The *a priori* problem is equivalent to solving a deterministic DDWP instance with $\bar{n}_{i,w}$ orders ready at node i and wave w, so we can assume that an optimal *a priori* plan will visit each location i at most once and will not wait at the depot after the first dispatch decision. The IP that solves for this optimal *a priori* plan is

$$\min_{\{\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{v},\mathbf{s}\} \in \mathscr{D}(\mathscr{W})} \sum_{i \in I} \beta_i \{\mathbb{E}(N_i(0)) z_i + \sum_{w=h(0,i)}^{W} (\mathbb{E}(N_i(0)) - \bar{n}_{i,w}) y_i^w\} + \sum_{w \in \mathscr{W}} \sum_{e \in E_w} \gamma d_e x_e^w \tag{4.14a}$$

$$\text{s.t.} \sum_{w=h(i,0)}^{W} y_i^w = 1, \qquad \forall i \in I^0; \tag{4.14b}$$

it shares its feasible region with (4.11), but has a different objective determined by expected future requests ready at node i and wave w for each $i \in I$ and $w \in \mathscr{W}$. In case the plan does not visit node i, the expected penalty cost is equal to β_i times $\mathbb{E}(N_i(0))$, *i*.e., the expected requests disclosed by

time $t = 0$; in case it visits node i at wave w, the expected penalty is β_i times $\mathbb{E}(N_i(0)) - \bar{n}_{i,w}$, the expected requests disclosed by time 0 that were not ready by wave w. The function **IniPlan**(AP, I^0) returns an optimal *a priori* plan solving (4.14); function **ArrivalUpdate**(AP, π, s, i, b) produces no change to the plan; and **DispatchUpdate**(AP, π, s) improves the performance of the *a priori* policy at each dispatch wave $w \in \mathscr{W}^\pi$ in state s by removing from route r_w^π all nodes i having no realized orders at w.

4.4.3 Myopic policy with fixed *a priori* dispatch

We next present a myopic policy (MPF) that predetermines a subset of vehicle dispatch waves $\mathscr{W}$ at $t = T$ based on the *a priori* solution. Intuitively, MPF may outperform AP through re-optimization of the plan, and may correct MP's myopic dispatch structure by incorporating probabilistic information. Let $\mathscr{W}^{AP}$ be the dispatch waves used by an *a priori* plan solving (4.14). At any state s, MPF keeps a feasible plan $\pi : (s, \pi) \in \mathscr{S}'$ and π will only dispatch vehicles at waves in $\mathscr{W}^{AP}$. The plan π is initialized in Algorithm 13 as the initial *a priori* plan, skipping all customer visits $I \setminus I^0$ not yet realized by $t = T$.

Algorithm 13 Initial dispatch plan for *MPF*

1: **procedure** INIPLAN(*MPF*,I^0)
2: Solve the *a priori* problem in 4.14 and save its plan π^{AP}.
3: Build a plan π by skipping all customer visits to $I \setminus I^0$ from all routes in π^{AP}.
4: **return** π,

When a request arrives at state $(t, \mathbf{a}, w)$ and node i with release wave b the plan is updated by

solving

$$\min_{\{\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{v},\mathbf{s}\}\in\mathscr{D}(\{1,\ldots,w\})} \beta_i\{z_i+\sum_{k=\max(b+1,h(0,i))}^{w} y_i^k\}+\sum_{k=1}^{w}\sum_{e\in E_k}\gamma d_e x_e^k \tag{4.15a}$$

$$\text{s.t.} \sum_{k=h(0,i)}^{\min\{a_i,w\}} y_i^k = 1, \qquad \forall i\in I_a, \tag{4.15b}$$

$$v_k^w = 0, \qquad \forall w,k\in\mathscr{W}: k<w, w\notin\mathscr{W}^{AP}. \tag{4.15c}$$

The problem is similar to 4.12, but adds a new set of constraints 4.15c to ban dispatch waves not belonging to the set $\mathscr{W}^{AP}$. Function **ArrivalUpdate** is implemented in Algorithm 14 and function **DispatchUpdate** leaves the plan unaltered.

Algorithm 14 Dispatch plan update upon arrival for *MPF*

1: **procedure** ARRIVALUPDATE(MPF,π,s,i,b)
2: Solve problem (4.15) with initial feasible solution π.
3: Get an optimal solution π^*, set $\pi\leftarrow\pi^*$, **return** π

We believe that a fixed dispatch policy such as MPF emulates and improves a system that practitioners may use; it builds a reasonable initial dispatch structure based on probabilistic information and, once the daily operation starts, the decision maker assign requests to dispatch time slots myopically.

4.4.4 Full rollout of the *a priori* policy

A better but more involved idea is to fully roll out the *a priori* policy (RP) and re-optimize the *a priori* problem at each time t, after new information arrives. In a myopic policy, useful new information is only disclosed each time an order arrives; conversely, for RP useful new information may be disclosed continuously over time, depending on expected future arrivals. Thus, to improve the policy's performance we re-optimize the plan when an order arrives, and before each dispatch decision.

The initial dispatch plan π for RP matches the *a priori* plan $\textbf{IniPlan}(RP, V^0) := \textbf{IniPlan}(AP, V^0)$. Define the expected number of requests realized after time $t \in \mathscr{T}$ at node $i \in I$ that are ready by wave $w \in \mathscr{W}$ as

$$f_{i,w}(t) = \begin{cases} \mathbb{E}(N_i(t_w + p) - N_i(t)) & \text{if } t > t_w + p, \\ 0 & \text{else;} \end{cases} \tag{4.16}$$

for a Poisson process with rate λ_i and cut-off time t^{ct}, this becomes

$$f_{i,w}(t) = \lambda_i \max(0, t - \max(t^{ct}, t_w + p)). \tag{4.17}$$

The function **ArrivalUpdate** is implemented in Algorithm 15 and sets π equal to the optimal *a priori* plan for state s in (4.18), conditioned on an order realization at node i,

$$\min_{\{\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{v},\mathbf{s}\} \in \mathscr{D}(\{1,\ldots,w\})} \beta_i \{z_i + \sum_{k=\max(b+1,h(0,i))}^{k} y_i^k\} + \sum_{k=1}^{w} \sum_{e \in E_k} \gamma d_e x_e^k$$
$$+ \sum_{j \in I} \beta_j \left\{ \mathbb{E}(N_j(0) - N_j(t)) z_j + \sum_{k=h(0,j)}^{w} \left(\mathbb{E}(N_j(0) - N_j(t)) - f_{j,k}(t) \right) y_j^k \right\} \tag{4.18a}$$
$$\text{s.t.} \sum_{k=h(j,0)}^{\min\{a_j, w\}} y_j^k = 1, \qquad \forall j \in I_{\mathbf{a}}, \tag{4.18b}$$

where the problem's domain equals (4.12), but it incorporates penalties for expected rejections of future order arrivals in the objective (4.18a). We save some computational time in line 2 of Algorithm 15 by skipping the re-optimization of the plan π for all orders already covered by the previous plan; preliminary computational experiments showed that this filter had no significant impact. Finally, function **DispatchUpdate**, implemented in Algorithm 16, updates the plan before

Algorithm 15 Update of plan upon an order arrival for *RP*

1: **procedure** ARRIVALUPDATE(RP, π, s, i, b)
2: **if** (π does not cover i after wave b) **then**
3: Solve the IP (4.18) and redefine π as its optimal solution
4: **return** π

a dispatch decision at wave w in state $s = (t_w, \mathbf{a}, w)$, solving

$$\min_{\{\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{v},\mathbf{s}\}\in\mathscr{D}(\{1,\ldots,w'\})} \sum_{k=1}^{w}\sum_{e\in E_k} \gamma d_e x_e^k$$

$$+\sum_{j\in I}\beta_i\left\{\mathbb{E}(N_j(0)-N_j(t_w))z_j+\sum_{k=h(0,i)}^{w}\left(\mathbb{E}(N_j(0)-N_j(t_w))-f_{j,k}(t_w)\right)y_j^k\right\} \tag{4.19a}$$

$$\text{s.t.} \sum_{k=h(j,0)}^{\min\{a_j,w\}} y_j^k = 1, \qquad \forall j \in I_{\mathbf{a}}. \tag{4.19b}$$

Algorithm 16 Update of dispatch plan upon a dispatch decision for RP

1: **procedure** DISPATCHUPDATE(RP, π, s)
2: Solve the IP (4.19) and redefine π as its optimal solution
3: **return** π

Computing RP may require the solution of an IP for each request arrived and each dispatch wave in the horizon, and these IP's will grow in difficulty as I, W and arrival frequency per node grow. We implement some procedures to speed up the solution of an IP at a given state s. First, we warm-start the incumbent solution of an IP with the latest feasible plan available from previous plan re-optimizations. Also, we keep all subtour elimination cuts from previous IPs sharing the same network structure. Finally, we do not solve each problem to optimality and set up an optimality tolerance (0.5%) and maximum solution time (1800 seconds). Particularly, the computational effort in **ArrivalUpdate** may be critical; we evaluate policy RP via computational experiments in section 4.6 and design an alternative heuristic rollout policy in 4.5.1 to improve computation times.

4.5 A generic heuristic

Now, we propose a fast meta-heuristic procedure to improve any plan π feasible for the generic IP

$$GC(w,g) = \min_{\{\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{v},\mathbf{s}\}\in\mathscr{D}(\{1,\ldots,w\})} \sum_{k=1}^{w}\sum_{e\in E_k} \gamma d_e x_e^k + \sum_{i\in I}\left\{ g_{i,0}z_i + \sum_{k=h(0,i)}^{w} g_{i,k}y_i^k \right\}, \tag{4.20}$$

where w is the earliest wave and $g_{i,k} \in \mathbb{R}_+$ represents a generalized cost for serving node i at wave k; case $k = 0$ represents no service. All IPs defined in Sections 4.3 and 4.4 can be stated in this form; if the original IP has constraints that enforce previous visit commitments to node i of the form $\sum_{k=h(0,i)}^{b} y_i^k = 1$, then we set $g_{i,k} = M, k = b+1,\ldots,w$ for an M value such that $GC(w,g) > M$ implies infeasibility.

The heuristic complements the IP solver and is used in two phases of the optimization process. First, we run it over any feasible plan π identified during the branch-and-bound tree search, and update the incumbent if the heuristic produces a plan with lower cost. Second, we run it over any initial feasible plan π and use the heuristically improved plan as an initial feasible solution. Also, our heuristic can be used as a solver-independent procedure. We use a local search procedure similar to [43] that exploits the wave structure of any feasible plan by running multiple neighborhood searches over it that solve instances of the prize-collecting Traveling Salesman Problem (PC-TSP). We extend this procedure by adding two improvements; we randomly destroy the local solutions to avoid local optimal points and use randomized acceptance rules to evaluate a candidate solution; see Appendix 4.8.2. Second, we implement a sub-heuristic to solve each PC-TSP to remove the dependence on a MIP solver; this sub-heuristic is defined in Appendix 4.8.3.

Algorithm 17 provides a high-level description of the heuristic; it requires an initial plan π^0 feasible for (4.5), and uses three separate neighborhood searches to improve upon the best available plan π^*: (1) intra-route local search (**IntraLS**), *i*.e., single route node selection and re-sequencing; (2) inter-route local search (**InterLS**), *i*.e., node exchanges between routes and re-sequencing; and

Algorithm 17 Heuristic Search Procedure

1: **procedure** RUNHEURISTIC(Initial feasible plan π^0, maximum random destructions k^{max})
2: $\pi \leftarrow Copy(\pi^0)$, $\pi^* \leftarrow Copy(\pi^0)$, $k \leftarrow 0$
3: **do**
4: **if** ($\neg$INTRALS(r, r^*) **and** $\neg$INTERLS(r, r^*) **and** $\neg$WAVESLS(r, r^*)) **then**
5: RANDOMDESTRUCTION(r)
6: $k \leftarrow k+1$
7: **while** $(k < k^{max})$
8: **return** r^*

(3) wave local search (**WavesLS**), *i*.e., changes in the number of routes and dispatch times. Each local search returns $true$ if it has updated and accepted a new local solution π, and else, it returns $false$. If no local search update is made, the procedure calls a solution destruction operator that randomly deletes percentage of the nodes and routes in plan π; this is done k^{max} times before the heuristic outputs plan π^*. The details of all local search functions and destruction operator can be found in Appendices 4.8.1 and 4.8.2.

We make a final improvement to the meta-heuristic by running a very large neighborhood search that makes two local search moves in series before evaluating a candidate plan's cost; we implement it by calling function *RunHeuristic* recursively. Specifically, we run function *RunHeuristic* a second time over each candidate plan π' that does not improve the local solution π after one local search move $\pi \rightarrow \pi'$, but has a small enough cost ($c_{\pi'} \leq (1+\delta)c_\pi$) to be a good candidate to start a subsequent heuristic procedure.

4.5.1 Heuristic acceptance rollout policy

Because a full rollout of the *a priori* policy may be computationally expensive, we propose a dynamic policy that re-optimizes the *a priori* problem before each dispatch decision, but which heuristically solves (4.18) upon order arrivals to adjust π before acceptance decisions; we refer to this policy as the Heuristic Acceptance Rollout Policy (HARP).

The initial dispatch plan (**IniPlan**) and the update before dispatch decisions (**DispatchUpdate**)

match RP's functions, but the update upon arrivals (**ArrivalUpdate**) differs from RP and calls Algorithm 17 instead of an IP solver to update the plan; it is implemented in Algorithm 18.

Algorithm 18 Update of dispatch plan upon an order arrival for *HARP*

1: **procedure** ARRIVALUPDATE($HARP, \pi, s, i, b$)
2: **if** (π does not cover i after wave b) **then**
3: Heuristically solve problem (4.18): $\pi' \leftarrow$RUNHEURISTIC(π)
4: **if** ($\pi' \neq \pi$) **then** set $\pi \leftarrow \pi'$.
5: **return** r

4.6 Computational Experiments

Now we present a series of computational experiments designed over randomly generated instances with the objective of testing and comparing the quality of our heuristic policies over multiple performance indicators, and performing sensitivity analysis over different instance parameters. Table 4.1 summarizes all policies computed for each instance. Besides all previously discussed policies, we added two infeasible solutions to the DDWP-IA operation used as benchmarks. The policy FLEX is a flexible rollout of the *a priori* policy that relaxes immediate acceptance and postpones it to the end of the day, following the model from [43]; unlike DDWP-IA, it only rejects orders left unserved at the end of the day. We expect FLEX to perform better than the feasible policies, given its less constrained feasible set of actions and similar methods employed to update the plan. Finally, LB corresponds to the perfect information relaxation that executes the optimal plan to each realization of the random variables and is a proven lower bound as described in section 4.3. All policies were programmed in Java and simulated running one thread of a Xeon E5620 processor with up to 12Gb RAM, and using CPLEX 12.6 when necessary as a IP solver.

Table 4.1: Policies computed in experiments

symbol	strategy	section
MP	Myopic re-optimization policy (calibrated $d^{max} = \frac{2T}{7}$)	4.4.1
AP	*A priori* policy + request skipping upon dispatch	4.4.2
MPF	Myopic re-optimization policy with *a priori* dispatches	4.4.3
RP	Rollout of the *a priori* policy	4.4.4
HARP	Heuristic acceptance with rollout of the *a priori* policy	4.5.1
FLEX	Flexible RP relaxing immediate acceptance *a priori* policy	See [43]
LB	Perfect Information Relaxation (deterministic solution)	4.3

4.6.1 Design of data sets for base experiment

We generated 135 data sets to evaluate our policies over different performance indicators. Each data set has a specific geography setting of 50 customer nodes $I = \{1,\dots,50\}$, a subset $I^0 \subset I$ of previously accepted commitments, and a vector of online arrival rates $\lambda \in \mathbb{R}^{|I|}$ determining the probabilities of $|I|$ independent Poisson arrival processes, one for each node $i \in I$.

The geography for each data set is defined by a random seed $g \in \{0,\dots,4\}$ used to assign 50 different locations over a square region of side 50 units following a discrete uniform distribution $\{0,\dots,50\}$ for each component of the location's coordinate; the depot is located at coordinate $(25,25)$. We ruled out repeated coordinates to have a more interesting geography. Travel times between locations are computed as the ℓ_1-norm between two locations' coordinates, in part to best model urban travel times; we assume all proportionality constants between edge travel time, cost and distance equal to 1. All data sets share a common continuous time horizon with $T = 882$ units, a service time at nodes equal to $u_i = 6$, a depot setup time $u_0 = 20$, and $W = 7$ possible dispatch waves homogeneously distributed over time with $t_W = 882$, $t_0 = 0$ and such that $t_w - t_{w-1} = 126$ for each $w \in \mathcal{W}$. Also, any node can be visited in a single dispatch wave since the duration of any direct dispatch round-trip is less than or equal to 126 units. The cut-off time is set at $2/7$ of the horizon, *i*.e., $t^{ct} = t_2 = 252$, and all order processing times are equal to $p_i = 20$ units. We use a penalty cost of the form $\beta_i = 2d_{0,i} + 1$ that makes any round-trip covering a single request

profitable when vehicle time is abundant.

For each geography setting g we set three probabilities $p_{start} = \{0\%, 15\%, 30\%\}$ to have an open request at each node $i \in I$ at time $t = T$ and simulated the sets I^0 three times for each value p_{start} using random seeds $s = 0, 1, 2$. Each data set also has a setting of $\lambda^* \in \{0.5, 1, 2\}$, the average number of realized online requests per node-day, to simulate different levels of arrival intensity. The arrival rate λ_i defining the Poisson process at node $i \in I$ is randomly generated in clusters of 10 nodes so that $10\lambda^* = (T - t^{ct}) \sum_{i=10(k-1)}^{10k} \lambda_i$ for each $k = 1, 2, 3, 4, 5$. We form clusters to create 540 instances with 4 different network sizes, each one made with the first $n = 20, 30, 40, 50$ nodes of a data set.

In addition, we created $M = 50$ arrival realizations for each data set using the probabilistic model above. These scenarios are used as a common sample to estimate the expected cost of all policies. Each instance is defined by a tuple $(g, p_{start}, s, \lambda^*) : g \in \{0, 1, 2, 3, 4\}, p_{start} \in \{0\%, 15\%, 30\%\}, s \in \{0, 1, 2\}, \lambda^* \in \{0.5, 1, 2\}$.

The following metrics are computed for each policy and realization, and then averaged for each instance:

- $cost/request$: the total cost of the realization divided by the total number of requests realized, including orders carried from previous days.
- fill rate (fr): the percentage of requests accepted by the vehicle over all realized requests,
- $distance/order$: the policy routes' distance traveled over the total number of orders accepted,
- gap^P: the percentage increase of the policy's cost over $P \in \{LB, FLEX, RP\}$, respectively,
- $nRoutes$: number of vehicle dispatches,
- $nWaves$: average dispatch length in waves used by the routes dispatched over the realization,
- $iWait$ and $pWait$: number of waves spent waiting at the depot before/after the initial dispatch,

- $afterCT$: percentage of orders covered that are dispatched after the cut-off time,
- $nodes/route$: average number of node visits per route,
- $time_{off}$: average off-line solution time, *i*.e., a call of **IniPlan** in Algorithm 10,
- $time_{dis}$: average solution time before dispatch decisions, *i*.e., a call of **DispatchUpdate** in Algorithm 10,
- $time_{acc}$: average solution time before acceptance decisions, *i*.e., a call of **ArrivalUpdate** in Algorithm 10.

4.6.2 Base experiments

Table 4.2 presents average results for each heuristic policy over all 540 instances. On average, AP and MP have costs 48.0% and 35.0% over the deterministic bound (LB) and may be explained due to a loss of 13.3% and 7.9% in the percentage of orders accepted. Nevertheless, both policies have totally different behavior. MP takes advantage of re-optimization capabilities to produce efficient routes, keeping the distance per order-served low. However, its myopic behavior does not generate enough vehicle returns for recourse possibilities, producing 27.9% fewer routes, longer route duration, and 27.8% more average nodes per route. Unlike MP, the *a priori* policy (AP) does not consider unexpected request arrivals and becomes inefficient when expected requests do not realize. However, it creates a dispatch plan closer to LB in terms of average number of routes, initial wait at the depot, and average number of waves per route.

This suggests combining both policies; a simple mix is MPF, which marginally reduces the cost per request by fixing vehicle dispatch times according to the *a priori* dispatch structure. A more sophisticated combination that produces better results is RP, which rolls out the optimal *a priori* policy (AP) and redesigns the dispatch plan before each decision epoch. Compared to MPF, RP cuts the average cost per request and percentage gap over LB by 8.6% and 36%, respectively. Its

Table 4.2: Average results averaged for each policy

metric \policy	**LB**	**FLEX**	**MP**	**AP**	**MPF**	**RP**	**HARP**
$cost/request$	11.5	13.3	15.4	17.0	15.2	13.9	14.2
fr	93.4%	88.6%	85.5%	80.1%	85.6%	87.8%	86.9%
$distance/order$	9.1	8.8	8.9	9.4	9.0	8.9	8.9
gap^{LB}	N/A	15.9%	35.0%	48.0%	33.0%	21.0%	23.8%
gap^{FLEX}	N/A	N/A	16.2%	27.1%	14.5%	4.4%	6.8%
gap^{RP}	N/A	-4.2%	11.2%	21.8%	9.7%	N/A	2.3%
$time_{off}$ (sec.)	121.9	529.2	0.00	529.2	529.2	529.2	529.2
$time_{disp}$ (sec.)	0.00	81.8	0.00	0.00	0.00	68.6	80.1
$time_{acc}$ (sec.)	0.00	0.00	2.4	0.00	1.2	18.1	1.1
$nRoutes$	2.69	2.51	1.94	2.48	2.21	2.52	2.52
$iWait$	2.87	3.21	3.43	3.09	3.35	3.20	3.21
$pWait$	0.00	0.00	0.00	0.01	0.04	0.00	0.00
$nWaves$	1.61	1.58	1.86	1.64	1.72	1.59	1.58
$nodes/Route$	7.2	7.4	9.2	6.7	8.3	7.4	7.3
$afterCT$	68.4%	79.0%	73.3%	78.3%	76.3%	78.7%	79.1%

benefits mostly arise from improving order coverage and reducing penalty costs; this is crucial for companies interested in providing the best possible customer service. In Figure 4.5 we show the performance of RP and MPF in terms of order fill rate (fr) and $distance/order$ for all instances having the same network size n; we also include the two benchmarks LB and FLEX. In all cases, fr decreases as n increases and RP improves its request fill rate gain over MPF as n increases. All policies are comparable in route efficiency ($distance/order$), which decreases with the network size (due to consolidation opportunities).

If we compare the dispatch structure of RP against the deterministic LB bound, we observe in 4.2 that it waits more before the first vehicle dispatch at the depot and increases the average percentage of accepted orders dispatched after the cut-off time by 10.3%; it pushes some dispatch decisions later in time to protect the plan against uncertainty. Also, as described in [43], our results suggest that waiting at the depot after the first dispatch ($pWave$) does not occur, and it may be better to keep the vehicle busy serving more orders. Compared to FLEX, RP increases the cost per request by 4.5% and provides an estimate to managers of the operational cost incurred by imposing

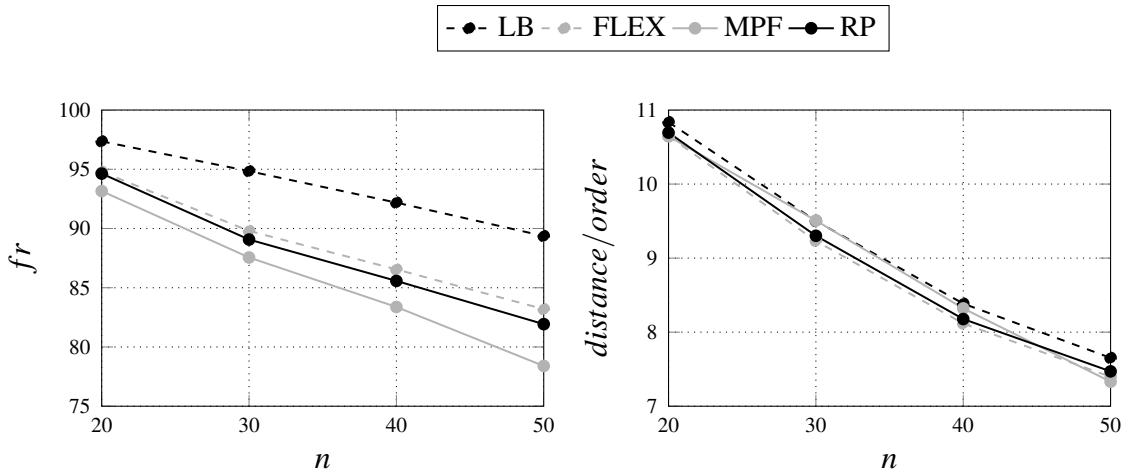


Figure 4.5: Average fill rate (fr) and distance per order for RP policy accepted versus number of nodes (n)

immediate order acceptance in SDD systems.

All policies using the *a priori* solution share offline computation time ($time_{off}$), but differ in online computation per dispatch ($time_{disp}$) and per request acceptance decision ($time_{acc}$); AP and MPF are simple and fast online policies, while RP requires additional computational power. The HARP policy is an alternative to RP that provides 16.5 times faster request acceptance times, making small sacrifices in $cost/request$ (2.2% increase); HARP still outperforms both myopic policies, even when these last two use optimization engines. The average solution times $time_{acc}$ and $time_{disp}$ over all instances aggregated by network size n are displayed in logarithmic scale in Figure 4.6. As expected due to the nature of exact MIP models, computational times increase exponentially with n. In case of $time_{acc}$, HARP removes this exponential growth and keeps this average time under 2 seconds for $n = 50$.

In Figure 4.7 we compare the average cost per request of our best policies, RP and HARP, over all instances sharing parameters of network size n and average arrival intensity per node λ^*. We experimentally observe small economies of scale as n grows for instances with low arrival intensity

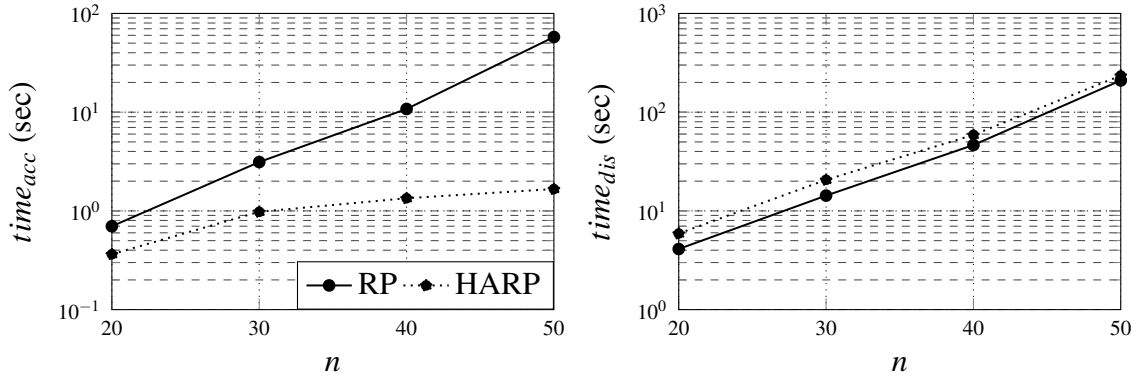


Figure 4.6: Average time per acceptance decision and average time per dispatch decision versus number of nodes (n)

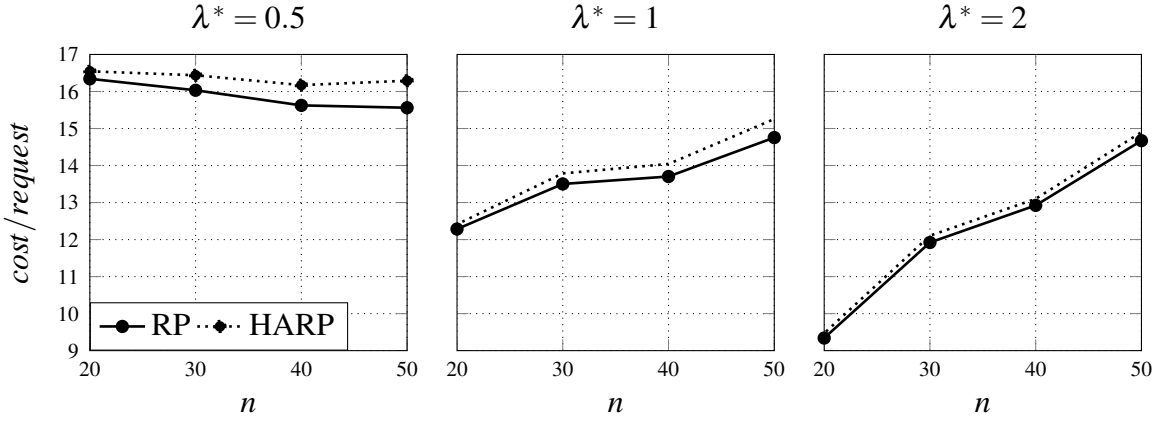


Figure 4.7: Average $cost/request$ versus number of nodes (n) and online arrival intensity (λ^*)

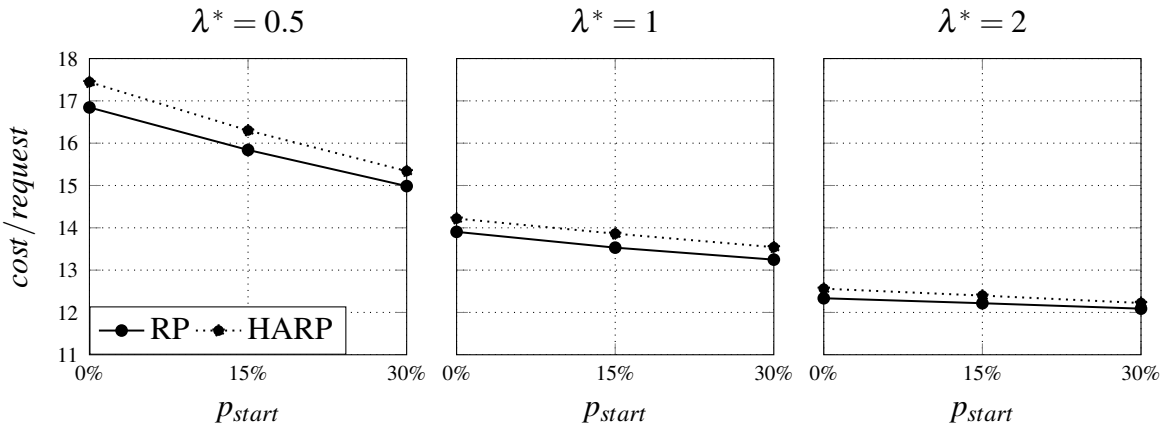


Figure 4.8: Average *cost*/*request* versus offline order probability (p_{start}) and online arrival intensity (λ^*)

($\lambda^* = 0.5$), possibly due to additional vehicle time per request and available order consolidation capacity. Conversely, the cost per order grows with n for moderate and high arrival intensities indicating congestion in the system; the increased request arrival frequency at nodes may reduce the system's remaining capacity and diminish the marginal acceptance of orders as n grows.

Also, our results suggest that the system is more efficient with higher request frequencies per node, since the *cost*/*request* ratio is reduced as λ^* increases for a fixed n, and this reduction marginally increases as n decreases. This implies that an instance with fewer nodes and higher requests per node can be managed at lower cost than an instance with a larger network and lower arrival intensity per node, even when both have the same total number of expected requests per day. As Figure 4.7 shows, instances with 40 nodes and $\lambda^* = 1$ produce an average *cost*/*request* 50% higher than instances with $n = 20$ and $\lambda^* = 2$. We also observe that HARP performs better as n decreases and λ^* increases. The former might be related to the optimization smaller problems, while the latter might be linked with a reduced need for complex solutions; a simpler policy might be good enough for order acceptance decisions when there is high request density.

Figure 4.8 presents the average cost per order of RP and HARP as a function of the probability of having an open order at a node before execution starts (p_{start}) and the average arrival intensity

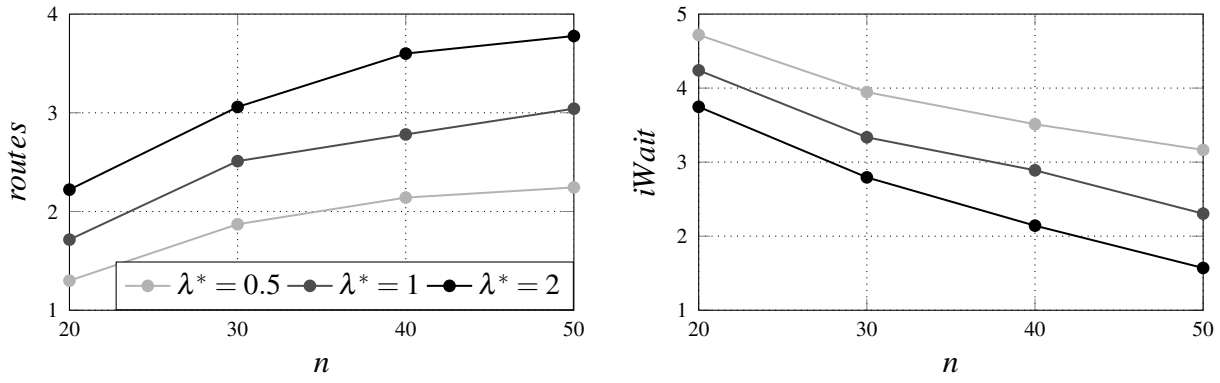


Figure 4.9: Average number of routes and waves waited before dispatch (*iWait*) for RP policy versus number of nodes (n) and online arrival intensity (λ^*)

(λ^*). As expected for each graph, the more information available at the start of the operation, the smaller the cost per request. The cost reduction with p_{start} is particularly high for low arrival intensity ($\lambda^* = 0.5$); in this case, orders carried over from previous days are relatively more important than possible requests realized during the operation; the cost per order tends to stabilize for instances with ($\lambda^* = 2$), where off-line orders are less important because of frequent request arrivals.

Figure 4.9 presents the average number of routes and initial wait at the depot before executing vehicle dispatches (*iWait*) for RP as a function of n and λ^*. For a relatively busier problem (bigger n and λ), our policy reacts by generating more dispatches and waiting less at the depot. It is also interesting to note that instances with congested and smaller networks ($n = 20, \lambda^* = 2$) produce fewer routes and wait more at the depot than instances with scattered and bigger networks ($n = 40, \lambda^* = 1$).

4.6.3 Experiment extension #1: Analysis of performance sensibility over processing time

We now extend the experiments, using all 45 instances with $n = 30$ and $\lambda^* = 1$ to study how sensible the performance of RP is to the order processing time p. We generate 225 new instances

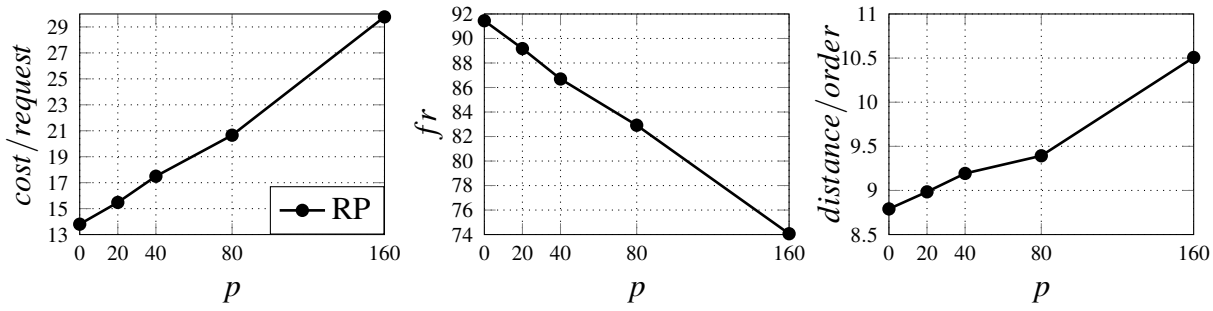


Figure 4.10: Average $cost/request$, fr, and $distance/order$ versus order processing time (p)

by combining these instances with different values of $p \in \{0, 20, 40, 80, 160\}$. We present the average cost per request, fill rate, and distance per accepted order as a function of p in Figure 4.10. Our experiments illustrate the direct impact that order processing times have in the performance of the distribution system. The cost per request grows almost linearly at a rate of 0.1 per unit of processing time. So, a system with $p = 80$ is 33.5% more expensive than one with $p = 20$. It seems that an increase in processing time hinders the system in both aspects: routing efficiency, increasing the distance per order covered because fewer orders are ready for dispatch at each wave; and customer service level, possibly due to a loss in the system's overall acceptance capacity when shifting release times forward in time.

Additionally, Figure 4.11 depicts how the average RP gap over LB and gap difference over the myopic policies reduce as p increases. These results suggest that the cost increase in LB is relatively higher due to a reduction in the feasible space of actions, removing flexibility and complexity even in the relatively simpler deterministic problem.

4.6.4 Experiment extension #2: Analysis of performance sensibility under varying levels of information dynamism

In this section, we study the performance of RP as a function of the order arrival dynamism related to the cut-off time value t^{ct}. We take again all instances with $\lambda^* = 1$, $n = 30$ and combine

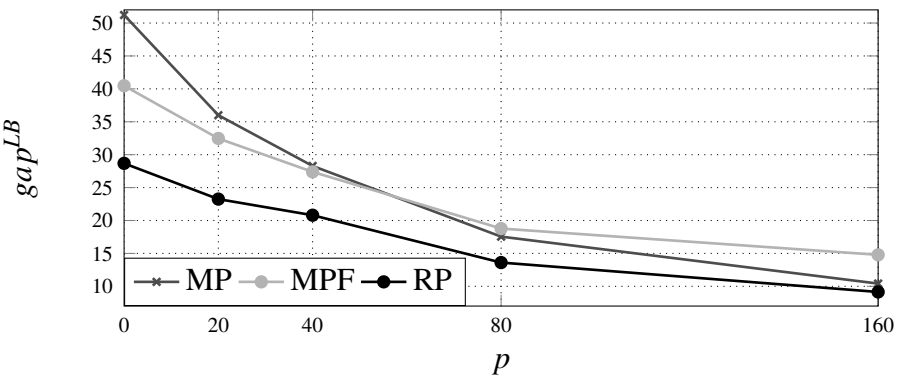


Figure 4.11: Average gap^{LB} versus order processing time (p)

with all values of $t^{ct} \in \{126, 252, 378\}$. To produce comparable instances in terms of number of expected requests, we take all originally simulated order arrivals for $t^{ct} = 252$ and distribute them proportionally between T and the new cut-off time. Formally, an arrival at $x_0 \in [t_0^{ct}, T]$ is relocated to time $x_1 = t_1^{ct} + \frac{T - t_1^{ct}}{T - t_0^{ct}}(x_0 - t_0^{ct}) \in [t_1^{ct}, T]$ when varying the cutoff time from t_0^{ct} to t_1^{ct}. Figure 4.12 presents an example with three requests relocated when t^{ct} changes from 252 to 126 and to 378.

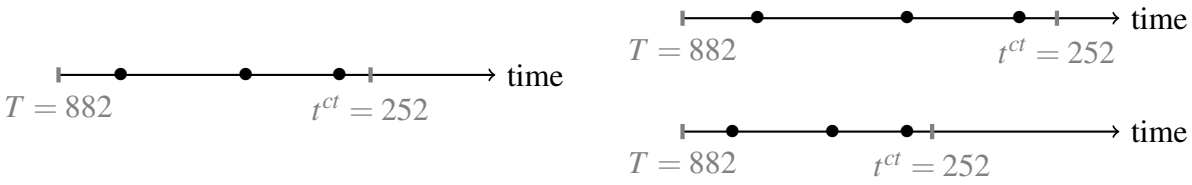


Figure 4.12: Example of request arrivals re-scaled from $t^{ct} = 252$ to $t^{ct} = 126$ and to $t^{ct} = 378$

In Table 4.3 we present average results over all instances as a function of the cut-off time. A higher value of t^{ct} indicates orders arriving relatively closer to the start of the horizon (less dynamism), while a lower value of t^{ct} indicates arrivals more dispersed over time (more dynamism). We first observe that the operation becomes cheaper with information disclosed earlier in time. When t^{ct} is earlier, dispatch decisions are executed having more information, and the percentage of accepted orders dispatched after t^{ct} increases from 65% to 94.8%, improving route efficiency and fr. The opposite effect is observed when dynamism increases and the percentage of orders

dispatched after the cut-off drastically reduces to 21.0%. The average gap over LB decreases with any change from the base setting. The reduction of gap is expected when the cut-off time is earlier, since earlier realized orders render the dynamic policy closer to a deterministic solution where all arrivals are disclosed beforehand; there is equality for the limiting case $t^{ct} = T$. The gap reduction when t^{ct} is later may be related to a reduction in the instance's acceptance capacities, which adds a fixed cost to both the deterministic and the dynamic instance.

Table 4.3: Average performance indicators of RP versus cutoff time (t^{ct})

t^{ct}	$cost/order$	$gap^{LB}(\%)$	$fr(\%)$	$dist/accepted$	$afterCT(\%)$
126	27.4	21.8	76.2	10.1	21.0
252	17.1	32.4	86.8	8.9	65.0
378	9.9	16.0	95.7	7.5	94.8

4.7 Conclusions

We have formulated the DDWP-IA for SDD operations, which integrates immediate request acceptance and processing with order dispatch and delivery. We formulate an integer program to solve the deterministic version of the problem, extending the DDWP's solution methodology. This model is used to design an optimal solution for the stochastic *a priori* problem by converting it into a deterministic equivalent. We also design a fast meta-heuristic capable of replacing a MIP solver with only a minor sacrifice in cost.

We designed a set of computational instances to test our heuristic policies under different settings of geography, problem size, online order arrival intensity, and percentage of accepted orders known before the operation starts. If the *a priori* policy is re-optimized before each decision epoch, the resulting rollout policy (RP) outperforms any of our benchmarks and reduces the system's cost per request by 8.6%.

The success of RP is related to optimization-guided decisions and increasing recourse oppor-

tunities by executing more vehicle dispatches compared to myopic policies. Its marginal benefit is concentrated in improving order acceptance rates rather than in routing efficiency; this may be highly desirable for SDD services. In particular, RP also outperforms the MPF policy that uses an initial optimal *a priori* solution to fix vehicle dispatch times before the operation starts. We also compared RP against an infeasible rollout of the optimal *a priori* policy that can postpone order acceptance decisions throughout the day. The marginal cost per request added by imposing immediate order acceptance is estimated to be 4.4%.

The rollout policy may be computationally expensive per acceptance decision, so we also proposed HARP, a faster variant using meta-heuristics to solve the *a priori* problem before each acceptance decision; it speeds up computations 16.5 times on average, incurring a small increase in cost/request (2.1%).

We also tested our policy against different levels of order processing times at the depot and different order cut-off times. We conclude that a reduction in the order processing times may linearly be transferred to a reduction in cost per request distributed. This suggests the importance of implementing faster warehousing operations for SDD. Our experiments also show that having more dynamism in the order arrival process (a later cut-off time) may significantly increase the system's cost per request; it is fundamental to design an SDD service with an appropriate order cut-off time, delivering the necessary amount of service flexibility to the customer while awarding enough of a time buffer to the operation's planner.

Future research on the DDWP includes the extension of this model to multiple vehicles, so that we can investigate the potential risk pooling effects, *e*.g., [2]. Another challenge is to study the optimal number and allocation of feasible dispatch waves. There are still many open challenges associated with same-day delivery for the logistics research community.

4.8 Appendix of chapter 4

4.8.1 Local Search Neighborhoods

As follows we define three local search (LS) procedures called within our meta-heuristic in Algorithm 17 and based on our heuristic in [43]: IntraLS, InterLS and WavesLS. Each LS procedure explores over different levels of a dispatch plan π structure and searches to improve the best plan available so far π^*.

IntraLS, defined in Algorithm 19, exploits the relation between the DDWP and a PC-TSP

$$PCTSP(d^{max}, Q, \rho) := \min_{S \subseteq Q:\, t^*(S) \leq d^{max}} \left\{ c^*(S) - \sum_{i \in S} \rho_i \right\} \tag{4.21}$$

solved over a subset $Q \subseteq I$ of nodes, prizes $\rho_i, i \in Q$, and a maximum route duration d^{max}. IntraLS is a best move procedure, where a move is described by re-optimizing one route r_w^π dispatched at wave w from the local solution π. Let π' be a copy of the local solution π after removing route r_w^π from it and leaving all remaining routes unaltered. The procedure solves a PC-TSP over the set of nodes in π' left unattended $\bar{I}(\pi') = \{i \in I : i \notin r_k^{\pi'}, \forall k \in \mathscr{W}^{\pi'}\}$, a maximum route duration equal to the duration of the waves left available after removing route r_w^π, and prizes $\rho_i = g_{i,0} - g_{i,w}$ defining penalty savings when visiting node i in a vehicle dispatch at w. The procedure updates the local solution π after each best improvement loop if the best move candidate $\hat{\pi}$ has a lower cost; it also updates the overall best solution π^*. The procedure returns a boolean variable with a $true$ value if the local plan π was improved and returns $false$ if not. Any local solution π processed by IntraLS contains only routes $r_k^\pi, k \in \mathscr{W}^\pi$ that are optimally sequenced and that cannot be improved by selecting a different subset of requests to service from $\bar{I}(\pi) \cup \{r_w^\pi\}$.

InterLS uses best move searches over pairs of routes using neighborhoods inspired by those in [66] for the capacitated vehicle routing problem (CVRP): two-edge exchanges between routes, removal and reinsertion of a k-customer sequence from one route to another, and customer swaps

Algorithm 19 Intra-route LS procedure

procedure INTRALS(plan π, best plan π^*)
  $\mu \leftarrow false$
  loop
    $\hat{\pi} \leftarrow \pi$ //initialize best candidate
    for $w \in \mathscr{W}^{\pi}$ **do**
      Let π' be a copy of π without route r_w^{π}
      Let $d^{max} \leftarrow t_w - t_{q(w,r_w^{\pi})}$
      Solve PCTSP($d^{max}, \bar{I}(r'), \{g_{i,0} - g_{i,w}\}$) and add the optimal route found to π' and dispatch it at wave w
      if $(c_{\pi'} < c_{\hat{\pi}})$ **then** $\hat{\pi} \leftarrow \pi'$ //update best candidate
    if $(c_{\hat{\pi}} < c_{\pi})$ **then**
      $\pi \leftarrow \hat{\pi}$, $\mu \leftarrow true$ //update local solution
      if $(c_{\hat{\pi}} < c_{\pi^*})$ **then** $\pi^* \leftarrow \hat{\pi}$ //update best solution
    else break loop
  return μ

between routes. To implement these ideas, we account for two differences between the CVRP and the DDWP. First, we model the prize-collecting component; a move changes penalty savings (due to the different dispatch time). Second, we check the durations of the new routes to ensure that they remain compatible with the fixed dispatch times of the unchanged routes. Just as IntraLS, this function updates the local solution π, the best solution π^* and returns *true* if the local solution π was updated and *false*, otherwise.

The third neighborhood search is a Waves Local Search (WavesLS), described in Algorithm 20. The search perturbs the dispatch structure $\mathscr{W}^{\pi}$ of a plan π using seven operations: Reorder,Cut,Merge,Insert,Delete,Enlarge, and Reduce. The Reorder operator is defined in [43] and uses a job scheduling approach to re-reassign the routes dispatched in π to the best possible dispatch waves, without altering the customer visit sequences or the route durations. The last six search over new candidate solutions by changing the dispatch structure of π and solving multiple PC-TSPs.

The Cut operator, described in Algorithm 21, searches over dispatch plans that result when

Algorithm 20 Waves Local Search (WavesLS)

1: **procedure** WAVESLS(local plan π, best plan π^*)
2: **loop**
3: **if** ($\neg$REORDER(π, π^*) **and** $\neg$CUT(π, π^*) **and** $\neg$MERGE(π, π^*) **and** $\neg$INSERT(π, π^*) **and** $\neg$DELETE(π, π^*) **and** $\neg$ENLARGE(π, π^*)**and** $\neg$REDUCE(π, π^*)) **then**
4: **break loop**

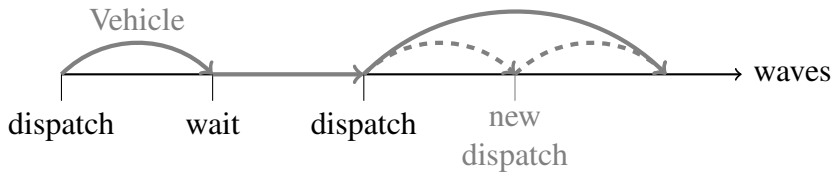


Figure 4.13: Example of a cut operation where a new dispatch plan is created (dashed flow) from an existing one (continuous flow) by adding an extra return to the depot.

splitting a single vehicle route r_w^π with duration $w - q(w, r_w^\pi) \geq 2$ waves into two dispatches with shorter wave duration; this operator adds an extra return trip to the depot, as depicted in Figure 4.13. The Merge operator, described in Algorithm 22, works reversing cut moves and searches over all dispatch profiles that arise when merging two consecutive dispatches into a single longer duration dispatch, as shown in Figure 4.14. The Insert operator, described in Algorithm 23, inserts a new route with duration one wave between two dispatched routes in a plan π shifting all previous dispatches a wave earlier in time; see Figure 4.15. The Delete operator, described in Algorithm 24, searches for a better solution by deleting one dispatch from the plan and pushing all preceding dispatches later in time, as depicted in Figure 4.16. The Enlarge operator, described in Algorithm 25, searches for a better solution by extending the duration of a vehicle dispatch by one wave and dispatching all preceding routes one wave earlier, as depicted in Figure 4.17. Finally, the Reduce operator, described in Algorithm 26, searches for a better solution by reducing a wave the duration

Algorithm 21 Cut operation

procedure CUT(local plan π, best plan so far π^*)
  for $w \in \mathcal{W}^{\pi}$ **do**
    $d^{max} \leftarrow t_w - t_{q(w,r_w^\pi)}$
    for $v : (w-1) \rightarrow (w - q(w, r_w^\pi) + 1)$ **do**
      Let π' a copy of π without route r_w^π
      Solve PCTSP($t_w - t_v, \bar{I}(\pi'), \{g_{i,0} - g_{i,w}\}$) and add optimal route to π' at wave w.
      Solve PCTSP($d^{max} - (t_w - t_v), \bar{I}(\pi'), \{g_{i,0} - g_{i,v}\}$) and add optimal route to π' at wave v
      if ($c_{\pi'} < c_\pi$) **then**
        if ($c_{\pi'} < c_{\pi^*}$) **then** $\pi^* \leftarrow \pi'$
        $\pi \leftarrow \pi'$ **and return** $true$
  return $false$

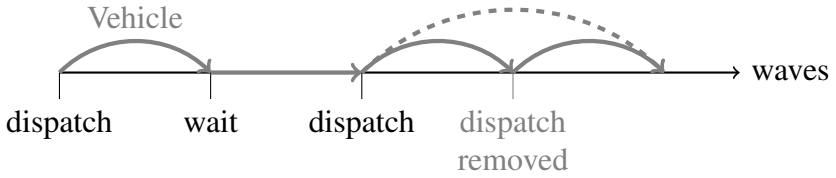


Figure 4.14: Example of a merge operation where a new dispatch plan is created (dashed flow) from an existing one (continuous flow) by removing one return to the depot and merging two dispatches.

Algorithm 22 Merge operation

procedure MERGE(local plan π, best plan π^*)
  for $w \in \mathcal{W}^{\pi}$ such that $q(w, r_w^\pi) > 0$ **do**
    Let $w' \leftarrow q(w, r_w^\pi)$
    Let π' be a copy of π without routes r_w^π and $r_{w'}^\pi$
    Let $d^{max} \leftarrow t_w - t_{q(w', r_{w'}^\pi)}$
    Solve PCTSP($d^{max}, \bar{I}(\pi'), \{g_{i,0} - g_{i,w}\}$) and add optimal route to π' at wave w
    if ($c_{\pi'} < c_\pi$) **then**
      if ($c_{\pi'} < c_{\pi^*}$) **then** $\pi^* \leftarrow \pi'$
      $\pi \leftarrow \pi'$ **and return** $true$
  return $false$

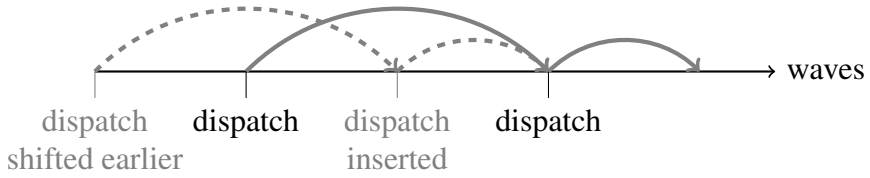


Figure 4.15: Example of an Insert operation where an new dispatch is inserted launching previous routes a wave earlier (dashed flow).

Algorithm 23 Insert operation

```
1: procedure INSERT(local plan π, best plan π*)
2:     if (max{k ∈ 𝒲^π} = W) then return false
3:     for w ∈ 𝒲^π ∪ {0} do
4:         Let π' a copy of π with routes r_k^π, k ∈ 𝒲^π : k > w dispatched one wave earlier.
5:         Solve PCTSP(t_{w+1} − t_w, Ī(π'), {g_{i,0} − g_{i,w+1}}) and add optimal route to π' at wave
   w+1.
6:         if (c_{π'} < c_π) then
7:             if (c_{π'} < c_{π*}) then π* ← π'
8:             π ← π' and return true
9:     return false
```

of one dispatch in the plan and executing all precedent dispatches one wave later, see Figure 4.18.

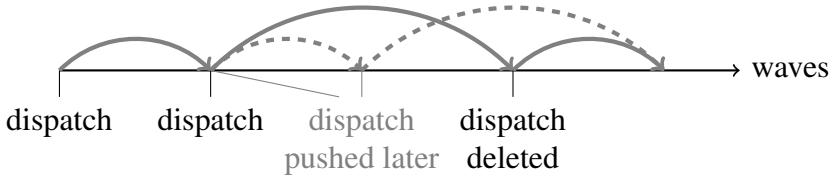


Figure 4.16: Example of a Delete operation where last dispatch is deleted pushing preceding ones later (dashed flow).

Algorithm 24 Delete operation

procedure DELETE(local plan π, best plan π^*)
  for $w \in \mathcal{W}^\pi$ **do**
    Let π' a copy of π with r_w^π deleted and all routes $r_k^\pi, k \in \mathcal{W}^\pi : k > w$ dispatched $w - q(w, r_w^\pi)$ waves forward in time.
    if $(c_{\pi'} < c_\pi)$ **then**
      if $(c_{\pi'} < c_{\pi^*})$ **then** $\pi^* \leftarrow \pi'$
      $\pi \leftarrow \pi'$ **and return** $true$
  return $false$

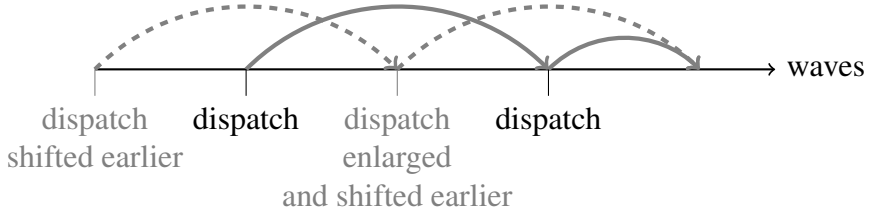


Figure 4.17: Example of an Enlarge operation where the last dispatch is enlarged and launched a wave earlier pushing the previous dispatch earlier too (dashed flow).

Algorithm 25 Enlarge operation

procedure ENLARGE(local plan π, best plan π^*)
  if $(\max\{k \in \mathcal{W}^\pi\} = W)$ **then return** $false$
  for $w \in \mathcal{W}^\pi$ **do**
    Let π' a copy of π without route r_w^π and all routes $r_k^\pi, k \in \mathcal{W}^\pi : k > w$ dispatched one wave earlier.
    Solve PCTSP$(t_{w+1} - t_{q(w,r_w^\pi)}, \bar{I}(\pi'), \{g_{i,0} - g_{i,w+1}\})$ and add optimal route to π' at wave $w+1$.
    if $(c_{\pi'} < c_\pi)$ **then**
      if $(c_{\pi'} < c_{\pi^*})$ **then** $\pi^* \leftarrow \pi'$
      $\pi \leftarrow \pi'$ **and return** $true$
  return $false$

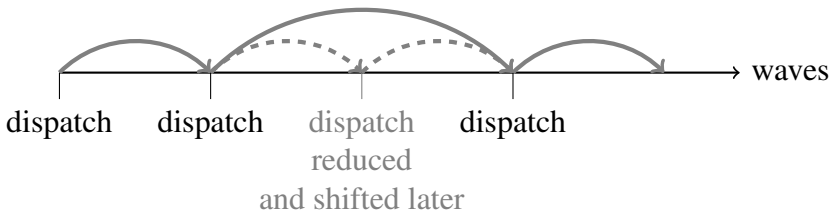


Figure 4.18: Example of a Reduce operation where one dispatch is reduced and launched one wave later pushing the previous dispatch later too (dashed flow).

Algorithm 26 Reduce operation

procedure REDUCE(local plan π, best plan π^*)
  for $w \in \mathscr{W}^{\pi}$ such that $w - q(w, r_w^{\pi}) > 1$ **do**
    Let π' a copy of π without route r_w^{π} and all routes $r_k^{\pi}, k \in \mathscr{W}^{\pi} : k > w$ dispatched one wave later.
    Solve PCTSP($t_{w-1} - t_{q(w,r_w^{\pi})}, \bar{I}(\pi'), \{g_{i,0} - g_{i,w-1}\}$) and add optimal route to π' at wave $w-1$.
    if ($c_{\pi'} < c_{\pi}$) **then**
      if ($c_{\pi'} < c_{\pi^*}$) **then** $\pi^* \leftarrow \pi'$
      $\pi \leftarrow \pi'$ **and return** $true$
  return $false$

4.8.2 Random Destruction

The heuristic presented in Algorithm 17 calls the function **RandomDestruction**, defined in Algorithm 27, that partially destroys a solution π to move the search to distant solutions and avoid local optimality. This function works at two levels of a solution's structure and deletes randomly chosen routes and customers from it.

Algorithm 27 Random destruction procedure for plan π

```
1:  procedure RANDOMDESTRUCTION(local solution π)
2:      Generate a random number x ∈ {1, ..., ⌊0.5|𝒲^π|⌋}
3:      Randomly delete x routes from plan π.
4:      if (π is empty) then return
5:      for (w ∈ 𝒲^π) do
6:          Set y equal to the number of nodes visited in r_w^π.
7:          if (y > 1) then
8:              Generate a random number z ∈ {⌈0.25y⌉, ..., ⌊0.75y⌋}.
9:              Randomly delete and skip z visits from r_w^π
10:     return
```

4.8.3 Heuristic Solution For the prize-collecting TSP

In Algorithm 28 we implement a metaheuristic solution to a PC-TSP over the set of nodes Q, prizes $\rho_i, i \in Q$, and maximum duration d^{max}. This solution implements simulated annealing running over an elementary route $r = \{0, i_1, i_2, .., 0\}$ with objective value v_p with a subset $Q^r_{out} \subseteq Q$ of unattended nodes.

The parameters T_0 and δ control the evolution of simulated annealing and k^{max} determines a maximum number of iterations. The search also executes a partial solution destruction when no improvement is found to avoid local optimality. The neighborhood $\mathcal{N}(r)$ used in line 10 is a compound one consisting of ten polynomially sized neighborhoods, each one based on the following moves:

Algorithm 28 Metaheuristic for the PC-TSP

procedure HPCTSP($d^{max}, Q, \rho, k^{max}$),
  Initialize best route: $r^* \leftarrow \emptyset$,
  for $s = 0$ to $NumSeeds$ **do**
    Set random seed s,
    Generate route r by sequentially inserting random nodes from Q int the last position of r; stop before violating the route duration constraint,
    Update the set of non-visited nodes $Q^{r'}_{out} \leftarrow Q$,
    Initialize temperature $T \leftarrow T_0$ and iteration counter $k \leftarrow 0$.
    while ($k < k^{max}$) **do**
      $update \leftarrow false$
      for (neighbor $r' \in \mathcal{N}(r)$) **do**
        Generate a random number p for a $Uniform(0,1)$ distribution,
        if ($e^{(v_{r'} - v_r)/T} > p$) **then**
          $r \leftarrow r'$, update Q^r_{out}, $update \leftarrow true$, and **break for.**
      if ($\neg update$) **then** Randomly skip and remove 30% of nodes from r. Update Q_{out}.
      $k \leftarrow k + 1$,
      $T \leftarrow T \times \varepsilon$.
      if ($v_r > v_{r^*}$) **then**
        $r^* \leftarrow r$,
        reset search $T \leftarrow T_0, k \leftarrow 0$.
  return r^*

1. a swap between an unvisited node $i \in Q^r_{out}$ and a visited node j in route r ($\mathcal{O}(|Q|^2)$ moves),
2. an insertion of an unvisited node $i \in Q^r_{out}$ after a visited node j in route r ($\mathcal{O}(|Q|^2)$ moves),
3. a removal of a visited node j from route r ($\mathcal{O}(|Q|)$ moves),
4. a removal of a visited node k from route r and the insertion of an unvisited node $i \in Q^r_{out}$ after a visited node j in route r ($\mathcal{O}(|Q|^3)$ moves),
5. 2-opt, *i*.e., 2-edge exchanges within route r ($\mathcal{O}(|Q|^2)$ moves),
6. all possible removal of a series of k nodes in r starting with node i and its reinsertion after node j in r () after $r_i, .., r_{i+k}$ after r_j ($\mathcal{O}(|Q|^3)$ moves),
7. a internal swap in route r between two visited nodes ($\mathcal{O}(|Q|^2)$ moves),
8. a swap between one visited node i in route r an two nodes $j,k \in Q_{out}$ inserted in series ($\mathcal{O}(|Q|^3)$ moves),
9. a swap between two consecutive visited nodes i, j in route r an one node $k \in Q_{out}$ ($\mathcal{O}(|Q|^2)$ moves),
10. a swap between two consecutive visited nodes i, j in route r and two nodes $k, q \in Q_{out}$ inserted in series ($\mathcal{O}(|Q|^3)$ moves).

All moves resulting in violations of the maximum duration limit are discarded.

REFERENCES

[1] D. Adelman, "A Price-Directed Approach to Stochastic Inventory/Routing," *Operations Research*, vol. 52, pp. 499–514, 2004.

[2] A. Ak and A. Erera, "A paired-vehicle recourse strategy for the vehicle-routing problem with stochastic demands," *Transportation science*, vol. 41, no. 2, pp. 222–237, 2007.

[3] M. Albareda-Sambola, E. Fernández, and G. Laporte, "The dynamic multiperiod vehicle routing problem with probabilistic information," *Computers & Operations Research*, vol. 48, no. 0, pp. 31–39, 2014.

[4] Amazon.com, "Annual report form (10-k)," 2015.

[5] E. Angelelli, N. Bianchessi, R. Mansini, and M. G. Speranza, "Short term strategies for a dynamic multi-period routing problem," *Transportation Research Part C: Emerging Technologies*, vol. 17, no. 2, pp. 106–119, 2009.

[6] E. Angelelli, M. W. P. Savelsbergh, and M. G. Speranza, "Competitive analysis of a dispatch policy for a dynamic multi-period routing problem," *Operations Research Letters*, vol. 35, no. 6, pp. 713–721, 2007.

[7] E. Angelelli, M. G. Speranza, and M. W. P. Savelsbergh, "Competitive analysis for dynamic multiperiod uncapacitated routing problems," *Networks*, vol. 49, no. 4, pp. 308–317, 2007.

[8] D. Applegate, R. Bixby, V. Chvátal, and W. Cook, *The Traveling Salesman Problem: A Computational Study*. Princeton, New Jersey: Princeton University Press, 2006.

[9] C. Archetti, D. Feillet, and M. Speranza, "Complexity of routing problems with release dates," *European Journal of Operational Research*, vol. 247, no. 3, pp. 797 –803, 2015.

[10] N. Azi, M. Gendreau, and J.-Y. Potvin, "A dynamic vehicle routing problem with multiple delivery routes," *Annals of Operations Research*, vol. 199, no. 1, pp. 103–112, 2012.

[11] ——, "An adaptive large neighborhood search for a vehicle routing problem with multiple routes," *Computers & Operations Research*, vol. 41, pp. 167 –173, 2014.

[12] ——, "An exact algorithm for a vehicle routing problem with time windows and multiple use of vehicles," *European Journal of Operational Research*, vol. 202, no. 3, pp. 756 –763, 2010.

[13] E. Balas, "The prize collecting traveling salesman problem," *Networks*, vol. 19, no. 6, pp. 621–636, 1989.

[14] R. Bent and P. van Hentenryck, "Scenario-based planning for partially dynamic vehicle routing with stochastic customers," *Operations Research*, vol. 52, no. 6, pp. 977–987, 2004.

[15] D. Bertsekas, *Dynamic programming and optimal control*. Athena Scientific Belmont, MA, 1995, vol. 1.

[16] D. Bertsekas and D. Castañón, "Rollout algorithms for stochastic scheduling problems," *Journal of Heuristics*, vol. 5, no. 1, pp. 89–108, 1999.

[17] D. Bertsimas, "A vehicle routing problem with stochastic demand," *Operations Research*, vol. 40, no. 3, pp. 574–585, 1992.

[18] L. Bianchi and A. Campbell, "Extension of the 2-p-opt and 1-shift algorithms to the heterogeneous probabilistic traveling salesman problem," *European Journal of Operational Research*, vol. 176, no. 1, pp. 131 –144, 2007.

[19] D. Brown, J. Smith, and P. Sun, "Information relaxations and duality in stochastic dynamic programs," *Operations research*, vol. 58, pp. 785–801, 2010.

[20] Y. Bukchin, E. Khmelnitsky, and P. Yakuel, "Optimizing a dynamic order-picking process," *European Journal of Operational Research*, vol. 219, no. 2, pp. 335–346, 2012.

[21] A. M. Campbell and M. W. P. Savelsbergh, "Decision support for consumer direct grocery initiatives," *Transportation Science*, vol. 39, no. 3, pp. 313–327, 2005.

[22] A. Campbell and B. W. Thomas, "Challenges and advances in a priori routing," in *The Vehicle Routing Problem: Latest Advances and New Challenges*, Springer, 2008, pp. 123–142.

[23] ——, "Probabilistic traveling salesman problem with deadlines," *Transportation Science*, vol. 42, no. 1, pp. 1–21, 2008.

[24] D. Cattaruzza, N. Absi, and D Feillet, "The multi-trip vehicle routing problem with time windows and release dates," *Transportation Science*, vol. 50, no. 2, pp. 676–693, 2016.

[25] D. Cattaruzza, N. Absi, D. Feillet, and J. González-Feliu, "Vehicle routing problems for city logistics," *EURO Journal on Transportation and Logistics*, pp. 1–29, 2015.

[26] E. Çeven and K. Gue, "Optimal wave release times for order fulfillment systems with deadlines," *Transportation Science*, vol. 1, no. 1, pp. 1–15, 2015.

[27] T. Cheong and C. White, "Dynamic traveling salesman problem: value of real-time traffic information," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 2, 2012.

[28] J. F. Cordeau, G. Laporte, M. W. P. Savelsbergh, and D. Vigo, "Vehicle routing," *Transportation, handbooks in operations research and management science*, vol. 14, pp. 367–428, 2006.

[29] R. DeNale and D. Weidenhamer, "U.s. census bureau news: quaterly retail e-commerce sales - second quarter 2016," *U.S. Department of Commerce*, Nov. 2015.

[30] V. Desai, V. Farias, and C. Moallemi, "Bounds for Markov decision processes," *Reinforcement Learning and Approximate Dynamic Programming for Feedback Control*, pp. 452–473, 2011.

[31] A. Erera, M. W. P. Savelsbergh, and E. Uyar, "Fixed routes with backup vehicles for stochastic vehicle routing problems with time constraints," *Networks*, vol. 54, no. 4, pp. 270–283, 2009.

[32] D. de Farias and B. van Roy, "The linear programming approach to approximate dynamic programming," *Operations Research*, vol. 51, no. 6, pp. 850–865, 2003.

[33] N. Gademann and S. Velde, "Order batching to minimize total travel time in a parallel-aisle warehouse," *IIE transactions*, vol. 37, no. 1, pp. 63–75, 2005.

[34] M. Gendreau, G. Laporte, and R. Séguin, "Stochastic vehicle routing," *European Journal of Operational Research*, vol. 88, no. 1, pp. 3–12, 1996.

[35] G. Godfrey and W. Powell, "An adaptive dynamic programming algorithm for dynamic fleet management, i: single period travel times," *Transportation Science*, vol. 36, no. 1, pp. 21–39, 2002.

[36] B. Golden, S. Raghavan, and E. Wasil, Eds., *The Vehicle Routing Problem: Latest Advances and New Challenges*. Springer, 2008.

[37] J. Goodson, J. Ohlmann, and B. W. Thomas, "Rollout policies for dynamic solutions to the multivehicle routing problem with stochastic demand and duration limits," *Operations Research*, vol. 61, no. 1, pp. 138–154, 2013.

[38] J. C. Goodson, B. W. Thomas, and J. W. Ohlmann, "A rollout algorithm framework for heuristic solutions to finite-horizon stochastic dynamic programs," *To appear in European Journal of Operational Research*, 2016.

[39] G. Gutin and A. Punnen, Eds., *The Traveling Salesman Problem and Its Variations*. Dordrecht, The Netherlands: Kluwer Academic Publishers, 2002.

[40] P. Jaillet, "A priori solution of a traveling salesman problem in which a random subset of the customers are visited," *Operations Research*, vol. 36, no. 6, pp. 929–936, 1988.

[41] P. Jaillet and M. Wagner, "Online vehicle routing problems: a survey," in, B. Golden, S. Raghavan, and E. Wasil, Eds., Springer, 2008, pp. 221–237.

[42] A. Kenyon and D. Morton, "Stochastic vehicle routing with random travel times," *Transportation Science*, vol. 37, no. 1, pp. 69–82, 2003.

[43] M. Klapp, A. Erera, and A Toriello, "The dynamic dispatch waves problem for same-day delivery," *under review*, 2016.

[44] ——, "The one-dimensional dynamic dispatch waves problem," *Transportation Science*, pp. 1–14, 2016.

[45] G. Laporte, F. Louveaux, and H. Mercure, "A priori optimization of the probabilistic traveling salesman problem," *Operations Research*, vol. 42, no. 3, pp. 543–549, 1994.

[46] ——, "The vehicle routing problem with stochastic travel times," *Transportation science*, vol. 26, no. 3, pp. 161–170, 1992.

[47] A. Larsen, O. Madsen, and M. Solomon, "Recent developments in dynamic vehicle routing systems," in *The Vehicle Routing Problem: Latest Advances and New Challenges*, Springer, 2008, pp. 199–218.

[48] C. Lee, K. Lee, and S. Park, "Robust vehicle routing problem with deadlines and travel time/demand uncertainty," *Journal of the Operational Research Society*, vol. 63, no. 9, pp. 1294–1306, 2011.

[49] T. Leipälä, "On the solutions of stochastic traveling salesman problems," *European Journal of Operational Research*, vol. 2, no. 4, pp. 291–297, 1978.

[50] X. Li, P. Tian, and S. Leung, "Vehicle routing problems with time windows and stochastic travel and service times: models and algorithm," *International Journal of Production Economics*, vol. 125, no. 1, pp. 137–145, 2010.

[51] T. Ma and P. Zhao, "A review of algorithms for order batching problem in distribution center," in *International Conference on Logistics Engineering, Management and Computer Science (LEMCS 2014)*, Atlantis Press, 2014.

[52] C. Novoa and R. Storer, "An approximate dynamic programming approach for the vehicle routing problem with stochastic demands," *European Journal of Operational Research*, vol. 196, no. 2, pp. 509–515, 2009.

[53] V. Pillac, M. Gendreau, C. Guéret, and A. Medaglia, "A review of dynamic vehicle routing problems," *European Journal of Operational Research*, vol. 225, no. 1, pp. 1–11, 2013.

[54] M. Pinedo, *Scheduling: theory, algorithms, and systems*. Springer Science & Business Media, 2012.

[55] W. Powell, *Approximate Dynamic Programming: Solving the curses of dimensionality*. John Wiley & Sons, 2007.

[56] H. Psaraftis, "Dynamic vehicle routing: status and prospects," *Annals of Operations Research*, vol. 61, no. 1, pp. 143–164, 1995.

[57] M. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2009.

[58] M. W. P. Savelsbergh and S. M., "The general pickup and delivery problem," *Transportation Science*, vol. 29, no. 1, pp. 17–29, 1995.

[59] M. W. P. Savelsbergh and T. Van Woensel, "50th anniversary invited article - city logistics: challenges and opportunities," *Transportation Science*, vol. 50, no. 2, pp. 579–590, 2016.

[60] P. Schweitzer and A. Seidmann, "Generalized polynomial approximations in markovian decision processes," *Journal of mathematical analysis and applications*, vol. 110, no. 2, pp. 568–582, 1985.

[61] N. Secomandi and F. Margot, "Reoptimization approaches for the vehicle-routing problem with stochastic demands," *Operations Research*, vol. 57, no. 1, pp. 214–230, 2009.

[62] H. Tang and E. Miller-Hooks, "Approximate procedures for probabilistic traveling salesperson problem," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 1882, pp. 27–36, 2004.

[63] D. Taş, N. Dellaert, T. van Woensel, and T. de Kok, "Vehicle routing problem with stochastic travel times including soft time windows and service costs," *Computers & Operations Research*, 2012.

[64] B. W. Thomas, "Dynamic vehicle routing," *Wiley Encyclopedia of Operations Research and Management Science*, 2010.

[65] A. Toriello, W. Haskell, and M. Poremba, "A Dynamic Traveling Salesman Problem with Stochastic Arc Costs," *Operations Research*, vol. 62, pp. 1107–1125, 2014.

[66] P. Toth and D. Vigo, "The granular tabu search and its application to the vehicle-routing problem," *Informs Journal on computing*, vol. 15, no. 4, pp. 333–346, 2003.

[67] ——, *Vehicle Routing: Problems, Methods, and Applications*. SIAM, 2014, vol. 18.

[68] M. W. Ulmer, D. C. Mattfeld, and N. Soeffker, "Dynamic multi-period vehicle routing: approximate value iteration based on dynamic lookup tables," *working paper*, 2016.

[69] M. W. Ulmer, B. W. Thomas, and D. C. Mattfeld, "Preemptive depot returns for a dynamic same-day delivery problem," *working paper*, 2016.

[70] S. Voccia, A. Campbell, and B. W. Thomas, "The probabilistic traveling salesman problem with time windows," *EURO Journal on Transportation and Logistics*, pp. 1–19, 2012.

[71] ——, "The same-day delivery problem for online purchases," *To appear in Transportation Science*, 2015.

[72] M. Wen, J.-F. Cordeau, G. Laporte, and J. Larsen, "The dynamic multi-period vehicle routing problem," *Computers & Operations Research*, vol. 37, no. 9, pp. 1615–1623, 2010.

VITA

Mathias Alberto Klapp was born in Santiago, Chile, in 1984. In 2003, he became a student at the Engineering School, Pontificia Universidad Católica de Chile (PUC), where he later received an Industrial Engineering degree with a diploma in Transport Engineering and Logistics. He received the outstanding student award from the Industrial Engineering and the Transport Engineering departments. Between 2009 and 2012 he was head of the R&D team at SHIFT, a workforce management consulting company, where he designed decision support tools to propose employee's shifts and staffing levels meeting demand requirements; he implemented these software services in retailers and public transport operators in Argentina, Chile, Colombia and Perú. Mathias also worked as an adjunct instructor at PUC for two years, where he taught an undergraduate optimization course. In 2012, Mathias moved to Atlanta to pursue doctoral studies in Operations Research at Georgia Tech's Industrial and Systems Engineering Department under the supervision of Drs. Alan Erera and Alejandro Toriello. His work is focused on delivery and routing problems that arise from city-logistics and dynamic optimization techniques to solve them. His research interests also include shift scheduling problems, stochastic vehicle routing and column generation. On a personal level, Mathias is happily married with Francisca Otero since 2012.