



UNIL | Université de Lausanne

Unicentre

CH-1015 Lausanne

<http://serval.unil.ch>

Year : 2020

MANAGING ADVANCED SYNCHRONIZATION ASPECTS IN LOGISTICS SYSTEMS

Coindreau Marc-Antoine

Coindreau Marc-Antoine, 2020, MANAGING ADVANCED SYNCHRONIZATION ASPECTS IN
LOGISTICS SYSTEMS

Originally published at : Thesis, University of Lausanne

Posted at the University of Lausanne Open Archive <http://serval.unil.ch>

Document URN : urn:nbn:ch:serval-BIB_BCEA3723A9E77

Droits d'auteur

L'Université de Lausanne attire expressément l'attention des utilisateurs sur le fait que tous les documents publiés dans l'Archive SERVAL sont protégés par le droit d'auteur, conformément à la loi fédérale sur le droit d'auteur et les droits voisins (LDA). A ce titre, il est indispensable d'obtenir le consentement préalable de l'auteur et/ou de l'éditeur avant toute utilisation d'une oeuvre ou d'une partie d'une oeuvre ne relevant pas d'une utilisation à des fins personnelles au sens de la LDA (art. 19, al. 1 lettre a). A défaut, tout contrevenant s'expose aux sanctions prévues par cette loi. Nous déclinons toute responsabilité en la matière.

Copyright

The University of Lausanne expressly draws the attention of users to the fact that all documents published in the SERVAL Archive are protected by copyright in accordance with federal law on copyright and similar rights (LDA). Accordingly it is indispensable to obtain prior consent from the author and/or publisher before any use of a work or part of a work for purposes other than personal use within the meaning of LDA (art. 19, para. 1 letter a). Failure to do so will expose offenders to the sanctions laid down by this law. We accept no liability in this respect.



UNIL | Université de Lausanne

FACULTÉ DES HAUTES ÉTUDES COMMERCIALES
DÉPARTEMENT DES OPÉRATIONS

**MANAGING ADVANCED SYNCHRONIZATION
ASPECTS IN LOGISTICS SYSTEMS**

THÈSE DE DOCTORAT

présentée à la

Faculté des Hautes Études Commerciales
de l'Université de Lausanne

pour l'obtention du grade de
Docteur en Business Analytics

par

Marc-Antoine COINDREAU

Directeur de thèse
Prof. Olivier Gallay

Co-directeur de thèse
Prof. Nicolas Zufferey

Jury

Prof. Guido Palazzo, Président
Prof. Ann Van Ackere, experte interne
Prof. Frédéric Semet, expert externe
Dr. Eleni Pratsini, experte externe

LAUSANNE
2020



UNIL | Université de Lausanne

FACULTÉ DES HAUTES ÉTUDES COMMERCIALES
DÉPARTEMENT DES OPÉRATIONS

**MANAGING ADVANCED SYNCHRONIZATION
ASPECTS IN LOGISTICS SYSTEMS**

THÈSE DE DOCTORAT

présentée à la

Faculté des Hautes Études Commerciales
de l'Université de Lausanne

pour l'obtention du grade de
Docteur en Business Analytics

par

Marc-Antoine COINDREAU

Directeur de thèse
Prof. Olivier Gallay

Co-directeur de thèse
Prof. Nicolas Zufferey

Jury

Prof. Guido Palazzo, Président
Prof. Ann Van Ackere, experte interne
Prof. Frédéric Semet, expert externe
Dr. Eleni Pratsini, experte externe

LAUSANNE
2020

IMPRIMATUR

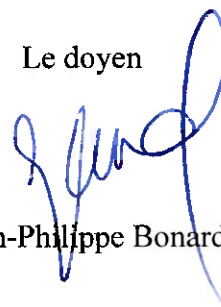
Sans se prononcer sur les opinions de l'auteur, la Faculté des Hautes Etudes Commerciales de l'Université de Lausanne autorise l'impression de la thèse de Monsieur Marc-Antoine COINDREAU, titulaire d'un bachelor en mathématiques de l'École Nationale Supérieure de Techniques Avancées et d'un master de recherche opérationnelle du Conservatoire National des Arts et Métiers, en vue de l'obtention du grade de docteur en Business Analytics.

La thèse est intitulée :

MANAGING ADVANCED SYNCHRONIZATION ASPECTS IN LOGISTICS SYSTEMS

Lausanne, le 19 décembre 2019

Le doyen



Jean-Philippe Bonardi



UNIL | Université de Lausanne

Members of the thesis committee

Prof. Olivier GALLAY

HEC Lausanne, Université de Lausanne

Thesis supervisor

Prof. Nicolas ZUFFEREY

GSEM, Université de Genève

Thesis co-supervisor

Prof. Guido PALAZZO

HEC Lausanne, Université de Lausanne

President of the jury

Prof. Ann van ACKERE

HEC Lausanne, Université de Lausanne

Internal member of the thesis committee

Dr. Eleni PRATSINI

Accenture, Zürich

External member of the thesis committee

Prof. Frédéric SEMET

CRISAL, Centrale Lille

External member of the thesis committee

University of Lausanne
Faculty of Business and Economics

PhD in Business Analytics

I hereby certify that I have examined the doctoral thesis of

Marc-Antoine COINDREAU

and have found it to meet the requirements for a doctoral thesis.

All revisions that I or committee members
made during the doctoral colloquium
have been addressed to my entire satisfaction.

Signature:  _____ Date: 28.11.2019

Prof. Olivier GALLAY
Thesis supervisor

University of Lausanne
Faculty of Business and Economics

PhD in Business Analytics

I hereby certify that I have examined the doctoral thesis of

Marc-Antoine COINDREAU

and have found it to meet the requirements for a doctoral thesis.
All revisions that I or committee members
made during the doctoral colloquium
have been addressed to my entire satisfaction.

Signature:  Date: 27.11.2019

Prof. Nicolas ZUFFEREY
Thesis co-supervisor

University of Lausanne
Faculty of Business and Economics

PhD in Business Analytics

I hereby certify that I have examined the doctoral thesis of

Marc-Antoine COINDREAU

and have found it to meet the requirements for a doctoral thesis.

All revisions that I or committee members
made during the doctoral colloquium
have been addressed to my entire satisfaction.

Signature:  _____ Date: 30/11/2019

Prof. Ann VAN ACKERE
Internal member of the doctoral committee

University of Lausanne
Faculty of Business and Economics

PhD in Business Analytics

I hereby certify that I have examined the doctoral thesis of

Marc-Antoine COINDREAU

and have found it to meet the requirements for a doctoral thesis.

All revisions that I or committee members
made during the doctoral colloquium
have been addressed to my entire satisfaction.

Signature:  Date: 30.11.2019

Dr. Eleni PRATSINI
External member of the doctoral committee

University of Lausanne
Faculty of Business and Economics

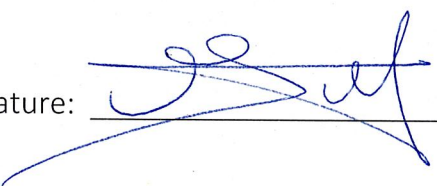
PhD in Business Analytics

I hereby certify that I have examined the doctoral thesis of

Marc-Antoine COINDREAU

and have found it to meet the requirements for a doctoral thesis.

All revisions that I or committee members
made during the doctoral colloquium
have been addressed to my entire satisfaction.

Signature:  Date: 17 / 12 / 2019

Prof. Frédéric SEMET
External member of the doctoral committee

Acknowledgements

First of all, I would like to thank my thesis supervisor, Olivier Gallay. I had the privilege to be your first PhD student. The trust you have placed in me allowed me to feel confident during this PhD and to explore more than just science. I had a real pleasure to participate to this fruitful collaboration. Next, I would like to thank Nicolas Zufferey, my thesis co-supervisor. Thanks for giving to this thesis a relevant industrial context. I would also like to thank Gilbert Laporte. It was a honor to collaborate with such an expert. Thanks for welcoming me at the CIRRELT in Montréal. The summer 2019 was a great accelerator for my thesis.

I want also to thank all the members of this thesis committee for the valuable comments they made. More generally, I would also like to thank the unknown reviewers of some of the chapters of this thesis. It allowed me to significantly improve my work.

I would also like to thank the industrial partners with whom I worked during these four years. Alain Nguyen and Siham Essodaigui, thanks for sharing your data and your industrial insights. Pascal Benchimol, Bayram Kadhour and Thomas Triboulet, thanks for the discussion we had in Lausanne and at the EDF Lab in Saclay. It was moving for me to come back to Saclay, this place where I finished my engineering studies.

Moreover, despite the fact that I was not at the office very often (that is an understatement), I had a very good time at UNIL. I would like to thank the people who have contributed to the friendly atmosphere in the department of operations. I wish I could have shared more meals with you. More importantly, if some of you were not here, I would still try to validate some ECTS at EPFL or at any other scientific institutions. Obtaining these 18 ECTS was a tough task.

I would like to thank the people who made me grow from a more personal point of view. I would like to thank my parents for giving me the opportunity to do long studies and for giving me the taste for effort. I would like to conclude these acknowledgements with a special thanks to the person with whom I share my life. Laura, without you, I would clearly not be here. You have been (and I am sure you will be) an unlimited support. You reassured me when the idea of giving up was growing on my mind. You made me met

Anicet. This little boy is an infinite source of wellness. No matter the sleepless nights, holding him in my arms gave me the strength to finish this thesis. For all the new horizons you made me discover, I will never be enough thankful. I love you.

Abstract

In this thesis, we model various complex logistics problems and develop appropriate techniques to solve them. We improve industrial practices by introducing synchronized solutions to problems that were previously solved independently. The first part of this thesis focuses on cross-docks. We simultaneously optimize supplier orders and cross-docking operations to either reduce the storage space required or evenly distribute workload over the week. The second part of this thesis is devoted to transport problems in which two types of vehicles are synchronized, one of which can be transported by the other. The areas of application range from home services to parcel delivery to customers.

After analyzing the complexity associated with these synchronized solutions (i.e., large-scale problems for which the decisions depend on each other), we design algorithms based on the ‘destroy-and-repair’ principle to find efficient solutions. We also introduce mathematical programs for all the considered problems.

The problems under study arose directly from collaborations with various industrial partners. In this respect, our achieved solutions have been benchmarked with current industrial practice. Depending on the problem, we have been able to reduce the environmental impact generated by the industrial activities, the overall cost, or the social impact. The achieved gains compared to current industrial practice range from 10 to 70%, depending on the application.

Keywords: Logistics, Synchronization, Transportation, Vehicle Routing, Cross-Docking, Mathematical Programming, Metaheuristics (Adaptive Large Neighborhood Search, Variable Neighborhood Search), Matheuristics (Fix-and-Optimize), Real-Life Instances

Résumé

Dans cette thèse, nous modélisons divers problèmes logistiques complexes et développons des techniques appropriées pour les résoudre. Nous cherchons à améliorer certaines pratiques industrielles en introduisant des solutions synchronisées à des problèmes qui étaient auparavant résolus indépendamment. La première partie de cette thèse porte sur les cross-docks. Nous optimisons simultanément les commandes fournisseurs et les opérations au sein de la plateforme de logistique pour réduire l'espace de stockage requis ou répartir uniformément la charge de travail sur la semaine. La deuxième partie de cette thèse est consacrée aux problèmes de transport dans lesquels deux types de véhicules sont synchronisés, l'un pouvant être transporté par l'autre. Les domaines d'application vont du service à domicile à la livraison de colis chez des clients.

Après avoir analysé la complexité des solutions synchronisées (c'est-à-dire des problèmes de grandes dimensions pour lesquels les décisions dépendent les unes des autres), nous concevons des algorithmes basés sur le principe de 'destruction / reconstruction' pour trouver des solutions efficaces. Nous modélisons également les problèmes considérés avec la programmation mathématique.

Les problèmes à l'étude viennent de collaborations avec divers partenaires industriels. A cet égard, les solutions que nous présentons sont comparées aux pratiques industrielles actuelles. En fonction du problème, nous avons pu réduire l'impact environnemental généré par les activités industrielles, le coût global, ou l'impact social des solutions. Les gains obtenus par rapport aux pratiques industrielles actuelles varient de 10 à 70%, selon l'application.

Mot-clefs: Logistique, Synchronisation, Problème de transport, Tournée de véhicules, Plateforme de Cross-dock (transbordement), Programmation Mathématiques, Métaheuristiques, Matheuristiques, Instances Réelles

Contents

Acknowledgements	i
Abstract	ii
Résumé	iii
1 Positioning of the Thesis	1
1.1 General Framework	1
1.2 Specific situations under study	3
1.3 Developed solution methods	5
1.4 Outline	9
2 Integrating workload smoothing and inventory reduction in three inter-modal cross-docking platforms of a European car manufacturer	10
2.1 Introduction	12
2.2 Literature review	15
2.3 Mathematical formulation	18
2.3.1 Complexity of the problem	19
2.3.2 Sets, parameters and variables	19
2.3.3 Quadratic linear programming formulation (Q)	21
2.3.4 (P): Mixed integer linear programming formulation	23
2.3.5 (P_t): decomposition of (P) for a given day t	24
2.3.6 (P_x): decomposition of (P) for a given assignment x of products to boxes in containers	25

2.3.7	$(P_{t,x})$: decomposition of (P_x) for a given day t	26
2.4	Elimination of variables	26
2.4.1	Fixing variables to 0	27
2.4.2	Single-value variables	27
2.4.3	Impact of variable elimination on the ECM instances	28
2.5	Methodology	31
2.5.1	Heuristic for formulation (P_x)	31
2.5.2	Heuristic for large instances	31
2.5.3	Lower bound on f^I	33
2.6	Computational results	34
2.6.1	Notation	34
2.6.2	Comparison of the various optimization approaches	35
2.6.3	Optimal results for the V and G instances	36
2.6.4	Comparison with current practice	36
2.7	Conclusions	40
3	Inbound and Outbound Flow Integration for Cross-Docking Operations	42
3.1	Introduction	44
3.2	Literature Review	47
3.3	Mathematical formulation	49
3.3.1	Sets, parameters and variables	49
3.3.2	Mixed integer linear programming formulation: $Q(O^{(nf)}, I^{(nf)})$. . .	52
3.3.3	Specific configurations of $Q(O^{(nf)}, I^{(nf)})$	54
3.4	Matheuristics	56
3.4.1	Decomposition matheuristic (DM)	56
3.4.2	Fix-and-optimize matheuristic (FOM)	57
3.4.3	Combined matheuristic (DM-FOM)	60
3.4.4	Facilitated implementation of FOM	60
3.5	Computational experiments	61
3.5.1	Test instances	61

3.5.2	Analysis of the performance of the proposed solution methods . . .	62
3.5.3	Managerial insights	67
3.6	Conclusions	71
4	Synchronizing Trucks and Drones for a Real-World Parcel Delivery	
	Problem with Time-Window Constraints	73
4.1	Introduction	75
4.2	Literature Review	77
4.3	Problem Formulation	82
4.3.1	Practical Assumptions	82
4.3.2	Truck-and-Drone Synchronization	83
4.3.3	Sets, Parameters, and Variables	84
4.3.4	Mixed-Integer Linear Program	86
4.4	Solution Methods for the MC-VRPTW-D	89
4.4.1	Adaptive Large Neighborhood Search (ALNS)	89
4.4.2	Route-First-Cluster-Second (RFCS)	93
4.4.3	Initial Solution	94
4.5	Speed up the Insertion Mechanism	95
4.5.1	Modeling Aspects and Notation	95
4.5.2	Greedy Algorithm to Insert a Job at its Best Position	97
4.5.3	Insertion of a Job into a Drone's Schedule	98
4.5.4	Inserting a Job into a Truck's Schedule	101
4.5.5	Complexity of an Insertion	101
4.6	Computational Experiments and Managerial Insights	103
4.6.1	Instances	104
4.6.2	Results	105
4.6.3	Sensitivity Analysis of the Percentage of Jobs Reachable by Drone .	109
4.6.4	Cost Structure of Truck-and-Drone Solutions	110
4.7	Conclusion	114

5	Vehicle Routing with Transportable Resources: Using Carpooling and Walking for On-Site Services	116
5.1	Introduction	118
5.1.1	Industrial context	118
5.1.2	Problem description	119
5.1.3	Contributions and outline	120
5.2	Literature review	122
5.3	Problem formulation	126
5.3.1	Definition and assumptions	126
5.3.2	Graph modeling and variables	127
5.3.3	Mathematical formulation	130
5.4	Methodology	134
5.4.1	VNS: motivation and general principles	134
5.4.2	VNS: shaking phase	136
5.4.3	Complexity of an insertion	139
5.4.4	Accelerating up the insertion phase	139
5.5	Computational experiments	142
5.5.1	Instances	142
5.5.2	Notation and considered configurations	144
5.5.3	MILP results for the VRPTR	145
5.5.4	Performance of VNS on the VRP configuration	145
5.5.5	VNS results for the VRPTR	147
5.5.6	Execution time	150
5.6	Managerial insights	152
5.6.1	Comparison with existing practices	152
5.6.2	Instance characteristics that favor carpooling	156
5.6.3	Expected impact of random perturbations	158
5.7	Conclusion, perspectives, and future works	159
5.8	Acknowledgement	160

6	General conclusion	161
6.1	Scientific contributions	161
6.2	Future work	164
	Abbreviations	166
	Bibliography	168
	Appendices	182
A	Appendix for chapter 5	182
A.1	Fastening the insertion heuristic	182
A.1.1	Modeling: aggregated nodes	182
A.1.2	Vehicle constraints	184
A.1.3	Temporal constraints	184
A.2	Detailed results for all instances	187

List of Tables

- 1.1 Overview of destroy-and-repair solution methods used. 8

- 2.1 Characteristics of the instances. 29
- 2.2 Number of variables in (P) 30
- 2.3 Sizes of the formulations (P) , (P_t) and (P_x) after the elimination of variables. 30
- 2.4 Optimal results for the V and G instances, and performance of Algorithm 2. 37
- 2.5 Results for the V and G instances (i.e., focusing on f^W). 38
- 2.6 Results for the M instances (i.e., focusing on f^I). 39

- 3.1 Considered configurations of the ECM problem. 55
- 3.2 Characteristics of the test instances. 62
- 3.3 Results of Q_z for TDM, FOM and TDM-FOM (M instances). 64
- 3.4 Results of Q for CPLEX, DM and FOM (V and G instances). 65
- 3.5 Results of Q for DM, FM and DM-FOM (M instances). 67
- 3.6 Results of Q for the V and G instances. 68
- 3.7 Results of Q for the M instances. 69

- 4.1 Comparison of related truck-and-drone formulations. 82
- 4.2 Complexity comparison for insertion procedures. 103
- 4.3 Comparison of ALNS and CPLEX for the smaller instances. 106
- 4.4 Comparison of ALNS and RFCS for the larger instances. 108
- 4.5 Result variation for different values of %A. 111
- 4.6 Cost structure of truck-and-drone solutions for different percentages of jobs
 reachable by drone. 113

5.1	Performance of VNS on configuration $P_{no\ walk}^{ W^* }$	147
5.2	Proportion of feasible instances for the different configurations involving less cars than workers.	149
5.3	Detailed results for the representative instances.	150
5.4	Average execution time of the VNS (in seconds) for each instance and time window size.	151
5.5	Aggregated results for the <i>Park-and-Loop</i> configuration $(P_{walk}^{ W^* })$	153
5.6	Results for both the vehicle fleet and the total driving distance for all scenarios.	154
A.1	Detailed results for instances involving 20 jobs.	187
A.2	Detailed results for instances involving 30 jobs.	188
A.3	Detailed results for instances involving 40 jobs.	188
A.4	Detailed results for instances involving 50 jobs.	189

List of Figures

- 1.1 Generic flows in a company. 2
- 1.2 Comparison between a configuration involving direct flows between suppliers and production plants (left side) with a configuration involving a cross-docking platform to consolidate these flows (right side). 3
- 2.1 Product flows associated with an ILP. 12
- 2.2 Assignment of boxes to a container and assignment of products to a box. . . 15
- 2.3 Comparison of the expected f^I -gains of the various approaches. 35
- 2.4 Quantification of the expected f^I -gains for various approaches (average values for all M instances). 40
- 3.1 Product flow in an ILP. 44
- 3.2 Percentage of the maximum theoretical improvement potential achieved by configurations Q_z and Q 70
- 4.1 Different types of truck-and-drone synchronizations allowed at a job location. 84
- 4.2 Path consistency for a drone. 87
- 4.3 Subgraph of the precedence graph after the insertion of job j into the schedule of a drone launched at i and retrieved at k 100
- 4.4 Truck-only (left side) versus truck-and-drone (right side) solutions. Plain (resp. dashed) lines are truck (resp. drone) trips. 109
- 4.5 Aggregated cost structure for instances involving 50 and 100 jobs. 113

5.1	Comparison between a VRPTR solution and the corresponding VRP optimal solution.	121
5.2	Different flow configurations for a WR performed by a motorized worker. . .	129
5.3	Modeling of the VRPTR solution displayed in Figure 5.1 using the introduced sets and variables	130
5.4	An optimal solution to the VRPTR, with both carpooling and walking. . .	146
5.5	Potential driving distance of the solutions found feasible with less cars than workers.	148
5.6	Illustration of a VRPTR solution for which one car is saved and the total driving distance is reduced by 1.3%. The optimal VRP solution and a VRPTR solution are presented for the same instance.	151
5.7	Distribution of the feasible instances and f_{dist} -gains for configuration $\left(P_{walk}^{ W^* -1}\right)$ (i.e., one car is removed from the VRP optimal solution).	157
A.1	Precedence graph representing the VRPTR solution of Figure 5.1. Dotted arcs represent time window constraints (for the sake of clarity, not all time window constraints are drawn), dashed arcs represent precedence constraints due to WRs, and both double and normal arcs represent precedence constraints due to the routes. The order of the nodes in the route must satisfy the constraints in Equations (A.1)–(A.3).	186

Chapter 1

Positioning of the Thesis

1.1 General Framework

This thesis aims at improving logistics operations to find competitive advantages. Logistics (or supply chain management) refer to the process of coordinating people and materials within a company. It includes the purchase and delivery of raw materials, production of goods, and packing, shipment or transportation of products to distributors, for example. In recent decades, supply chain management has received considerable attention in the literature. Continuing to work on existing problems could only lead to marginal gains, as there are already very effective methods to solve these problems. A more promising avenue of research is to identify and solve new formulations related to logistics. In this thesis, I propose to synchronize the solutions of problems that were previously solved independently. More precisely, I focus on the flows existing within a company. Figure 1.1 displays some of these generic flows. In the first part of this thesis, I focus on the flows entering a company (between the suppliers and a company). Next, in a second part, I focus on the flows exiting a company (between the company and its customers). These flows involve multiple actors (e.g., the suppliers and the company or the company and its

customers) and hence it makes sense to consider a high degree of synchronization between them. Indeed, the more actors are synchronized, the better the performance is expected to be.

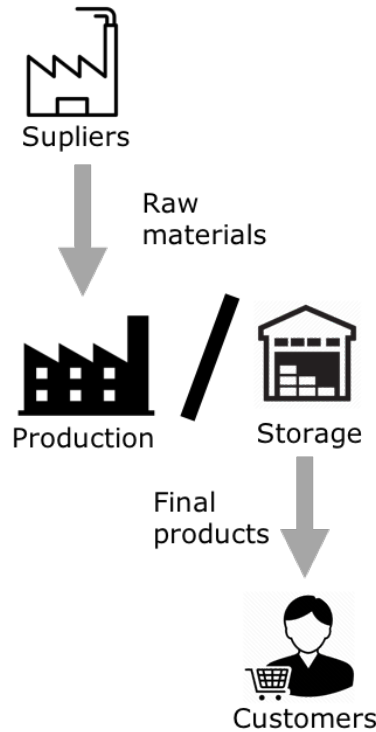


Figure 1.1: Generic flows in a company.

First, I consider a transportation problem occurring in the early stages of the supply chain between the suppliers and the company. More precisely, I consider the case of a company for which there exists a large number of costly flows of raw material between the suppliers and the company. In that case, logistics platforms, called cross-docks, can be introduced in the supply chain to consolidate flows of products. Figure 1.2 shows how flows can be consolidated with the use of cross-docking platforms. An overview of the cross-docking literature can be found in [Ladier and Alpan, 2016]. Despite the fact that synchronization is already the basis of cross-docking activities as the inbound flows of products must be synchronized with the outbound flows. I propose to go one step further by synchronizing supplier orders with operations happening in the cross-docks. The cross-docking platforms considered in this thesis first optimize the supplier orders (i.e., the number of products arriving each day) and then manage the remaining operations (i.e., build and schedule the

outbound flows). Simultaneously optimizing these two problems allows reducing storage space and smoothing workloads during the week.

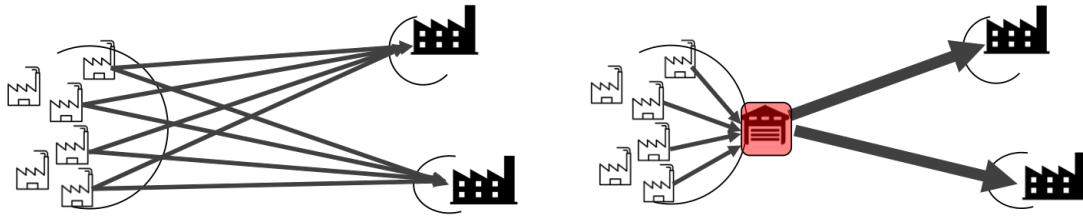


Figure 1.2: Comparison between a configuration involving direct flows between suppliers and production plants (left side) with a configuration involving a cross-docking platform to consolidate these flows (right side).

In the second part of this thesis, I consider the flows exiting a company, between a company and its customers. I either consider home delivery of final products or on-site services performed at customer locations. Synchronization is already part of such problems as the delivery companies must be synchronized with their customers to avoid increased wait times. I propose going one step further by synchronizing two resource types to keep improving the operations. I envision a futuristic but realistic use of drones to transport goods. I show that when synchronized with trucks (a truck can transport a drone and refill it on the way), the delivery cost can be significantly reduced. I also consider the case of a company that provides maintenance work or support at their customers' locations. In that case, I synchronize on-foot workers and motorized workers. As ride-sharing is allowed, I evaluate how synchronization can reduce the environmental impact of the solution (e.g., fewer cars used and less pollution).

1.2 Specific situations under study

First, I consider the cross-docking platforms operated by a *European car manufacturer* (ECM), where the products arrive by trucks from inland suppliers and are sent by containers to offshore production plants. The ECM manages complex loading constraints for the trucks and containers. These constraints force the ECM to wait for the arrival of

several trucks before being able to send the first containers. It results in an increased need for storage and in an unbalanced workload during the week (most of the loading is done during the latter days of the week). I propose to integrate, within the same optimization framework, decisions concerning container loading, truck loading, and container scheduling. Integrating all these decisions creates a difficult optimization problem that has not been addressed in the literature. Chapter 2 describes the optimization of cross-docks' internal decisions (i.e., the suppliers orders are fixed and the remaining decisions focus on the container content and container loading days). Next, Chapter 3 presents a greater optimization problem where cross-dock internal decisions and supplier orders are made simultaneously.

The second part of the thesis is devoted to transportation problems that are extensions of the *Vehicle Routing Problem* (VRP) [Laporte, 2009]. The VRP aims to find optimal routes (e.g., the shortest or cheapest) for a set of vehicles to visit a set of customers. When visiting customers, several types of operations can happen: collect product, deliver parcels or provide on-site services, for example. The applications of such a framework are numerous, it ranges from garbage collection [Kim et al., 2006], mail distribution [Hollis et al., 2006], home health care [Fikar and Hirsch, 2017] and parcel delivery [Cattaruzza et al., 2017]. In this thesis, I introduce an extension of the VRP named the *Vehicle Routing Problem with Transportable Resource* (VRPTR), which designs routes for two types of transportation resources. One resource type, called the light one, can be embedded and transported by the other one, called the heavy one. Depending on the application, each resource type has its own strengths and weaknesses. Synchronizing these two types of resources allows considering a new of solutions that takes advantage of a resource to compensate for the drawbacks of another one.

Chapter 4 presents the case of parcel delivery. The light resources are drones and the heavy resources are trucks. The drones' strengths are their faster speed and lower costs, but they suffer from limited autonomy and capacity. Trucks, however, can move large

amounts of goods but suffer from congestion in cities, cause pollution and noise, and are slower than drones. I show that synchronizing trucks and drones helps reduce the global cost of deliveries when compared to solutions where only trucks are used to transport parcels. Chapter 5 presents the case of on-site services. The light resource is on-foot workers, and the heavy resource is motorized workers. On-foot workers are efficient in congested city centers; because they are not using motorized vehicles, they are not increasing pollution levels and can travel in pedestrian zones, but they are slower than workers who use motorized transportation. Compared with the case where all workers are motorized, I identify configurations that allow reducing the environmental impact of the solution. I show that both the number of vehicles and the driving distance can be reduced to serve the same customers. In these two chapters, synchronization is crucial at rendezvous points. Indeed, drones or non-motorized workers must wait for the vehicles to be transported or to be reloaded in the drone case. Efficient solutions should produce numerous synchronization nodes (where the light and the heavy resources meet) while minimizing the wait times at these nodes.

1.3 Developed solution methods

Combinatorial optimization aims to find in a finite set of solutions the one that maximizes or minimizes a given objective. Logistics problems can be modeled with this framework. Indeed, a solution represents the values assigned to each decision. For example, in cross-docking platforms, one can decide whether a container is loaded on a given day and if product A is transported in container B. However, in a transportation problem, one can decide whether a vehicle goes to customer B after visiting customer A. The solution space denotes all possible values for any decision in the problem (e.g., is the path existing between two customers is selected in the solution?). Next, the objective can vary depending on the goal: financial when minimizing the cost, environmental when minimizing pollution incurred by the activities at stake, or social when improving labour conditions. The

impact of all decisions can be evaluated according to the proposed objective function.

The combinatorial optimization problems are divided into two categories depending on their complexity. An algorithm, for which the time required to return its output increases polynomially with the size of the instance, is called a polynomial algorithm. The combinatorial optimization problems that can be solved (i.e., find the optimal solution) with a polynomial algorithms (i.e., the execution time is a polynomial function of the size of the instance) are assigned to class \mathcal{P} . Conversely, class \mathcal{NP} comprises the problems for which there exists no polynomial algorithms to solve them. In the latter case, the best algorithms that has been developed for \mathcal{NP} -hard problems have a computation time that grows exponentially with the size of the instance. More precisely, a problem is said to be \mathcal{NP} -hard if any other proven \mathcal{NP} -hard problem can be transformed with a polynomial algorithm into it. Then, if a \mathcal{NP} -hard problem is solved with a polynomial algorithm, all other \mathcal{NP} -hard problem can be solved with a polynomial algorithm too. In other words, for the time being, no polynomial algorithm exists to solve \mathcal{NP} -hard problems, and solving optimally a \mathcal{NP} -hard problem can only be done for instances of limited size. Last, it should be noted that increasing synchronization in logistics significantly complicates the problems. Indeed, it creates larger problems for which the solution space is more difficult to explore.

To solve combinatorial optimization problems, several solution methods have been developed. An intuitive approach, called exhaustive search, consists in exploring the whole solutions space to find the best one. Unfortunately, despite the constant progress achieved in computer science, the size of the problems (and, hence, their solution spaces) considered in this thesis prevents such an approach from being feasible. For instance, there are $\frac{1}{2}(n-1)! = (n-1) \times (n-2) \times \dots \times 1$ possible tours that connect n points on a map. There are 6.1×10^{16} different tours that connect twenty points. Assuming a computer can compute one billion tours per second, it would need a few months to evaluate all the possible tours. Operational research is a discipline that focuses on developing optimization tech-

niques to find optimal or good solutions to combinatorial optimization problems. Exact methods (e.g., mathematical programming, column generation, or constraint generation) can guarantee the quality of the solution returned (i.e., prove that it is optimal or at least give a gap with the optimal solution). The main drawback of these approaches is the curse of dimensionality (i.e., when the size of the problem is too large, exact methods might be unable to find either a competitive solution within a given time budget or might even be unable to find a feasible solution even after a long execution time). However, heuristics use practical methods to build a solution. With these methods, there is no guarantee that the optimal solution will be found, but good quality solutions can be found quickly. Another method, called metaheuristics, orchestrates several heuristics within the same framework to continue improving the solution. Matheuristics differ from metaheuristics as they combine heuristics with mathematical programming to explore the solution space. From a practical standpoint, mathematical programming is a good starting point to model new problems. Generic solvers, either commercial or open source (e.g., CPLEX, Gurobi, GLPK and COIN-OR), have been developed to solve mathematical programs. Despite the constant progress made in the development of these solvers, they are unable to solve large problems (as those encountered in this thesis). In this case, one can either develop more refined exact methods (e.g., improve the modeling or decompose or strengthen the problem) or build heuristic methods. To solve the problems under study, I developed metaheuristics and a matheuristic.

Inspired by industrial partners, the problems presented in this thesis are all \mathcal{NP} -hard. I propose a *Mixed Integer Linear Program* (MILP) and metaheuristics or matheuristics based on the destroy-and-repair principle to solve them. In the latter case, the solutions are iteratively improved by repeating a procedure called destroy-and-repair cycle. A destroy-and-repair cycle generates a new solution by working on a limited part of the solution space (i.e., the part of the solution that is ‘destroyed’) instead of reconsidering the whole solution space at a time. These smaller problems are easier to solve, and repeating this procedure allows the algorithm to find competitive solutions. Table 1.1 displays

the differences between the three destroy-and-repair solution methods introduced in this thesis. I differentiate these methods according to the solution methods used to destroy and to repair a solution and according to the used acceptance criterion (“Accept. Crit.”). The acceptance criterion refers the choice that is made each time a new solution is generated after a destroy-and-repair cycle. When the new solution does not improve the current solution, is it better to start working on the new generated solution or to continue working on the previous solution? While a descent algorithm only accepts improving solutions, the metropolis criterion accepts some deteriorating solutions with a probability which depends on the deterioration and on the execution time. The metaheuristics proposed in this thesis (the *Adaptive Large Neighborhood Search* in Chapter 4 and the *Variable Neighborhood Search* in Chapter 5) use the large neighborhood search algorithm (LNS) [Shaw, 1998] to destroy and to repair a solution, the proposed matheuristic (Fix-and-Optimize Matheuristic detailed in Chapter 3) uses mathematical programming during the repair phase. After highlighting the limited efficiency of a commercial solvers (CPLEX) to find solution of the MILP, I demonstrate that this panel of destroy-and-repair solution methods performs well.

Table 1.1: Overview of destroy-and-repair solution methods used.

Methods Accept. Crit.	MILP Solver	LNS
Descent	Fix-and-Optimize Matheuristic (Chapter 3)	Variable Neighborhood Search (Chapter 5)
Metropolis		Adaptive Large Neighborhood Search (Chapter 4)

Some of the introduced formulation involve multiple objectives. In some cases, the objectives can be in conflict (see e.g., Chapter 5). In the latter case, optimizing an objective might deteriorate the other one. In this thesis, I use lexicographic optimization to manage multiple objectives. In this framework, objectives are classified (e.g., the industrial partners rank the objectives) and are solved independently. The primary objective (i.e., the most important one) is optimized first and then, the secondary objective is optimized

while keeping the primary objective at its best value. By doing so, I was able to reduce the execution time (the simultaneous consideration of all objectives makes the problem more complex).

1.4 Outline

The thesis is organised as follows. Chapters 2 and 3 discuss cross-docking. Chapters 4 and 5 focus on the VRPTR in two different contexts. All these chapters are independent and some of them have been published in international journals. Chapter 2 has been published in the journal *Computers and Operations Research* and Chapter 5 has been published in the *European Journal Of Operational Research*. These chapters are followed by a general conclusion that summarizes the common scientific contributions and opens the door for future research.

Chapter 2

Integrating workload smoothing and inventory reduction in three intermodal cross-docking platforms of a European car manufacturer

MARC-ANTOINE COINDREAU - *University of Lausanne, Switzerland*

OLIVIER GALLAY - *University of Lausanne, Switzerland*

NICOLAS ZUFFEREY - *University of Geneva, Switzerland*

GILBERT LAPORTE - *HEC Montréal, Canada*

*Chapter published in **Computers and Operations Research** [Coindreau et al., 2019b]*

Abstract

We consider the optimization of container loading at three intermodal logistics platforms (ILP) of a large European car manufacturer (ECM). The decisions focus both on the loading day of each container and on its filling with the products in inventory, which are gradually received over the week from inland suppliers. The objective is either to reduce the largest inventory level needed in the ILP, or to smooth the weekly workload. We develop a solution methodology that allows the handling of complex loading constraints related to dimensions and weight of the products. We model the problem as a mixed integer linear program and we develop a decomposition heuristic to solve it. We perform extensive computation tests on real instances provided by ECM. Compared with current industrial practices, our solutions yield an average improvement of 46.8% for the inventory reduction and of 25.8% for the smoothing of the workload. Our results highlight the benefit of jointly optimizing container loading and operations scheduling.

Keywords: Logistics, intermodal logistics platforms, cross-docking, loading constraints, MILP, decomposition heuristics.

2.1 Introduction

We consider the operational management of intermodal logistics platforms (ILP) of a large European car manufacturer denoted by ECM because of a non-disclosure agreement. We refer to this problem as the *ECM problem*. Over a given planning horizon (a week in this work, excluding the weekend), each ILP consolidates product flows from inland suppliers to offshore production plants, which are the ILP clients. Every day, products are unloaded from trucks and are then loaded into containers. Once loaded, the containers are transmitted to the shipping company that will ship them at the end of the week. The products not shipped at the end of a day are stored and wait until the next day to be loaded on a container. Figure 2.1 illustrates the sequence of operations at an ILP. It shows a product flow from trucks to an ILP, to containers, ships to clients.

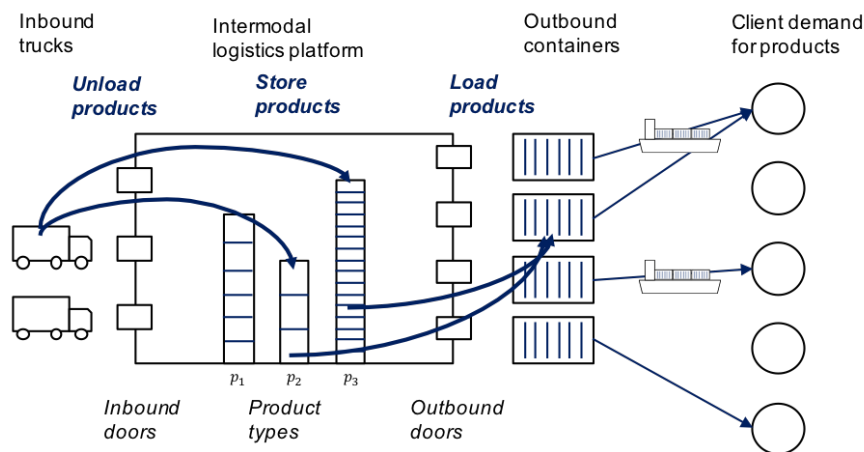


Figure 2.1: Product flows associated with an ILP.

The managers have to decide when to load each container in order to minimize the largest weekly inventory space (f^I) employed, or to smooth the workload activities (f^W). Focusing on f^I , it is preferable to load the containers as soon as possible. Minimizing f^W yield solutions with approximately the same number of containers filled each day. The two objectives f^I and f^W are minimized lexicographically, and the priority of an objective

on the other depends on the considered ILP: the ILPs with a loaded volume larger than 10,000 m³ per week focus on f^I , whereas the smaller ILPs focus on f^W .

We assume that the number of containers loaded per day is not constrained and the sequence of container loadings has no impact on the objective function. Each client can receive multiple containers, but each container can only be sent to its assigned client. Products of the same type are interchangeable among clients. The daily workload is measured as the volume of products unloaded from the trucks and the volume of products loaded into containers (i.e., it is the sum of the two volumes). The objective f^I is measured as the largest volume, over the week, of products remaining in the inventory at the end of a day. Regarding the inbound side, the trucks gradually deliver over the week all products needed for the ILP clients. The truck arrival days and their contents are considered as input data.

Concerning the outbound side, the client demand must be sent in containers by ship. A container can only be loaded when its full content is available in the ILP inventory. A first decision is to determine the loading day of each container, given that its content is fixed. Minimizing the number of containers (by optimizing their contents) to ship all the demand is a difficult problem due to the presence of complex loading requirements, which involve 3D constraints (each box has a 3D shape and overlaying is forbidden in the containers), a total weight limitation (the total weight of all boxes loaded in the same container cannot exceed 22 tonnes), and the arrangement of boxes in stacks (the boxes are loaded in stacks and the range of allowed weights is limited by the height of the boxes in the stack). For an overview of common loading constraints, see [Toffolo et al., 2017]. The current practice at ECM consists of first optimizing the loading of the containers, and next of computing a weekly schedule for the associated operations. In the present work, the loading and the scheduling of the containers are optimized simultaneously through the minimization of f^I or f^W .

As highlighted by [Toffolo et al., 2017], the loading problem itself is rather complicated and cumbersome. ECM solves this problem by using a dedicated algorithm that minimizes the number of containers (to ship the weekly demand) and satisfies the full set of 3D loading constraints. Therefore, ECM provided us with a feasible initial assignment of products to containers, where the products are packaged into boxes, and the boxes are loaded into containers. There are several product types and different box types. Usually, various product types are eligible to be loaded in a box. However, once loaded, a box can only contain a single product type to be determined (whatever the product type loaded, the boxes are always filled to the maximum capacity). Starting from the ECM box-to-container assignments, we propose to revoke the decisions concerning the allocation of products to boxes. More precisely, we can modify the full content of each box with a tolerance of 10 kg (hence precluding any violation of the weight of a stack), but not its dimensions. In other words, we allow some permutations between the content of the boxes. Based on the employed real data, we have observed that 70% of the boxes can contain different product types, which means that the proposed permutation search space is likely to be large enough for the generation of very different solutions and for exhibiting a significant optimization potential. This type of box-content permutations allows us to keep tractable the high complexity related to *container loading*, this in order to integrate it with *container scheduling*. This integration would be very cumbersome if we were to consider the loading problem in its full complexity. The left part of Figure 2.2 depicts an assignment of boxes to a container. In this example, there are three types of boxes: b_1 , b_2 and b_3 . The container is loaded with two boxes of type b_1 , five boxes of type b_2 and three boxes of type b_3 . The right part of the figure shows that each box of type b_1 can hold either four products of type p_1 (with a total weight of 74 kg) or two products of type p_2 (with a total weight of 71 kg).

We make the following scientific contributions. We propose a mixed integer linear programming (MILP) model and a decomposition heuristic to solve the ECM problem. Using real data, we compare our results with the current practice for three different ILPs, and

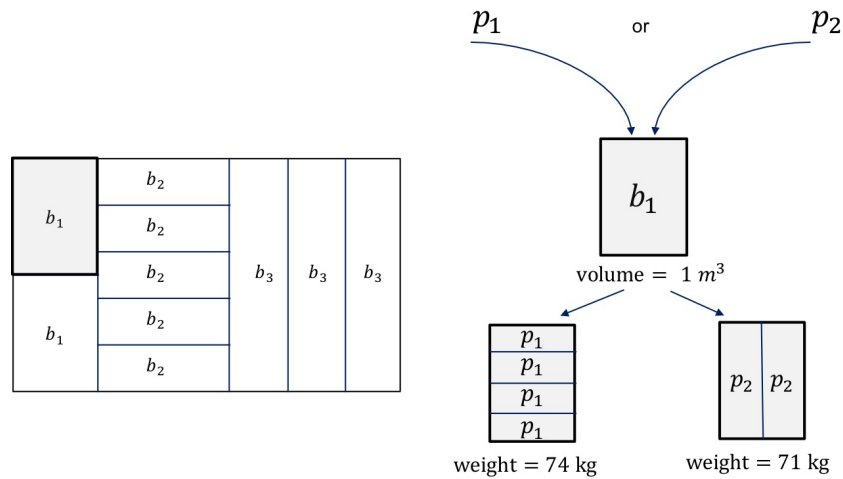


Figure 2.2: Assignment of boxes to a container and assignment of products to a box.

we assess the benefits of integrating container loading and container scheduling.

The remainder of the paper is organized as follows. Section 2.2 surveys the related literature. In Section 2.3, after determining the complexity of the problem, we present the MILP, as well as three decomposition strategies. Section 2.4 proposes two ways of eliminating variables from the MILP formulation in order to reduce its size. The solution method is described in Section 2.5, followed in Section 2.6 by computational experiments, where the efficiency of the proposed heuristic is assessed by making comparison with optimal solutions and with current industrial practices. Conclusions follow in Section 2.7.

2.2 Literature review

The ECM problem shares some similarities with cross-dock scheduling. As in cross-docks, the ILPs act as consolidating points, the aim of which is to receive and unload an incoming flow of products arriving by truck, and then to load these products into outbound containers after a sorting process [Boysen and Fliedner, 2010]. Whereas in most of the cross-dock literature, the incoming products are immediately reloaded, hence precluding

the use of storage, in the ILPs, products may have to be stored and wait at most until the end of the current week before being loaded into the containers. This is due to the fact that the totality of the content of a container must be available in order to launch the loading process. However, at a weekly level, all products received in the ILPs are sent (i.e., it is not possible to decide to hold any item in inventory for additional weeks) and the final storage at the end of the week is expected to be null (like in classical cross-dock platforms at a daily level). The following paragraphs review the contributions on cross-docking that are the most related to the ECM problem.

According to the survey of [Van Belle et al., 2012], most of the research on cross-docking has been undertaken after 2004. Only the literature concerning operational decisions is relevant to our study. Even though a number of papers that consider operational decisions share some specificities with the ECM problem (e.g., scheduling outbound flows when all the requested products are in inventory), their focus is usually on operational modeling, such as internal activities in the cross-dock [Bellanger et al., 2013], truck-to-door assignment (determining at which door a truck must be unloaded or loaded in order to minimize the movements in the cross-dock) [Enderer et al., 2017, Maknoon et al., 2017], or combining cross-dock and vehicle routing to minimize routing costs while satisfying internal constraints [Maknoon and Laporte, 2017].

The ECM problem contains some features of the *Truck Scheduling Cross-Dock* (TSCD) problem, for which a review can be found in [Boysen and Fliedner, 2010]. The TSCD focuses on the synchronization of inbound and outbound trucks to maximize the number of products that can directly be loaded in outbound trucks [Buijs et al., 2014], ideally while avoiding storage [Boysen, 2010]. [Yu and Egbelu, 2008] similarly consider a TSCD with storage considerations. As in the ECM problem, the products arrive at the inbound doors from suppliers and are then sent by trucks to clients. The products are also interchangeable. The main difference between the work of [Yu and Egbelu, 2008] and ours lies in the fact that these authors consider only scalar constraints for the loading of outbound

trucks (i.e., the constraints only concern the total weight of the transported products, but neither their size nor their position in the container is considered).

[Serrano et al., 2017] studied a similar cross-docking platform, where temporary storage is allowed between truck arrival and container loading. However, in contrast to the ECM problem, scalar loading constraints are considered. The goal of the authors is to delay the minimum number of inbound trucks so that all internal constraints of the cross-docking platform are satisfied (i.e., storage, repacking and sorting activities).

Smoothing the workload has already been considered as an objective by [Ladier et al., 2014]. However, the context of their paper differs significantly from ours since the decisions involved concern the workforce dimensioning at a strategic level and the scheduling of specific activities during the day. As highlighted in [Merengo et al., 1999, Emde et al., 2010] in the context of assembly line balancing, there exist various criteria for workload smoothing (e.g., minimize either the sum of the divergences to the mean or the maximum divergence to the mean). In our situation, the choice of minimizing the difference between the day with the smallest workload and the day with the largest one results from discussions with ECM. We could also have minimized the largest workload observed during the week, but in the latter case, we could have observed a non-balanced workload for the less loaded days. Since the number of container-loading stations is fixed at the ECM ILPs, any high daily workload results in overtime, and any low daily workload creates idle times for some workers. Ultimately, the solutions obtained when minimizing the above-proposed function are likely to be similar to those that would be obtained when minimizing *the maximum divergence with respect to the mean daily workload*.

As highlighted by [Toffolo et al., 2017], real-world multiple-container loading problems generalize the 3D packing problem. Accordingly, the associated combinatorial optimization problems are computationally complex due to the considered large set of specific constraints. As mentioned in Section 2.1, these constraints do not only concern the weight

and size of the boxes to be loaded in the container, but also the specific way in which these boxes can be arranged into stacks and layers. In general, the objective focuses on minimizing the total number of containers required for packing the boxes. Container loading problems are most often solved using approximate solution methods. In particular, [Toffolo et al., 2017] propose a two-phase metaheuristic, the first phase of which consists of quickly generating a feasible assignment of the boxes to containers, which is then improved in a second phase through diversification and intensification mechanisms. Whereas loading and routing decisions have already been considered jointly (e.g., [Gendreau et al., 2006]), no study seems to have addressed the complex set of real-world loading constraints arising in our problem in the context of optimizing the container-scheduling operations of a cross-dock. Here, we propose to integrate container loading and container scheduling decisions in the following way: starting from a feasible assignment of boxes to containers, we exploit the fact that some box types can transport different product types. As a result, we are able to take advantage of the huge variety of feasible assignments of products to containers while still satisfying the loading constraints.

2.3 Mathematical formulation

We discuss the complexity of the ECM problem in Section 2.3.1. In Section 2.3.2, we introduce the sets, parameters and variables needed to describe the model. Next, we present in Section 2.3.3 a quadratic formulation (Q) to accurately express the constraints and objectives. This formulation is then linearized in Section 2.3.4. After discussing the size of the linear formulation (P), we propose three decompositions, fixing either the day in (P_t) (Section 2.3.5), the container contents in (P_x) (Section 2.3.6), or both in ($P_{t,x}$) (Section 2.3.7).

2.3.1 Complexity of the problem

To determine the complexity of the problem, assume the following:

- The planning horizon is limited to one day. Then, f^W is dropped and minimizing f^I is equivalent to maximizing the volume of the shipped products.
- The inbound trucks do not deliver all the demand during the limited planning horizon (the remaining part of the demand will be delivered on the other days).
- The number of box types is equal to the number of product types (i.e., the content of the containers and their volumes is an input data that cannot be modified).

The resulting subproblem consists of choosing which containers to load during the day in order to maximize the shipped volume. The loading of a container is limited by the products available in the ILP inventory. Moreover, not all containers can be loaded because the demand has only been partially delivered. If there is only one product type, the subproblem is the *Knapsack Problem*, i.e., maximize a utility function while satisfying a volume constraint [Pisinger, 1997]. Since there is an inventory for each product type, this special case actually corresponds to the *Multidimensional Knapsack Problem* (MKP) [Puchinger et al., 2010], i.e., maximize a utility function (which is the shipped volume here) under multiple volume constraints (which correspond to the product availability here). This subproblem is described in Section 2.3.7. Since the MKP is \mathcal{NP} -hard [Puchinger et al., 2010], the ECM problem is also \mathcal{NP} -hard.

2.3.2 Sets, parameters and variables

We now introduce our notations:

Sets

- T : set of time periods (i.e., days),
- C : set of clients,
- O : set of outbound containers,
- $O^c \subseteq O$: subset of containers assigned to client $c \in C$,
- P : set of product types,
- B : set of box types.

Parameters

- $d_{cp} \in \mathbb{N}$: demand (in units) of client $c \in C$ for product type $p \in P$,
- $r_{pt} \in \mathbb{N}$: number of units of product type $p \in P$ received on day $t \in T$,
- $n_{ob} \in \mathbb{N}$: number of units of boxes of type $b \in B$ transported in container $o \in O$,
- $q_{pb} \in \mathbb{N}$: number of units of product type $p \in P$ that can be transported in box type $b \in B$,
- $l_{pb} \in \mathbb{R}^+$: weight (in kg) of a box of type $b \in B$ when filled with product type $p \in P$,
- $l_M \in \mathbb{R}^+$: maximum allowed weight (in kg) that can be transported by a container,
- $a_b \in \mathbb{R}^+$: volume (in m^3) of a box of type $b \in B$,
- $h_p = \min_{b \in B} \{a_b/q_{pb}\}$: volume (in m^3) of a product of type $p \in P$.

Decision variables

- $x_{obp} \in \mathbb{N}$: number of boxes of type $b \in B$ assigned to product type $p \in P$ in container $o \in O$,
- $y_{ot} = 1$ if outbound container $o \in O$ is loaded on day $t \in T$; $y_{ot} = 0$, otherwise,
- $u_{pt} \in \mathbb{N}$: number of units of product type $p \in P$ in stock on day $t \in T$ before loading the containers,
- $v_{pt} \in \mathbb{N}$: number of units of product type $p \in P$ in stock on day $t \in T$ after loading the containers,,
- $s_{pt} \in \mathbb{N}$: number of units of product type $p \in P$ sent on day $t \in T$,
- $w_t \in \mathbb{R}^+$: workload on day t (in m^3),
- $f^I \in \mathbb{R}^+$: largest inventory value (in m^3) encountered during the planning horizon,
- $f^W \in \mathbb{R}^+$: largest workload imbalance (in m^3), i.e., largest difference between the most loaded day and the least loaded one.

2.3.3 Quadratic linear programming formulation (Q)

ECM considers two objectives f^I and f^W , as mentioned in Section 2.1. These objectives are minimized in a lexicographic fashion (i.e., the higher-level objective is first minimized, and the lower-level objective is then minimized while constraining the first one at its best value). The priority of an objective depends on the considered ILP.

Objectives

$$\text{minimize } f^I \tag{2.1}$$

$$\text{minimize } f^W \tag{2.2}$$

Over the planning horizon, objective (2.1) aims at minimizing the largest storage space used in the ILP (f^I), whereas objective (2.2) focuses on minimizing the workload imbalance (f^W). Both objectives are measured in m^3 .

Constraints

$$\sum_{t \in T} y_{ot} = 1 \quad \forall o \in O \quad (2.3)$$

$$\sum_{o \in O^c} \sum_{b \in B} q_{pb} \cdot x_{obp} \geq d_{cp} \quad \forall c \in C, p \in P \quad (2.4)$$

$$\sum_{b \in B} \sum_{p \in P} l_{pb} \cdot x_{obp} \leq l_M \quad \forall o \in O \quad (2.5)$$

$$\sum_{p \in P} x_{obp} \leq n_{ob} \quad \forall o \in O, b \in B \quad (2.6)$$

$$s_{pt} = \sum_{o \in O} \sum_{b \in B} q_{pb} \cdot x_{obp} \cdot y_{ot} \quad \forall p \in P, t \in T \quad (2.7)$$

$$u_{pt} = v_{p,t-1} + r_{pt} \quad \forall p \in P, t \in T \quad (2.8)$$

$$v_{pt} = u_{pt} - s_{pt} \quad \forall p \in P, t \in T \quad (2.9)$$

$$f^I \geq \sum_{p \in P} h_p \cdot v_{pt} \quad \forall t \in T \quad (2.10)$$

$$w_t = \sum_{p \in P} h_p \cdot r_{pt} + \sum_{o \in O} \sum_{b \in B} \sum_{p \in P} a_b \cdot x_{obp} \cdot y_{ot} \quad \forall t \in T \quad (2.11)$$

$$f^W \geq w_{t_1} - w_{t_2} \quad \forall t_1, t_2 \in T. \quad (2.12)$$

$$x_{obp}, v_{pt}, r_{pt}, s_{pt} \in \mathbb{N} \quad (2.13)$$

$$y_{ot} \in \{0, 1\} \quad (2.14)$$

$$w_t, f^W, f^I \in \mathbb{R} \quad (2.15)$$

Constraints (2.3) prevent a container from being loaded multiple times. Constraints (2.4) impose that the demand of each client is satisfied. Constraints (2.5) ensure that the weight of the transported products does not exceed the container capacity. Similarly, constraints (2.6) ensure that the number of boxes transported in a container does not exceed the allowed limit. Constraints (2.7) compute the amount of product type $p \in P$ sent on day $t \in T$. Constraints (2.8) (resp. (2.9)) update the available inventory in the

ILP before (resp. after) loading containers on day $t \in T$. In constraints (2.9), $v_{p0} \geq 0$ are parameters that give the quantity of products of type p initially present in the ILP (some products are not received during the week, but are in the inventory at the beginning of the week). Constraints (2.10) compute the largest amount of storage space used in the ILP. Constraints (2.11) compute the workload for day $t \in T$. Constraints (2.12) evaluate the gap between the heaviest and lightest workloads over all days. Constraints (2.13 – 2.15) give the domain of the variables.

2.3.4 (P): Mixed integer linear programming formulation

We denote by (P) the linearized formulation of (Q) and by (P^I) (resp. (P^W)) formulation (P) in which f^I (resp. f^W) is the objective. Additionally, (P^{IW}) (resp. (P^{WI})) corresponds to formulation (P) where f^I (resp. f^W) is minimized and f^W (resp. f^I) is constrained at its best-known value. The variables z_{obpt} are introduced to linearize the product $x_{obp} \cdot y_{ot}$. Formulation (P) keeps the constraints of (Q) that do not involve the product $x_{obp} \cdot y_{ot}$ (i.e., constraints (2.3–2.6, 2.8–2.10, 2.12)). In (P), constraints (2.7) and (2.11) become constraints (2.16) and (2.17), respectively. In addition, constraints (2.18–2.20) are added to fix z_{obpt} at its appropriate value (i.e., $z_{obpt} = 0$ if $y_{ot} = 0$, and $z_{obpt} = x_{obp}$ if $y_{ot} = 1$). The constraints of the linearized model are then:

$$s_{pt} = \sum_{o \in O} \sum_{b \in B} q_{pb} \cdot z_{obpt} \quad \forall p \in P, t \in T \quad (2.16)$$

$$w_t = \sum_{p \in P} h_p \cdot r_{pt} + \sum_{o \in O} \sum_{b \in B} \sum_{p \in P} a_b \cdot z_{obpt} \quad \forall t \in T \quad (2.17)$$

$$z_{obpt} \leq n_{ob} \cdot y_{ot} \quad \forall t \in T, o \in O, p \in P, b \in B \quad (2.18)$$

$$z_{obpt} \leq x_{obp} \quad \forall t \in T, o \in O, p \in P, b \in B \quad (2.19)$$

$$z_{obpt} + n_{ob} \cdot (1 - y_{ot}) \geq x_{obp} \quad \forall t \in T, o \in O, p \in P, b \in B. \quad (2.20)$$

There are $|O| \cdot |P| \cdot |B| \cdot |T|$ variables z_{obpt} in the linearization, i.e., more than 7 million

for the smallest instance and above 32 billion for the largest instance considered in this study (the exact number of variables for the instances provided by ECM is given in Section 2.4.3). The size of the proposed MILP precludes commercial solvers from finding solutions for the large instances, and even from inputting the data. We therefore introduce some decompositions.

2.3.5 (P_t) : decomposition of (P) for a given day t

Formulation (P) can be decomposed into subproblems (P_t) for each day $t \in T$. In (P_t) , the variables concerning the inventory (i.e., u_{pt} and v_{pt}) are dropped and the other variables remain the same (but the index related to day $t \in T$ is removed). Some notations are also modified: $u_p^{(t)}$ represents the amount of product type $p \in P$ available at the beginning of day t ; $O^{(t)} \subseteq O$ represents the subset of containers to load at day $t \in T$; $d_{cp}^{(t)}$ is the demand (in units) of client $c \in C$ for product $p \in P$ not already loaded before day $t \in T$ (since other containers can be already shipped in previous days, and a part of the demand may be already satisfied); it replaces d_{cp} in constraints (2.4).

Regarding the objectives, f^W is dropped when optimizing over a single day, and f^I is equivalent to maximizing the volume of products sent at the end of the day. At the end of Section 2.5.2, we explain how (P_t) is used in the heuristic proposed to solve the ECM problem, and how f^W can be optimized a posteriori. The objective and constraints hence

become:

$$\text{maximize } \sum_{o \in O^{(t)}} \sum_{p \in P} \sum_{b \in B} q_{pb} \cdot h_p \cdot z_{obp} \quad (2.21)$$

$$\text{subject to } \sum_{o \in O^{(t)}} \sum_{b \in B} q_{pb} \cdot z_{obp} \leq u_p^{(t)} \quad \forall p \in P \quad (2.22)$$

$$z_{obp} \leq n_{ob} \cdot y_o \quad \forall o \in O^{(t)}, p \in P, b \in B \quad (2.23)$$

$$z_{obp} \leq x_{obp} \quad \forall o \in O^{(t)}, p \in P, b \in B \quad (2.24)$$

$$z_{obp} + n_{ob} \cdot (1 - y_o) \geq x_{obp} \quad \forall o \in O^{(t)}, p \in P, b \in B. \quad (2.25)$$

Objective (2.21) maximizes the volume sent at the end of the day. Constraints (2.4–2.6), on the assignment of products to containers are kept. The inventory constraints (2.8, 2.9, 2.16) are replaced with (2.22) which impose that the amount of products sent does not exceed the available inventory. Constraints (2.18–2.20) which are linked to the linearization are modified into constraints (2.23–2.25).

2.3.6 (P_x) : decomposition of (P) for a given assignment x of products to boxes in containers

If the assignment of products to containers is known, like in the ECM current solution, the only decisions concern the loading day of each container. We call this problem the *Container Scheduling Problem*. It is related to a problem proposed by [Larbi et al., 2011] in which a TSCD is considered and the content of both inbound and outbound trucks are known. A subproblem (P_x) is derived from (P) , where the decision variables x_{obp} are fixed (e.g., those taken in a feasible solution, for instance the solution used by ECM), $m_{op} = \sum_{b \in B} x_{obp} \cdot q_{pb}$ is an input that represents the number of units of product p in container $o \in O$. The set B of boxes and the designation C of the clients are no longer needed. As for formulation (P) , the following problems can be considered: (P_x^I) , (P_x^W) , $(P_x^{I|W})$, $(P_x^{W|I})$. With respect to formulation (P) , the objectives remain the same. Constraints

(2.4–2.6), related to the assignment of products to boxes, become redundant. Constraints (2.3, 2.8–2.10, 2.12) remain unchanged; constraints (2.7), which compute the amount of shipped products, become constraints (2.26); constraints (2.11), which set the workload, become constraints (2.27). The constraints then become:

$$s_{pt} = \sum_{o \in O} m_{op} \cdot y_{ot} \quad \forall p \in P, t \in T \quad (2.26)$$

$$w_t = \sum_{p \in P} h_p \cdot r_{pt} + \sum_{o \in O} \sum_{p \in P} m_{op} \cdot h_p \cdot y_{ot} \quad \forall t \in T. \quad (2.27)$$

2.3.7 $(P_{t,x})$: decomposition of (P_x) for a given day t

The model $(P_{t,x})$ uses the same formalism as (P_t) and (P_x) . For a given day t , $(P_{t,x})$ maximizes the volume of containers loaded under the constraint of available inventory $(u_p^{(t)})$, as formulated below. This model is equivalent to the MKP. Indeed, the objective is to choose the appropriate containers to load in order to maximize the volume shipped at the end of the day, while imposing a capacity constraint for each product type:

$$\text{maximize } \sum_{o \in O^{(t)}} \left(\sum_{p \in P} m_{op} \cdot h_p \right) \cdot y_o \quad (2.28)$$

$$\text{subject to } \sum_{o \in O^{(t)}} m_{op} \cdot y_o \leq u_p^{(t)} \quad \forall p \in P. \quad (2.29)$$

2.4 Elimination of variables

As already highlighted in Section 2.3.4, formulation (P) (as well as its decomposition (P_t)) involves the variables x_{obp} , the number of which increases with the cardinality of the sets O , B and P , which yields a number of x_{obp} variables so large that commercial solvers cannot even read the model (an accurate evaluation of the number of variables is given in Section 2.4.3). In this section, we present a technique to fix the variables that can only

take a single value at optimality, thus allowing their removal.

2.4.1 Fixing variables to 0

The variable x_{obp} counts the number of boxes of type b loaded with product of type p in container o . If container o does not transport a box of type b (i.e., if $n_{ob} = 0$), then the value of x_{obp} is forced to zero for each product type $p \in P$. Similarly, if box type b cannot transport a product of type p (i.e., $q_{pb} = 0$) because of non-matching sizes, weight limitations or product requirements, then x_{obp} is forced to zero for each container $o \in O$. These fixed variables can therefore be dropped from (P) , thus reducing the size of the model. Let $X = \{x_{obp} \mid n_{ob} \cdot q_{pb} > 0\}$ be the set of variables that have not been fixed to 0.

2.4.2 Single-value variables

A further reduction of X is achieved as follows. Formulation (P) looks for an optimal assignment of products to containers under the constraint that each box type b can only transport a subset of product types and that the assignment of boxes to containers is known (while satisfying complex loading constraints). We can count the maximum number of items of a given product type p that can be transported by the containers associated with each client c (i.e., $\sum_{o \in O^c} \sum_{b \in B} n_{ob} \cdot q_{pb}$). If this number is equal to the demand of client c , then the only way to satisfy this demand is to assign the products to the boxes that can transport them. More formally, let $\tilde{X} = \{x_{obp} \in X \mid \sum_{o' \in O^{c_o}} \sum_{b \in B} n_{ob} \cdot q_{pb} > d_{cp}\}$, where c_o is the client served by container o . The set of the non-fixed variables is $\tilde{X} \subseteq X$.

2.4.3 Impact of variable elimination on the ECM instances

ECM operates three ILPs denoted by V, G and M. For each ILP, a representative set of data was provided by ECM, which captures the essential characteristics and situations observed over the past years. The V and G ILPs are not challenged by the storage space, and hence the main objective at these locations is f^W . In contrast, f^I is the main objective for the ILP M since its large quantity of transiting products requires an important storage space, and the restricted available space must be used efficiently. Table 2.1 provides the characteristics of the 17 ECM instances under study. Column $|O|$ gives the number of containers to load during the week. Column $|B|$ (resp. $|P|$) indicates the number of different box types (resp. product types). Column $|C|$ gives the number of clients served by the ILP. Column “Nb. Boxes” (resp. “Nb. Products”) gives the total number of boxes transported by the containers (resp. the total number of products transiting in the ILP). Column “% Boxes” displays the percentage of transported boxes that can receive a product different from the one carried in the ECM solution (within a tolerance of 10 kg per box). $I^{(sent)}$ represents the total inventory volume (in m^3) sent during the week (relative to all the products in column “Nb. Products”), and $I^{(init)}$ represents the inventory level of the ILP at the beginning of the week. In our instances, the products are either delivered during the week by inbound trucks, or are carried in the inventory from a previous week, and are therefore already available at the beginning of the week.

Table 2.2 provides the value of the product $|O| \cdot |B| \cdot |P|$ for each instance (i.e., the number of variables x_{obp}). Column “ $|X|$ ” (resp. “ $|\tilde{X}|$ ”) indicates the size of set X (resp. \tilde{X}). Column “% non-fixed” gives the percentage of variables that are not fixed. After the variables elimination process, the resulting numbers of variables and constraints are presented in Table 2.3 for formulations (P) , (P_t) and (P_x) . Column “Nb. Var.” (resp. “Nb. Constr.”) indicates the number of variables (resp. constraints). From Table 2.3, we can observe that there is a huge gap (in terms of number of variables and constraints) between the G and the M instances (the number of variables in M1 (the smallest M

Table 2.1: Characteristics of the instances.

Instance	$ O $	$ B $	$ P $	$ C $	Nb. Products	Nb. Boxes	% Boxes	$I^{(sent)}$	$I^{(init)}$
V1	28	166	326	17	377,211	1,323	70.7%	1,680	659
V2	51	192	358	20	417,207	2,175	72.3%	3,285	1,277
V3	49	210	424	21	613,650	2,147	63.4%	2,915	962
V4	59	222	453	20	751,305	2,967	77.0%	3,920	1,821
G1	67	429	1,181	8	1,491,701	5,080	78.0%	4,461	1,283
G2	71	445	1,199	7	1,578,173	5,668	77.2%	4,921	1,250
G3	68	447	1,343	8	1,585,825	6,703	84.0%	5,131	1,301
G4	88	495	1,401	8	1,956,969	6,517	78.3%	5,978	2,027
G5	80	499	1,548	8	2,333,344	8,477	85.2%	6,109	1,830
G6	85	507	1,676	7	2,370,924	8,656	83.6%	6,442	1,409
M1	383	799	6,564	17	20,476,895	51,230	97.3%	78,398	16,417
M2	543	893	7,890	23	21,644,192	58,210	97.2%	89,937	16,244
M3	644	864	7,865	22	24,671,883	68,764	97.0%	105,082	15,946
M4	699	862	7,529	23	21,090,036	68,806	96.8%	109,501	27,048
M5	623	895	8,349	23	24,078,054	67,561	97.0%	106,073	16,661
M6	789	896	8,546	21	30,928,572	84,073	97.2%	130,476	27,328
M7	829	905	8,649	22	35,282,299	93,403	97.0%	142,679	30,716

instance) is more than 10 times bigger than the number of variables in G6 (the largest G instance). Experiments (see Section 4.6) show that for formulation (P), CPLEX is able to solve (within one hour of execution time) all V and G instances but is unable to solve any of the M instances.

Table 2.2: Number of variables in (P) .

Instance	$ O \cdot B \cdot P $	$ X $	$ \tilde{X} $	% non-fixed
V1	1,515,248	668	442	0.029%
V2	3,505,536	883	519	0.015%
V3	4,362,960	1,065	675	0.015%
V4	5,933,394	1,305	883	0.015%
G1	33,945,483	13,758	12,797	0.038%
G2	37,882,405	15,284	14,188	0.037%
G3	40,821,828	18,622	17,507	0.043%
G4	61,027,560	19,618	17,944	0.029%
G5	61,796,160	27,683	26,282	0.043%
G6	72,227,220	28,874	27,603	0.038%
M1	2,008,695,588	420,090	407,608	0.020%
M2	3,825,853,110	518,412	503,904	0.013%
M3	4,376,211,840	601,015	584,093	0.013%
M4	4,536,508,602	542,537	527,444	0.012%
M5	4,655,277,165	651,286	634,152	0.014%
M6	6,041,543,424	736,934	718,116	0.012%
M7	6,488,869,005	833,631	812,775	0.013%

Table 2.3: Sizes of the formulations (P) , (P_t) and (P_x) after the elimination of variables.

Instance	(P)		(P_t)		(P_x)	
	Nb. Var.	Nb. Constr.	Nb. Var.	Nb. Constr.	Nb. Var.	Nb. Constr.
V1	7,683	22,122	913	11,870	5,030	5,274
V2	8,740	30,597	1,090	18,918	5,625	5,809
V3	10,656	36,231	1,400	21,692	6,605	6,863
V4	12,389	42,799	1,826	25,319	7,090	7,337
G1	94,833	249,206	25,662	77,830	18,050	18,993
G2	103,469	272,164	28,448	83,822	18,340	19,285
G3	125,528	325,399	35,083	95,072	20,485	21,586
G4	129,120	346,550	35,977	110,089	21,455	22,534
G5	181,313	471,492	52,645	132,778	23,620	24,878
G6	191,184	495,888	55,292	139,397	25,565	26,931
M1	2,546,024	6,637,545	815,600	1,647,376	100,375	105,437
M2	3,144,490	8,352,285	1,008,352	2,186,514	121,065	126,813
M3	3,625,754	9,617,999	1,168,831	2,490,234	121,195	126,514
M4	3,281,095	8,809,257	1,055,588	2,366,265	116,430	121,193
M5	3,933,263	10,396,752	1,268,928	2,661,040	128,350	134,237
M6	4,440,832	11,796,494	1,437,022	3,050,093	132,135	137,555
M7	5,010,531	13,272,220	1,626,380	3,388,326	133,880	139,243

2.5 Methodology

We now describe two heuristics as well as the computation of lower bounds on f^I . The first heuristic is a greedy algorithm for the non-integrated version of the ECM problem (i.e., each container content is known and cannot be revoked). The second heuristic minimizes f^I or f^W while making decisions on both the loading of containers and on their loading day.

2.5.1 Heuristic for formulation (P_x)

The current practice at ECM complies with the following streamlined Algorithm 1. It starts from a given assignment of products to containers, which is the output of a first optimization step at ECM, and works day by day according to the level of products in inventory. Each day, containers are loaded as soon as the associated products are available, and container loading stops when a workload target is reached. For the subset of containers allowed to be loaded (i.e., for which the required products are in the inventory), various rules for the selection of the first container to load can be applied, which range from random selection to the consideration of specific characteristics of the ILP. Below, we only evaluate random selection, but we discuss the interest of considering specific rules in Section 2.6.4.

2.5.2 Heuristic for large instances

For those instances that are too large to be solved by commercial solvers, we describe in Algorithm 2 a heuristic based on the decomposition (P_t). Each day, based on the available inventory, the shipped volume is maximized. At the end of the day, the demand,

Algorithm 1 Greedy algorithm for (P_x)

Input: Assignment of products to containers (x_{obp}); initial inventory for each product p at the beginning of the week (u_{p0}); inflow product delivery schedule (r_{pt}).

For each day in the week, **do**

While there are containers allowed to be loaded **and** current workload is below average workload, **do**

- (1) Choose a container: select a container from the set of containers allowed to be loaded.
 - (2) Load the selected container: remove the selected container from the list of containers to load and update the resulting inventory in the ILP.
 - (3) Update the set of allowed containers.
-

the available products, and the list of non-loaded containers are updated. The strength of this heuristic lies in the fact that it allows decision makers to load containers based on past and present information only, without using forecasts on future product arrivals. As a result, the output of this algorithm is robust even though some suppliers do not deliver some products on their expected day. The overall computation time is proportional to the length of the planning horizon, since Algorithm 2 works day by day. Furthermore, at each step (i.e., every day) of the algorithm, the number of variables and constraints in (P_t) is smaller than the numbers reported in Table 2.3. Indeed, for the early steps, the cardinality of P becomes smaller since only a subset of products have been received. For the later steps, the cardinality of O also becomes smaller since many containers have been already shipped.

Algorithm 2 focuses on f^I , but it can easily be adapted to tackle f^W . Indeed, in addition to the loading day of each container, Algorithm 2 returns an assignment of products to containers that aims at maximizing the volume of product sent at each step t (day). To smooth the workload, the loading of some containers can simply be delayed. This can be achieved by solving (P_x^W) , with the product-to-box assignment returned by Algorithm 2.

Algorithm 2 Heuristic to minimize the largest inventory

Input: Assignment of boxes to containers (n_{ob}); initial inventory for each product p at the beginning of the week (u_{p0}); inflow product delivery schedule (r_{pt}); client demand (d_{cp}).

For day $t = 1$ to 5, **do**

- (1) Solve (P_t).
 - (2) Fix variables: move products from the inventory to the containers, with respect to (P_t).
 - (2) Update data: remove the loaded containers and update the available inventory and the client demand.
-

2.5.3 Lower bound on f^I

Considering a cumulative inventory for each day (i.e., the sum of all products received), we can compute the largest volume that can be shipped at the end of day $t \in T$. This yields an upper bound on the largest volume in inventory at the end of each day, and therefore a lower bound LB^I on f^I . This bound is formally computed based on the difference between the cumulative volume ($I_t^{(c)} = I^{(init)} + \sum_{t' \leq t} \sum_{p \in P} h_p \cdot r_{pt'}$) of products in stock at the beginning of day t , and the largest volume $f(P_t)$ that can be sent at the end of day t . The value of LB^I is then

$$LB^I = \max_{t \in T} \{I_t^{(c)} - f(P_t)\}. \quad (2.30)$$

The bound LB^I is used to evaluate – a posteriori – the quality of the solutions returned by Algorithm 2. Unfortunately, the computing time required to compute LB^I is of the same order of magnitude as the computing time required to perform Algorithm 2. Indeed, to compute LB^I , we need to solve five times formulation (P_t), which is similar to what is performed in Algorithm 2. If we first compute LB^I , and we next perform Algorithm 2 tightened by LB^I , this will not reduce the overall computing time.

2.6 Computational results

Section 2.6.1 introduces some notation needed to understand our numerical experiments. Section 2.6.2 compares, qualitatively, the different optimization approaches that can be applied to the ECM problem, which vary with respect to their degree of integration. Section 2.6.3 details the values of the optimal solutions for the V and G instances, which have a tractable size for CPLEX (see Table 2.3 for the size of the formulation after eliminating variables). Moreover, the output of Algorithm 2 is compared to these optimal values. Section 2.6.4 compares our results with the ECM current practice on all instances.

The formulations and all the heuristics were coded in C++. The solver is CPLEX 12.4 and is called with the concert technology. Computations were launched on a 2.2 GHz Intel Core i7 with 16 Go 1600 MHz DDR3 of RAM memory.

2.6.1 Notation

We use the formalism $f_{(ind)}(Form)$. The index *ind* indicates the solution method. More precisely, $ind = h$ if our *heuristic* (i.e., Algorithm 2) is applied, whereas $ind = g$ if the current-practice *greedy* heuristic (i.e., Algorithm 1) is employed. *Form* indicates the formulation (see Section 2.3.4), among (P) , (P^I) , (P^{IW}) , (P^{WI}) , (P_x^I) and (P_x^W) . Regarding the formulation, the exponent gives the considered objective and the index indicates the variables that are fixed. Hence, (P_x^I) (resp. (P_x^W)) denotes the formulation in which the container content is fixed and for which f^I (resp. f^W) is the only objective minimized. In both (P_x^I) and (P_x^W) , the assignment of products to containers is fixed as in the ECM current solution. $f^*(Form)$ refers to the optimal solution of formulation *Form*. Tables (2.4–2.6) provide the execution times in minutes. Gaps are expressed in percent. The percentage gap between $f_{(h)}(P^I)$ and $f^*(P^I)$ is denoted as $\%f^*(P^I)$, and is computed as $\frac{f_{(h)}(P^I) - f^*(P^I)}{f^*(P^I)} \cdot 100$.

2.6.2 Comparison of the various optimization approaches

For objective f^I and for any instance, Figure 2.3 shows the expected ranking of the values LB^I , $f^*(P^I)$, $f_{(h)}(P^I)$, $f^*(P_x^I)$ and $f_{(g)}(P_x^I)$. We use a uniform step size between each pair of values, since initially we have no quantitative insight. The grey and black rectangles highlight the benefits of the main approaches. The rectangle “Non-integrated method” indicates the range of values that can be obtained when the container content is fixed: $f^*(P_x^I)$ is the value of the optimal non-integrated solution. The rectangle “Integrating loading constraints” covers the solution values that can be reached in the case of a full, accurate but cumbersome, integration of the loading constraints to the container scheduling problem. The rectangle “Revoking products to boxes” shows where our results are expected to lie: $f_{(h)}(P^I)$ sets for the best-known solution value. More precisely, the difference between $f^*(P_x^I)$ and $f^*(P^I)$ corresponds to the largest achievable gain ensuring the non-violation of the loading constraints. Depending on the rule used to select the containers to be loaded each day, Algorithm 1 can be more or less efficient. Because the ECM practice of selecting the containers involves an experience-based understanding of the ILP that is hard to replicate, we use a random container selection for an estimation of the ECM results. Therefore, any current-practice solution value lies between $f^*(P_x^I)$ and $f_{(g)}(P_x^I)$ (see the rectangle “Current practice”). The actual size of each rectangle of Figure 2.3 will be discussed in Section 2.6.4, relying on Figure 2.4.

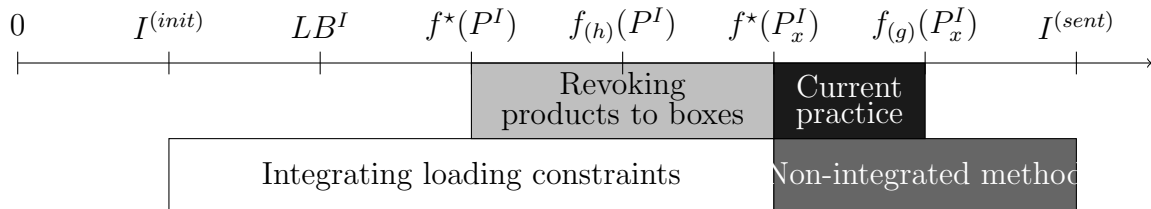


Figure 2.3: Comparison of the expected f^I -gains of the various approaches.

2.6.3 Optimal results for the V and G instances

Table 2.1 indicates that instances V and G are at least 10 times smaller than instances M. These much smaller sizes mean that these instances can be solved optimally by CPLEX with formulation (P) . While in these two cases f^W is the main objective, Table 2.4 gives the optimal values for both f^I and f^W , and presents a comparison with Algorithm 2 for both objectives f^I and f^W . The value of the second objective is also provided in columns “ $f^*(P^I|W)$ ” and “ $f^*(P^W|I)$ ”.

Algorithm 2 returns an optimal solution in 17 of the 20 cases. Considering f^I , we obtain a low gain when the loading of a container allowed to be loaded on a given day is postponed by one day. In other words, it is preferable to load the containers as soon as possible. In addition, the objectives f^I and f^W may conflict. It may indeed be preferable not to load the containers early and to spread the work over the less busy following days. For example, in instance G4, the optimal value $f^*(P^W)$ for the largest workload imbalance is 2,341 m³ and the associated required largest storage space is 3,827 m³. When decreasing the storage space to its optimal value $f^*(P^I)$ (i.e., 3,745 m³), the difference between the most and least busy days ($f^*(P^W|I)$) increases to 2,699 m³. Since the workload also takes into account the unloading of inbound trucks, if many of them are unloaded during a specific day, it is preferable to delay some container loadings in order to smooth the workload.

2.6.4 Comparison with current practice

In this section, we compare the best-known solution values for (P^W) and (P^I) with the output of Algorithm 1 and with the optimal solution values of (P_x^W) and (P_x^I) (i.e., an estimate of the values observed in practice). Only those results concerning the main objective are considered for the comparison (i.e., f^W for instances V and G, f^I for instances

Table 2.4: Optimal results for the V and G instances, and performance of Algorithm 2.

Instance	Formulation (P^W)			Formulation (P^I)			Algorithm 2				
	$f^*(P^W)$	$f^*(P^{IW})$	Time	$f^*(P^I)$	$f^*(P^{WI})$	Time	$f_{(h)}(P^W)$	Time	$\%f^*(P^W)$	$f_{(h)}(P^I)$	$\%f^*(P^I)$
V1	893	1,260	< 1	981	1,237	< 1	893	< 1	0.0%	982	0.1%
V2	1,579	1,922	< 1	1,246	1,960	< 1	1,579	< 1	0.0%	1,246	0.0%
V3	1,669	2,060	< 1	1,546	1,976	< 1	1,669	< 1	0.0%	1,546	0.0%
V4	2,233	2,715	< 1	1,894	2,690	< 1	2,233	< 1	0.0%	1,894	0.0%
G1	2,756	2,679	1	2,679	2,756	10	2,756	< 1	0.0%	2,679	0.0%
G2	1,508	3,182	13	3,076	1,718	6	1,508	< 1	0.0%	3,076	0.0%
G3	3,357	3,642	< 1	2,977	4,022	5	3,357	< 1	0.0%	2,977	0.0%
G4	2,341	3,827	14	3,745	2,699	12	2,479	< 1	5.8%	3,745	0.0%
G5	3,060	4,061	13	3,966	4,118	9	3,060	< 1	0.0%	3,968	0.1%
G6	3,204	4,603	13	4,065	3,345	1	3,204	< 1	0.0%	4,065	0.0%

M), since no additional insight can be gained from the conflict between the two objectives.

f^W -values for instances V and G

Table 2.5 compares the optimal solution values $f^*(P^W)$ for formulation (P^W), the optimal solution values $f^*(P_x^W)$ for formulation (P_x^W) (when the container contents are fixed by ECM), and the output $f_{(g)}(P^W)$ of Algorithm 1 (which estimates the ECM results). The gaps between the different formulations with respect to $f^*(P^W)$ and $f^*(P_x^W)$ are expressed in the columns “%...”. We observe that even if Algorithm 1 is fast (it requires less than a second of execution time), it delivers results on formulation (P_x^W) for which there is substantial room for further improvement. When the containers content is fixed, the percentage gap between the values returned by Algorithm 1 and the optimal values is on average of 16.8% for the V instances and of 12.6% for the G instances. Algorithm 1 could be further improved (with respect to $f^*(P_x^W)$) with the use of more refined rules for selecting the containers to load. This has not been investigated since the optimal solution values for formulation P_x^W are obtained within less than a minute with CPLEX. Table 2.5 also shows that a substantial gain can be achieved by integrating the loading decisions in the container scheduling problem (see column “ $\%f^*(P^W)$ ” under “Formulation (P_x^W)”). Indeed, the average percentage gap between (1) the optimal solution values of (P_x^W) using

the ECM current assignment of products to containers, and (2) the optimal solution values of (P^W) , is on average 6.1% for the V instances and 33.6% for the G instances. More generally, we observe that the average gain becomes larger as the instance size increases. Indeed, the G instances involve a volume of handled products that is on average five times larger than for the V instances.

Table 2.5: Results for the V and G instances (i.e., focusing on f^W).

Instance	Formulation (P^W)		Formulation (P_x^W)			Current practice (Algorithm 1)			
	$f^*(P^W)$	Time	$f^*(P_x^W)$	Time	$\%f^*(P^W)$	$f_{(g)}(P_x^W)$	Time	$\%f^*(P_x^W)$	$\%f^*(P^W)$
V1	893	< 1	893	< 1	0.0%	995	< 1	11.4%	11.4%
V2	1,579	< 1	1,671	< 1	5.8%	2,028	< 1	21.4%	28.4%
V3	1,669	< 1	1,758	< 1	5.3%	2,055	< 1	16.9%	23.1%
V4	2,233	< 1	2,445	< 1	9.5%	2,829	< 1	15.7%	26.7%
G1	2,756	1	3,407	< 1	23.6%	3,650	< 1	7.1%	32.4%
G2	1,508	13	2,122	< 1	40.7%	3,313	< 1	56.1%	119.7%
G3	3,357	< 1	4,397	< 1	31.0%	4,940	< 1	12.3%	47.2%
G4	2,341	14	3,125	< 1	33.5%	3,717	< 1	18.9%	58.8%
G5	3,060	13	4,152	< 1	35.7%	4,241	< 1	2.1%	38.6%
G6	3,204	13	4,469	< 1	39.5%	4,542	< 1	1.6%	41.8%

f^I -values for instances M

Table 2.6 compares three approaches for minimizing f^I , which is the main objective for the M instances: the output $f_{(h)}(P^I)$ of Algorithm 2, the optimal results using the ECM current assignment $f^*(P_x^I)$ of products to containers, and the output $f_{(g)}(P^I)$ of Algorithm 1. The lower bound LB^I is also provided for each instance.

We observe from Table 2.6 that Algorithm 1 is very efficient for formulation (P_x^I) . Indeed, its optimality gap never exceeds 2%. In other words, given a feasible assignment of products to containers, only marginal gains can be achieved by using CPLEX to minimize the storage in the ILP. As mentioned in Section 2.5.1, Algorithm 1 randomly selects the next container to load (see step (1) of Algorithm 1). Therefore, using more refined selecting rules at step (1) could only yield marginal gains (between 0.2% and 1.9%). We observe that for (P_x^I) , which is a more difficult problem than MKP (see Section 2.3.6),

both CPLEX with an execution time of less than a second, and Algorithm 1, with an optimality gap below 2%, exhibit good performances. This can be explained by the fact that the instances have a significant proportion (between 50% and 70%) of the product types that can only be assigned to a single container. Therefore, the solution space of these instances remains of tractable size.

As shown in Table 2.6, revoking the assignment of products to containers can lead to substantial gains. Column “ $\%f_{(h)}(P^I)$ ” under “Formulation (P_x^I)” displays the gain found by revoking the loading of containers (compared to the optimal solutions with the ECM current assignment of products to containers). It lies between 26.6% and 72.5%. The average gap between the best-known solution value and the best non-integrated solution value is 46.8%. Additionally, the lower bounds presented in column “ LB^I ” indicate that the output of Algorithm 2 is close to optimality, except for one instance for which the optimality gap is 18.8%. We conclude that an algorithm that would consider future information to schedule the containers (i.e., the inbound flows during the next days) could only yield a marginal gain.

Table 2.6: Results for the M instances (i.e., focusing on f^I).

Instance	Algorithm 2 (P^I)				Formulation (P_x^I)			Current practice (Algorithm 1)			
	LB^I	$f_{(h)}(P^I)$	Time	$\%LB^I$	$f^*(P_x^I)$	Time	$\%f_{(h)}(P^I)$	$f_{(g)}(P_x^I)$	Time	$\%f^*(P_x^I)$	$\%f_{(h)}(P^I)$
M1	33,653	36,162	16	6.9%	62,399	< 1	72.5%	62,829	< 1	0.7%	73.7%
M2	42,980	44,465	29	3.3%	59,049	< 1	32.8%	59,314	< 1	0.4%	33.4%
M3	54,897	55,386	30	0.9%	75,727	< 1	36.7%	75,875	< 1	0.2%	37.0%
M4	49,313	50,751	32	2.8%	72,109	< 1	42.1%	72,296	< 1	0.3%	42.5%
M5	56,605	57,889	35	2.2%	73,294	< 1	26.6%	73,461	< 1	0.2%	26.9%
M6	53,614	56,191	41	4.6%	93,335	< 1	66.1%	93,843	< 1	0.5%	67.0%
M7	51,415	63,293	45	18.8%	98,630	< 1	55.8%	100,540	< 1	1.9%	58.8%

Figure 2.4 quantifies the qualitative aspects displayed in Figure 2.3 with the average values computed on all M instances. It illustrates that the current practice offers very few improvement opportunities when considering the used non-integrated approach (the rectangle “Current practice” is small compared with the rectangle “Revoking product to boxes”). It also shows that the potential improvement on our heuristic is small since the gap to the lower bound LB^I is on average of 6.3% over all instances. Finally, among the

97% of boxes for which a change of product assignment is possible (average over all M instances, see Table 2.1), our solutions yield a box content that is different from the one proposed by ECM for 64.8% of the boxes. In other words, our optimization results yield very different container loading plans while satisfying all loading constraints.

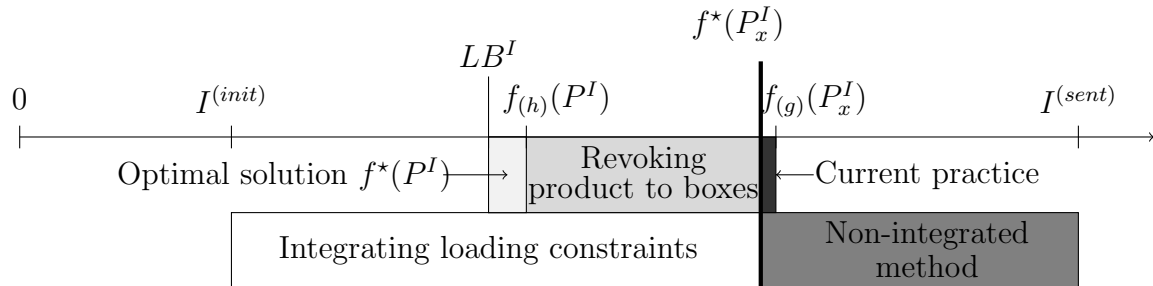


Figure 2.4: Quantification of the expected f^I -gains for various approaches (average values for all M instances).

2.7 Conclusions

We have modeled and solved a scheduling of outbound flows in an ILP under complex loading constraints. This problem was proposed by ECM and is encountered in three of its ILPs. The aim was to compute an improved schedule for the loading day of each container to either reduce the largest required inventory space or the weekly workload imbalance.

To solve the problem, we have developed both an exact algorithm and a heuristic. Whereas the exact approach is able to solve to optimality two thirds of the provided industrial instances, the heuristic can efficiently tackle the remaining larger instances. Matching the industrial requirements concerning the execution time (i.e., less than one hour), the heuristic returns results with an optimality gap lower than 7% for 85% of the large instances. Furthermore, the heuristic allows handling the uncertainty in the inbound flows. Indeed, the heuristic solves independently the container loading and scheduling problems

for each day. Therefore, the decisions are based on product arrivals for a given day and can be adapted to any delivery of products (e.g., if the suppliers did not deliver the exact amount of products, the schedule can be easily adapted by relaunching the heuristic). Last, the results show that compared with current practice, our heuristic yields a 26% to 73% improvement in the inventory level, with an average of 46.8%. Similarly, an average improvement of 25.8% was found for the smoothing of the workload. In other words, substantial gains can be achieved for both objectives.

From a managerial point of view, the efficiency gains for the ILPs are achievable without involving third-parties, since neither the suppliers nor the clients of the ILP are impacted by the considered decisions (i.e., each product arrival day at the ILP remains the same and all the client demands are covered at the end of the week). We have shown that optimizing only the container loading days yields marginal gains. Indeed, if the container contents remain unchanged, a gain not exceeding 2% can be achieved on the largest required inventory space. In contrast, adapting the container contents based on the inbound deliveries, while optimizing the loading day of the containers, helps to significantly improve the objective value.

As a future avenue of research, we mention the integration of additional types of decisions. For instance, one may also determine the arrival schedule of the inflow trucks and the allocation of resources to the container loading platforms.

Acknowledgement

This work was partly supported by the Canadian Natural Sciences and Engineering Research Council under grant 2015-06189. This support is gratefully acknowledged. Thanks are due to the reviewers for their valuable comments.

Chapter 3

Inbound and Outbound Flow Integration for Cross-Docking Operations

MARC-ANTOINE COINDREAU - *University of Lausanne, Switzerland*

OLIVIER GALLAY - *University of Lausanne, Switzerland*

NICOLAS ZUFFEREY - *University of Geneva, Switzerland*

GILBERT LAPORTE - *HEC Montréal, Canada*

Abstract

We consider the optimization of the cross-docking operations at three intermodal logistics platforms (ILPs) of a large European car manufacturer (ECM). The planning horizon is a week and the time bucket is a day. An inbound flow of products is gradually received over the week by truck from inland suppliers, and has to be loaded into containers which are then shipped to offshore production plants. The full content of a container must be available at the ILP to enable its loading operations to start, hence temporary storage is needed. The objective is to minimize an inventory penalty, computed as the largest daily volume of temporary product storage observed over the planning horizon. The current practice at ECM is to first optimize the content of the inbound trucks and of the outbound containers independently, and then determine the loading day of each container to be shipped based on these fixed contents. We propose to integrate, within the same optimization framework, the decisions on both truck and container contents, which involve complex loading constraints related to the dimensions and weights of the products, with those on the scheduling of container loading. We model the resulting problem as a mixed integer linear program, and we develop a decomposition scheme for it, as well as a fix-and-optimize matheuristic. We perform extensive computational experiments on real instances provided by ECM. Results show that a combination of these two matheuristics is able to generate solutions that reduce the average inventory penalty by 40%.

Keywords: Logistics, cross-dock scheduling, matheuristic, fix-and-optimize.

3.1 Introduction

We model and solve an operations management problem encountered by a large European car manufacturer (denoted here as ECM as a result of a non-disclosure agreement) which consolidates product flows from inland suppliers to offshore production plants at inter-modal logistics platforms (ILPs). Over a given planning horizon (from Monday to Friday in this work), the products, which are collected by trucks at different supplier locations, are first unloaded and repacked at the ILP. The products are then immediately loaded into containers, or temporarily stored until a full container content is available at the ILP, hence allowing the loading operations to be launched. It is assumed that the necessary products for all container contents are received by truck over the week, hence allowing all planned container loading operations to take place. The containers are finally sent by ship at the end of the week to offshore production plants, which are the ILP clients. We refer to this problem as the ECM Problem. Figure 3.1 illustrates the sequence of operations just described.

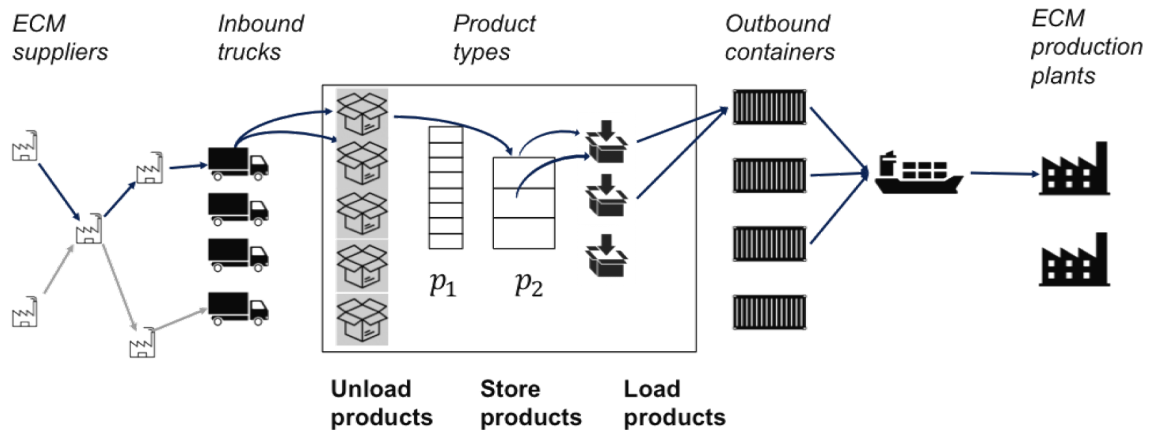


Figure 3.1: Product flow in an ILP.

Inbound truck transportation is subcontracted. As a consequence, the truck routes cannot be modified, as they are contractually fixed for the long term. The complex routing subproblems associated with the inbound trucks have been previously and independently solved by ECM. They can typically be modeled as a Traveling Purchaser Problem [Boctor et al., 2003]. However, the ILP managers can still decide which products should

be collected on the truck routes.

Regarding the outbound side, a container can only be loaded after its full content has been delivered to the ILP. This entails temporary storage, which generates inventory costs at the ILP. Furthermore, high inventory levels may lead to an imbalanced workload since the stored products will ultimately have to be loaded into containers during the last days of the week. Therefore, ECM aims at minimizing, over a one-week planning horizon a penalty computed as the largest daily inventory volume required at the ILP. To this end, three different types of decisions are inherent to the ECM problem: determining the contents of the trucks, that of the containers, and the loading day of each container.

By the end of the week, the demand of each client, i.e., the requested quantity of each product type, must be satisfied and loaded in its assigned containers. The products arriving on the inbound side can be sent to any outbound client requesting them. Since the containers are all sent by boat at the end of the week, the container loading sequence is unconstrained. Additionally, the number of containers loaded per day is unlimited.

The truck and container loading problems are rather complicated since they involve three-dimensional constraints (the three dimensions of the products are taken into account when loading, and overlaying is forbidden), a total weight limitation (the total weight of all products loaded in the same container cannot exceed 20 tonnes for the trucks and 22 tonnes for the containers), and specific arrangements of packaged products in stacks. In the latter case, the weight of the product restricts its position in the stack (e.g., heavy products cannot be loaded above lighter ones). For an overview of common loading constraints, see [Toffolo et al., 2017].

Because of such complex loading constraints, reassigning a product from a truck or a container to another one is not straightforward. Yet the problem is slightly simplified since the products are packaged into standardized boxes, and the loading constraints

concern the box types only, irrespective of the products they contain. The complete consideration of the set of loading constraints is extremely complex, hence we restrict the solution space to product permutations between boxes of the same type. While this simplification allows to control the size of the problem, it still gives rise to a rich solution space to explore. Indeed, it has been observed that more than 70% of the boxes are filled with different products but with a weight variation of less than 10 kg, which precludes any violation of the weight of a stack. Hence, starting from a feasible assignment of boxes to trucks or containers, numerous different assignments of products to boxes are possible. As a consequence, product permutations between boxes can be applied to optimize the truck or the container contents, while ensuring that the loading constraints will be satisfied.

The current practice at ECM is to determine the loading day of each container over the planning horizon, without revoking previously made decisions. Indeed, in a first phase, ECM uses standalone optimization tools to independently determine the truck and container contents. These contents are taken as inputs to a second phase dedicated to the scheduling of container loading operations. A particular case of the ECM problem was investigated in [Coindreau et al., 2019b], in which the truck contents are fixed and the decisions focus only on the loading day and the content of the containers. Even if substantial reductions were already observed for the inventory penalty, our work aims at extending the previous study by further integrating the decisions on the truck contents.

This paper makes the following scientific contributions. We introduce the ECM problem which integrates the optimization of both inbound and outbound product flows with container scheduling. To solve the ECM problem, we propose a mixed integer linear programming (MILP) model as well as two matheuristics, namely a decomposition matheuristic (DM) and a fix-and-optimize matheuristic (FOM). We perform extensive computational tests on the instances provided by ECM and we quantify the inventory penalty reduction resulting from our integrated approach.

The remainder of this paper is organized as follows. Section 3.2 provides a survey of the related literature. Section 3.3 introduces the MILP formulation of the ECM problem. Section 3.4 presents our two matheuristics. Section 3.5 compares the performance of the proposed solution methods and quantifies the gain achieved by our integrated approach with respect to the non-integrated current practice. This is followed by conclusions and perspectives in Section 3.6.

3.2 Literature Review

We first review the cross-docking literature that shares some similarities with the ECM problem. Next, we give an overview of matheuristics that are relevant for the present case, and we focus in particular on FOMs.

A cross-docking facility aims at consolidating inbound and outbound flows (here, from inland suppliers to offshore production plants) by making, as much as possible, direct product transfers from trucks to containers [Van Belle et al., 2012]. More specifically, the ECM problem shares some features of the truck scheduling in a cross-dock (TSCD), for which a review can be found in [Boysen and Fliedner, 2010]. In the TSCD, the unloading and loading operations of the trucks are viewed as a set of jobs, as defined in the job scheduling literature. The aim of the TSCD is to determine a sequence of inbound trucks arriving at the cross-docking platform and a sequence of outbound containers that are then loaded, in order to minimize a given objective, e.g., the makespan, as in [Chen and Lee, 2009] and [Ye et al., 2018]). Whereas in some cases, the product transfers can be done without any need for temporary product storage [Boysen, 2010], other situations require a temporary inventory (as for the ECM problem, momentary storage is observed in [Yu and Egbelu, 2008]). Despite its similarities with cross-docking, the ECM configuration precludes the use of the existing related methodologies. First, direct transfers of products from trucks to containers cannot always take place in the ILP due

to the constraints imposed on the scheduling of operations related to container loading. Indeed, it is required that the whole content of a container be available at the ILP before proceeding to its loading. This creates an increased need for temporary storage that is rarely observed in standard cross-docking configurations. Second, and in contrast with the TSCD, the content of the trucks and that of the containers are modified during the optimization of the operations, and hence jobs can no longer be defined by a set of products to be unloaded and then loaded, as is done, for example, in [Bellanger et al., 2013]. Focusing solely on the inbound side, [Serrano et al., 2017] consider the reassignment of the content of inbound trucks in a container scheduling context. In contrast with the ECM problem, a simplifying assumption is made by considering scalar loading constraints. To the best of our knowledge, no existing work provides loading solutions that ensure the non-violation of the complex loading constraints considered here and described in [Toffolo et al., 2017]. We refer to [Coindreau et al., 2019b] for a more extensive review of related cross-docking problems.

In a recent paper, [Coindreau et al., 2019b] have proposed a decomposition matheuristic for a subcase of the ECM problem. In a similar context of warehouse management, [Cattaruzza et al., 2018] also show that a decomposition matheuristic is efficient, and the problem is iteratively solved by fixing some variables. We present below suitable types of matheuristics that can help tackle the further complexity brought by the integration of decisions for the truck content, in addition to considering those for the container content and container scheduling. Matheuristics typically combine mathematical programming and heuristics [Jourdan et al., 2009]. Among the wide existing range of available matheuristics, FOMs (originally introduced by [Gintner et al., 2005]) consist in iteratively fixing a subset of decision variables to create a smaller MILP that can be solved with a generic solver. Repeatedly fixing some variables and optimizing some others often allows to outperform the direct use of a solver applied to the full set of variables. In particular, FOM has been successfully applied to lot sizing [Sahling et al., 2009, Helber and Sahling, 2010], timetabling [Dorneles et al., 2014], and location-routing prob-

lems [Rieck et al., 2014]. Recently, FOM has been combined with generic metaheuristic frameworks such as variable neighborhood search (VNS) in [Della Croce and Salassa, 2014] and in [Chen, 2015], or variable neighborhood descent (VND) in [Dorneles et al., 2014]. These papers indicate that combining FOM with a metaheuristic outperforms the use of FOM only.

3.3 Mathematical formulation

Section 3.3.1 introduces the variables and sets related to the ECM problem. Section 3.3.2 presents a MILP model for minimizing the proposed inventory penalty (i.e., the largest temporary storage required over the week) by making decisions on the content of a given subset of trucks and containers. Section 3.3.3 describes specific configurations of the MILP that are relevant for the ECM problem.

3.3.1 Sets, parameters and variables

The superscripts “*in*” and “*out*” refer to inbound and outbound, respectively. Furthermore, “*nf*” and “*f*” refer to the set of trucks or containers for which the content is not fixed and fixed, respectively.

Sets:

- T : set of time periods (i.e., days),
- C : set of clients,
- P : set of product types,

- S : set of suppliers,
- B : set of box types,
- I : set of inbound trucks, which contains the following subsets:
 - $I^{(nf)}$: subset of trucks for which the content is not fixed and can therefore be optimized with the MILP,
 - $I^{(f)}$: subset of trucks for which the content is fixed,
 - I_t : subset of trucks that arrive on day $t \in T$,
 - $I_t^{(nf)}$: subset of trucks that arrive on day $t \in T$, for which the content is not fixed,
 - $I_t^{(f)}$: subset of trucks that arrive on day $t \in T$, for which the content is fixed,
- O : set of outbound containers, which contains the following subsets:
 - $O^{(nf)}$: subset of containers for which the content is not fixed and can therefore be optimized with the MILP,
 - $O^{(f)}$: subset of containers for which the content is fixed,
 - O_c : subset of containers assigned to client $c \in C$,
 - $O_c^{(f)}$: subset of containers assigned to client $c \in C$, for which the content is fixed,
 - $O_c^{(nf)}$: subset of containers assigned to client $c \in C$, for which the content is not fixed.

Parameters:

- d_{cp} : demand (in units) of client $c \in C$ for product type $p \in P$,
- $n_{ib}^{(in)}$: number of units of boxes of type $b \in B$ transported in truck $i \in I$,

- $n_{ob}^{(out)}$: number of units of boxes of type $b \in B$ transported in container $o \in O$,
- $\pi_{pi} = 1$ if truck $i \in I$ visits the supplier that can provide product type $p \in P$,
 $\pi_{pi} = 0$ otherwise,
- q_{pb} : number of units of product type $p \in P$ that can be transported in box type $b \in B$,
- $q_{op}^{(out)}$: number of products of type $p \in P$ sent by container $o \in O^{(f)}$,
- $q_{ip}^{(in)}$: number of products of type $p \in P$ delivered by truck $i \in I^{(f)}$,
- l_{pb} : weight (in kg) of a box of type $b \in B$ when filled with product type $p \in P$,
- $l^{(in)}$: maximum allowed weight (in kg) that can be transported by a truck,
- $l^{(out)}$: maximum allowed weight (in kg) that can be transported by a container,
- h_p : volume (in m³) of a product of type $p \in P$,
- g_p : number of units of product type $p \in P$ available in the inventory at the beginning of the week; due to various reasons (e.g., lot sizing or wrong orders), there is an initial inventory in the ILP that cannot be determined in the optimization process and is therefore taken as an input (the magnitude of this initial inventory is detailed later),
- M_{op} : largest amount of products of type $p \in P$ that can be transported in container $o \in O$.

Decision variables:

- z_{ibp} : number of boxes of type $b \in B$ assigned to product type $p \in P$ in truck $i \in I^{(nf)}$,
- x_{obp} : number of boxes of type $b \in B$ assigned to product type $p \in P$ in container $o \in O^{(nf)}$,

- $y_{ot} = 1$ if container $o \in O$ is loaded on day $t \in T$; $y_{ot} = 0$ otherwise,
- w_{opt} : number of units of product type $p \in P$ sent by container $o \in O^{(nf)}$ on day $t \in T$,
- u_{pt} : number of units of product type $p \in P$ in stock on day $t \in T$ before loading the containers,
- v_{pt} : number of units of product type $p \in P$ in stock on day $t \in T$ after loading the containers,
- r_{pt} : number of units of product type $p \in P$ received on day $t \in T$,
- s_{pt} : number of units of product type $p \in P$ sent on day $t \in T$,
- f : largest inventory-penalty value (in m^3) encountered during the planning horizon.

3.3.2 Mixed integer linear programming formulation: $Q(O^{(nf)}, I^{(nf)})$

We denote by $Q(O^{(nf)}, I^{(nf)})$ the MILP formulation of the ECM problem for which the content of the $I^{(nf)}$ trucks and the $O^{(nf)}$ containers can be revoked and optimized. The problem is stated as follows:

$$\text{minimize } f \tag{3.1}$$

subject to

$$f \geq \sum_{p \in P} h_p \cdot v_{pt} \quad t \in T \quad (3.2)$$

$$v_{pt} = u_{pt} - s_{pt} \quad p \in P, t \in T \quad (3.3)$$

$$v_{p0} = g_p \quad p \in P \quad (3.4)$$

$$u_{pt} = v_{p,t-1} + r_{pt} \quad p \in P, t \in T \quad (3.5)$$

$$r_{pt} = \sum_{b \in B} \sum_{(i \in I_t^{(nf)} | \pi_{pi} > 0)} q_{pb} \cdot z_{ibp} + \sum_{i \in I_t^{(f)}} q_{ip}^{(in)} \quad p \in P, t \in T \quad (3.6)$$

$$s_{pt} = \sum_{o \in O^{(nf)}} w_{opt} + \sum_{o \in O^{(f)}} q_{op}^{(out)} \cdot y_{ot} \quad p \in P, t \in T \quad (3.7)$$

$$\sum_{t \in T} y_{ot} = 1 \quad o \in O \quad (3.8)$$

$$w_{opt} \leq M_{op} \cdot y_{ot} \quad t \in T, o \in O, p \in P \quad (3.9)$$

$$w_{opt} \leq \sum_{b \in B} q_{pb} \cdot x_{obp} \quad t \in T, o \in O, p \in P \quad (3.10)$$

$$\sum_{o \in O_c^{(nf)}} \sum_{t \in T} w_{opt} \geq d_{cp} - \sum_{o \in O_c^{(f)}} q_{op}^{(out)} \quad c \in C, p \in P \quad (3.11)$$

$$\sum_{b \in B} \sum_{p \in P} l_{pb} \cdot x_{obp} \leq l^{(out)} \quad o \in O^{(nf)} \quad (3.12)$$

$$\sum_{p \in P} x_{obp} \leq n_{ob}^{(out)} \quad o \in O^{(nf)}, b \in B \quad (3.13)$$

$$\sum_{p \in P | \pi_{pi} > 0} z_{ibp} \leq n_{ib}^{(in)} \quad i \in I^{(nf)}, b \in B \quad (3.14)$$

$$\sum_{b \in B} \sum_{p \in P} l_{pb} \cdot z_{ibp} \leq l^{(in)} \quad i \in I. \quad (3.15)$$

$$z_{ibp}, x_{obp}, w_{opt}, u_{pt}, v_{pt}, r_{pt}, s_{pt} \in \mathbb{N} \quad (3.16)$$

$$y_{ot} \in \{0, 1\} \quad (3.17)$$

$$f \in \mathbb{R} \quad (3.18)$$

Constraints (3.2) compute the largest amount of storage space required in the ILP. Constraints (3.3) (resp. (3.5)) compute the available inventory in the ILP at the end (resp. at the beginning) of the day. Constraints (3.4) fix the initial inventory in the ILP at the beginning of the planning horizon (i.e., the products that are not received during the

week are assumed to be in inventory at the beginning of the week). Constraints (3.6) compute the amount of products received on each day at the ILP. Constraints (3.7) compute the number of units of each product type sent on each day. Constraints (3.8) prevent a container from being loaded multiple times. Constraints (3.9) impose that products are sent on the loading day of a container. Constraints (3.10) limit the amount of products sent by containers. Constraints (3.11) impose that the demand of each client is satisfied. Constraints (3.12) and (3.13) (resp. (3.14) and (3.15)) define the loading constraints of the containers (resp. of the trucks). More precisely, constraints (3.12) (resp. (3.15)) ensure that the weight of the transported products does not exceed the container (resp. truck) capacity, and constraints (3.13) (resp. (3.14)) ensure that the number of boxes transported in a container (resp. truck) does not exceed the allowed limit. Constraints (3.16 – 3.18) give the domain of the variables.

3.3.3 Specific configurations of $Q(O^{(nf)}, I^{(nf)})$

The following specific configurations are introduced.

- Q : configuration where the full content of both containers and trucks is optimized (i.e., $O^{(nf)} = O$ and $I^{(nf)} = I$),
- Q_z : configuration where only the content of all the containers is optimized (i.e., the z_{ibp} variables are fixed: $O^{(nf)} = O$ and $I^{(nf)} = \emptyset$),
- Q_x : configuration where only the content of all the trucks is optimized (i.e., the x_{obp} variables are fixed: $O^{(nf)} = \emptyset$ and $I^{(nf)} = I$),
- $Q_z(O^{(nf)})$: configuration where all truck contents are fixed (i.e., $I^{(nf)} = \emptyset$) and the content of a subset of containers ($O^{(nf)}$) is optimized,
- $Q_x(I^{(nf)})$: configuration where all container contents are fixed (i.e., $O^{(nf)} = \emptyset$) and the content of a subset of trucks ($I^{(nf)}$) is optimized,

- $Q_{x,z}$: configuration where the decision making only focuses on the loading day of the containers (i.e., $O^{(nf)} = \emptyset$ and $I^{(nf)} = \emptyset$); it corresponds to current practice at ECM, according to which the content of the trucks and the containers is built in a pre-processing phase using two independent optimization tools, and $Q_{x,z}$ is then solved “by hand” (i.e., in a constructive fashion) by the decision maker.

Configurations Q_z and $Q_{x,z}$ have been considered in [Coindreau et al., 2019b]. Furthermore, as considered in [Coindreau et al., 2019b], configuration $Q_{z,t}$ (resp. $Q_{x,z,t}$) stands for the decomposition of Q_z (resp. $Q_{x,z}$) that aims at maximizing the volume of products sent at the end of day t when the content of the trucks (resp. the content of both trucks and containers) is fixed. It has been shown in [Coindreau et al., 2019b] that $Q_{x,z,t}$ is equivalent to the multiple knapsack problem, which is known to be \mathcal{NP} -hard [Puchinger et al., 2010].

Table 3.1 summarizes the above configurations. For each configuration, the decision sets and the fixed variables are given. “ \times ” indicates that the corresponding decision variables are taken into account. In the decompositions aimed at optimizing the volume shipped when fixing the loading day of the containers, the decisions concerning the loading operations of the containers are made on a subset of containers. Accordingly, “ (\times) ” means that the decision variables are partially taken into account. The first four configurations have been studied in [Coindreau et al., 2019b] and correspond to the situations where the content of the trucks is fixed.

Table 3.1: Considered configurations of the ECM problem.

Configuration	Fixed variables	Decision variables		
		Truck content	Container content	Loading day
Q_z	Truck content		\times	\times
$Q_{z,t}$	Truck content, loading day		\times	(\times)
$Q_{x,z}$	Truck and container content			\times
$Q_{x,z,t}$	Truck and container content, loading day			(\times)
Q	-	\times	\times	\times
Q_x	Container content	\times		\times

3.4 Matheuristics

Since neither Q nor Q_z cannot be solved with CPLEX for the largest instances provided by ECM, we propose two matheuristics capable of handling large and complex cases. First, we introduce DM to solve configuration Q in Section 3.4.1. Section 3.4.2 details FOM, which aims at solving multiple times $Q(O^{(nf)}, I^{(nf)})$ with different selections of trucks and containers to be optimized. Section 3.4.3 proposes a matheuristic based on a combination of DM and FOM. Finally, Section 3.4.4 highlights additional advantages for ECM to favor FOM over alternative solution methods.

3.4.1 Decomposition matheuristic (DM)

As discussed by [Archetti and Speranza, 2014], the key idea behind a decomposition matheuristic is to divide the main problem into smaller subproblems that are easier to solve. Each subproblem is then solved by mathematical programming.

To solve Q , we propose to sequentially optimize the content of the containers and then the content of the trucks (i.e., solve Q_z and then Q_x). [Coindreau et al., 2019b] introduced a temporal decomposition matheuristic (called TDM) to solve Q_z . Each day, the container contents are reorganized so as to maximize the sent volume of products that is shipped. In other words, the configuration $Q_{z,t}$ is solved from $t = 1$ to 5.

It turns out that Q_x is easier to solve than Q_z . Indeed, CPLEX is able to solve Q_x for all ECM instances within an hour. In contrast to Q_x , Q_z yields a much larger number of variables, since decisions can be made on both the container contents and their loading day. Whereas Q_z involves the x_{obp} variables for the container contents and the w_{opt} variables for the products sent on each day (the z_{ibp} variables being fixed), this configuration only considers the z_{ibp} variables for the truck contents (the x_{obp} variables being fixed and the

w_{opt} variables being deduced from the values of x_{obp}).

The proposed DM for solving Q is straightforward. It first solves Q_z with TDM. It then solves Q_x with CPLEX (i.e., by optimizing the truck contents and taking as input the previously optimized container-loading schedule and contents).

3.4.2 Fix-and-optimize matheuristic (FOM)

The FOM aims at optimizing the content of both the trucks and the containers by successively considering different subsets of trucks and containers to be optimized. The pseudocode of FOM is given in Algorithm 3. At each step, $|I^{(nf)}|$ trucks and $|O^{(nf)}|$ containers are randomly selected. Preliminary experiments (not reported here) show that selecting in priority containers and trucks with a high potential for improvement (e.g., the containers and trucks that can transport the largest number of different products) leads to subproblems $Q(O^{(nf)}, I^{(nf)})$ that are harder to solve (as there are more permutations allowed, the solution space is bigger). Selecting randomly the set of containers and trucks allows to build problems $Q(O^{(nf)}, I^{(nf)})$ of similar size at each step of the FOM and to visit a sufficiently large number of solutions during the allowed execution time. $Q(O^{(nf)}, I^{(nf)})$ is then solved with CPLEX, and the provided solution is taken as input for the next iteration (we propose to adaptively update the size of the $I^{(nf)}$ and $O^{(nf)}$ sets with Algorithm 4 below). Algorithm 3 takes as input an initial feasible solution s_0 , e.g., the one currently used by ECM. It stops after η_{max} iterations without improvement or after t_{max} minutes of execution time (see the ‘While’ loop). (σ, t_{MILP}) are the MILP parameters used to solve $Q(O^{(nf)}, I^{(nf)})$ in Step 2. More precisely, the MILP stops when the gap to optimality is below $\sigma\%$ or after t_{MILP} minutes of execution time. An initial pair of percentages $(\rho_1^I < \rho_2^I)$ (resp. $(\rho_1^O < \rho_2^O)$) is also given as input for the proportion of trucks (resp. containers) to be optimized in $Q(O^{(nf)}, I^{(nf)})$. Such proportions are updated each η iterations of Algorithm 3 (see Step 4). We con-

sider two different values to be able to determine, during the execution of Algorithm 3, whether smaller percentages (i.e., ρ_1^I and ρ_1^O) or larger percentages (i.e., ρ_2^I and ρ_2^O) should be favored for the next iterations (see Algorithm 4 below). To evaluate the gain associated with the percentage (ρ_i^I, ρ_j^O) selected in Step 1, Step 3 computes the achieved inventory penalty reduction Δ_{ij} after η iterations and the associated required execution time τ_{ij} . Preliminary experiments (not reported here) have indicated that the tuning ($\eta = 12$, $\sigma = 2\%$, $t_{MILP} = 10$ minutes, $\rho_1^I = 15\%$, $\rho_2^I = 17\%$, $\rho_1^O = 5\%$, $\rho_2^O = 6\%$) is efficient.

Algorithm 3 Fix-and-optimize matheuristic (FOM)

Input: s_0 , (σ, t_{MILP}) , (η_{max}, t_{max}) , (ρ_1^I, ρ_2^I) , (ρ_1^O, ρ_2^O) , η .

Initialization: set $l = 1$; set $\Delta_{ij} = 0$ and $\tau_{ij} = 0$ ($\forall i, j \in \{1, 2\}$).

While (execution time $< t_{max}$) **or** (a solution improvement has been made in the last η iterations), **do:**

1. *Select the set of trucks and the set of containers to optimize:* choose randomly (i, j) (where $i, j \in \{1, 2\}$), and select randomly $|I^{(nf)}| = \lceil \rho_i^I \cdot |I| \rceil$ trucks and $|O^{(nf)}| = \lceil \rho_j^O \cdot |O| \rceil$ containers.
2. *Solve* $Q(O^{(nf)}, I^{(nf)})$ with CPLEX and let s_l be the resulting solution.
3. *Evaluate the performance of the selected* (ρ_i^I, ρ_j^O) : set $\Delta_{ij} = f(s_l) - f(s_{l-1})$ (where $f(s_l)$ is the inventory penalty of s_l) and add to τ_{ij} the execution time required to solve $Q(O^{(nf)}, I^{(nf)})$.
4. *Periodically update the truck/container percentages:* if $(l \bmod \eta) = 0$, update (ρ_1^I, ρ_2^I) and (ρ_1^O, ρ_2^O) with Algorithm 4; re-initialize $\Delta_{ij} = 0$ and $\tau_{ij} = 0$, $\forall i, j \in \{1, 2\}$.
5. *Move to the next iteration:* set $l = l + 1$.

Return: s_l (i.e., the last generated solution).

Algorithm 4 aims at choosing the pairs of percentages that will be used for the next sequence of η iterations of Algorithm 3 (see its Step 4). It takes as input the values of the percentages (ρ_1^I, ρ_2^I) and (ρ_1^O, ρ_2^O) used during the previous η iterations, as well as their associated inventory penalty reductions (Δ_{ij}) and execution times (τ_{ij}) .

For each couple (ρ_i^I, ρ_j^O) (where $i, j \in \{1, 2\}$), Algorithm 4 first computes the improvement score $\theta_{ij} = \frac{\Delta_{ij}}{\tau_{ij}}$ (in m^3/minute) provided by $Q(O^{(nf)}, I^{(nf)})$ (with $I^{(nf)} = \lceil \rho_i^I \cdot |I| \rceil$ and $O^{(nf)} = \lceil \rho_j^O \cdot |O| \rceil$) during the last sequence of η iterations of Algorithm 3. If no percentage configuration has improved the solution (i.e., if $\theta_{ij} = 0 \forall i, j \in \{1, 2\}$, case 1), it can either be due to the fact that the percentages (ρ_2^I, ρ_2^O) are too small (hence the solution space explored in $Q(O^{(nf)}, I^{(nf)})$ is too narrow), or (ρ_1^I, ρ_1^O) are too large (hence CPLEX cannot explore the solution space of $Q(O^{(nf)}, I^{(nf)})$ within t_{MILP} minutes to find a better solution than the current one). Therefore, in that case, we move the smaller percentage ρ_1^I (resp. ρ_1^O) of trucks (resp. containers) to an even smaller value, and the larger percentage ρ_2^I (resp. ρ_2^O) to an even larger value.

When at least one percentage pair has allowed the MILP to improve the input solution, let $(i^*, j^*) = \arg \max_{(i,j) \in \{1,2\}^2} \theta_{ij}$ (break ties randomly). In case 2 (resp. case 4), corresponding to $i^* = 1$ (resp. $j^* = 1$), as the smaller percentage ρ_1^I (resp. ρ_1^O) of trucks (resp. containers) to be optimized has yielded higher improvement score, we move the two percentages of trucks (resp. containers) to even smaller values. Conversely, in case 3 (resp. case 5), corresponding to $i^* = 2$ (resp. $j^* = 2$), the larger percentage ρ_2^I (resp. ρ_2^O) of trucks (resp. containers) to be optimized has yielded higher improvement score, and we thus move the two percentages of trucks (resp. containers) to even larger values.

Algorithm 4 Update of the percentages of trucks and containers to be optimized in $Q(O^{(nf)}, I^{(nf)})$

Input: $(\rho_i^I, \rho_j^O), \Delta_{ij}, \tau_{ij}, \forall i, j \in \{1, 2\}$.

Initialization:

- Set $\delta^I = \rho_2^I - \rho_1^I$ and $\delta^O = \rho_2^O - \rho_1^O$.
- Compute the improvement score for (ρ_i^I, ρ_j^O) : set $\theta_{ij} = \frac{\Delta_{ij}}{\tau_{ij}}, \forall i, j \in \{1, 2\}$.

If $\theta_{ij} = 0 \forall i, j \in \{1, 2\}$ (case 1), set: $\rho_1^I = \rho_1^I - \delta^I$; $\rho_2^I = \rho_2^I + \delta^I$; $\rho_1^O = \rho_1^O - \delta^O$; $\rho_2^O = \rho_2^O + \delta^O$.

Else Determine $(i^*, j^*) = \arg \max_{(i,j) \in \{1,2\}^2} \theta_{ij}$ (break ties randomly).

If $i^* = 1$ (case 2), set $\rho_2^I = \rho_1^I$ and $\rho_1^I = \rho_1^I - \delta^I$;

If $i^* = 2$ (case 3), set $\rho_1^I = \rho_2^I$ and $\rho_2^I = \rho_1^I + \delta^I$;

If $j^* = 1$ (case 4), set $\rho_2^O = \rho_1^O$ and $\rho_1^O = \rho_1^O - \delta^O$;

If $j^* = 2$ (case 5), set $\rho_1^O = \rho_2^O$ and $\rho_2^O = \rho_1^O + \delta^O$.

Return: $(\rho_i^I, \rho_j^O), \forall i, j \in \{1, 2\}$.

3.4.3 Combined matheuristic (DM-FOM)

When the available execution time is larger than the run time of DM, we propose the following combined matheuristic, referred to as DM-FOM. In a first phase, we launch DM. In a second phase, we use the remaining available execution time to run FOM, taking the DM solution as an input and further improving it. Whereas DM-FOM aims at solving configuration Q (i.e., both the truck and container contents are optimized), TDM-FOM combines TDM and FOM in the same fashion to solve configuration Q_z (where $|I^{nf}| = 0$, i.e., the truck contents are fixed).

3.4.4 Facilitated implementation of FOM

Additional advantages of FOMs were highlighted by [Papageorgiou et al., 2018]. In the context of the ECM problem, FOM stands out from other matheuristics, and more generally from metaheuristics, by the fact that sustainability and simplified maintenance of the

code is ensured by ECM, implying that it is easier for the optimization team to manage one single MILP that relies on a general purpose solver rather than a low level code that requires high maintenance. Furthermore, FOM is able to handily adapt to new business settings since, it requires less effort to update one single MILP rather than customized algorithms.

3.5 Computational experiments

The models were coded in C++ and CPLEX 12.4 was called to solve the induced MILPs. Computations were launched on a 2.2 GHz Intel Core i7 with 16 Go 1600 MHz DDR3 of RAM memory. The ECM problem is solved once a week. In accordance with ECM, it is therefore reasonable to consider an overall execution time of 10 hours. However, for most of the experiments presented below, an execution time of one hour was sufficient to obtain the presented solutions.

Section 3.5.1 describes the set of instances provided by ECM. We compare the solution methods in Section 3.5.2; results for configurations Q_z are given in Section 3.5.2 and results for Q in Section 3.5.2. Configuration Q_x is not treated here since it can be solved directly with CPLEX. Finally, Section 3.5.3 presents managerial insights by comparing the results of configuration Q with those of Q_z and Q_x .

3.5.1 Test instances

Table 3.2 gives the characteristics of the 17 instances provided by ECM. Three ILPs are considered (V, G and M), denoting three different sites where ECM is operating. The first column indicates the name of the instances, columns 2 to 7 indicate the size of the sets

introduced in Section 3.3.1. “ V^{init} ” gives the volume of the boxes located in the inventory at the beginning of the week. The last two columns describe the size of configuration Q for each instance: “Nb. Var.” (resp. “Nb. Const.”) gives the number of variables (resp. constraints). We have removed from the model the variables that can only take a single value (e.g., for instance M7, there are $|O| \times |B| \times |P| \approx 10^{10}$ x_{obp} variables, but after variable elimination, configuration Q involves less than six millions variables (see [Coindreau et al., 2019b] for more details on this variable elimination procedure).

Table 3.2: Characteristics of the test instances.

Instance	$ O $	$ I $	$ P $	$ B $	$ S $	$ C $	V^{init}	Nb. Var.	Nb. Const.
V1	28	48	326	206	151	17	634	12,031	14,107
V2	51	78	358	290	171	20	1,154	14,054	17,234
V3	49	67	424	315	190	21	898	16,554	20,492
V4	59	82	454	334	191	20	1,411	19,022	23,700
G1	67	98	1,181	616	544	8	1,098	119,238	166,937
G2	71	112	1,199	644	554	7	942	132,638	182,749
G3	68	89	1,353	575	572	8	1,341	161,969	218,955
G4	88	112	1,401	718	606	8	1,503	162,171	231,079
G5	80	122	1,548	605	646	8	18,92	243,688	315,456
G6	85	136	1,676	748	678	7	917	244,916	330,369
M1	383	677	6,564	999	542	17	15,392	3,283,199	4,382,861
M2	543	653	7,890	1,262	626	23	14,995	3,808,742	5,403,274
M3	644	903	7,865	1,226	568	22	14,933	4,372,938	6,234,108
M4	699	778	7,529	1,167	597	23	23,458	4,062,400	5,641,036
M5	623	741	8,349	1,159	608	23	14,211	4,923,679	6,754,536
M6	789	1,085	8,546	1,377	590	21	23,828	5,272,599	7,626,068
M7	829	1,104	8,649	1,387	597	22	26,115	5,883,213	8,577,182

3.5.2 Analysis of the performance of the proposed solution methods

We now proceed to the analysis of our matheuristics on the configurations Q_z and Q .

Results on configuration Q_z

In this section, we focus on configuration Q_z involving only the decisions on the contents and on the loading days of the containers. We benchmark FOM and TDM-FOM (both with $|I^{nf}| = 0$, as the truck contents are fixed) with respect to TDM. For Q_z , [Coindreau et al., 2019b] showed that TDM is able to find optimal solutions on the smaller instances V and G. For the larger instances M, TDM is able to find solutions exhibiting a significant gain compared with the ECM current practice. We do not report the results for the V and G instances as, starting from the ECM solutions, FOM is able to find optimal solutions within five minutes. Recall that FOM and TDM-FOM are limited to 10 hours of execution time (TDM always returns its solution within less than 62 minutes for the larger instances M).

Table 3.3 compares the results of TDM, FOM and TDM-FOM for the M instances. Columns “Obj.” give the value of the objective function. “Time” indicates the time (in minutes) at which TDM returned its solution. “% best” provides the percentage gap with respect the best found inventory penalty, which is always returned by TDM-FOM. The percentage gap is computed as follows: $100 \cdot \frac{f_{TDM} - f_{TDM-FOM}}{f_{TDM-FOM}}$, where f_{TDM} (resp. $f_{TDM-FOM}$) denotes the inventory penalty of the solution returned by TDM (resp. $TDM - FOM$). The columns “% $O^{(nf)}$ ” give the average percentage of containers optimized at each iteration of FOM.

On the one hand, one can observe that the TDM solutions can be further improved by FOM during the remaining available execution time. Indeed, FOM is only able to improve the results of TDM in four out of the seven instances. On the other hand, TDM-FOM is able to improve the results of TDM for all instances with an average percentage gap of 1.1%. Such improvements could not be achieved without the use of FOM, as different runs of TDM always return the same solution. Finally, the “% $O^{(nf)}$ ” values show that all instances do not require the same average percentage of containers to be

optimized at each iteration of FOM. For example, large values are not appropriate for M7 because of the complexity due to its size (on average 7.4% of the containers are optimized for FOM). In contrast, for the smaller instances M3 and M5, much larger percentages of the containers are optimized (more than 20% for FOM). These results highlight the importance of dynamically updating, during the execution of FOM, the values of these percentages (see Algorithm 4).

Table 3.3: Results of Q_z for TDM, FOM and TDM-FOM (M instances).

Instance	TDM			FOM			TDM-FOM	
	Obj.	Time[min]	% best	Obj.	% $O^{(nf)}$	% best	Obj.	% $O^{(nf)}$
M1	43,947	11	1.1%	43,783	14.2%	0.7%	43,474	15.1%
M2	46,321	49	0.9%	46,215	13.5%	0.7%	45,897	12.2%
M3	53,416	23	0.3%	53,532	20.4%	0.5%	53,251	16.7%
M4	48,581	23	0.3%	48,987	14.0%	1.1%	48,435	12.0%
M5	51,501	24	0.6%	51,370	22.9%	0.3%	51,205	13.0%
M6	56,886	35	1.3%	56,687	11.9%	1.0%	56,144	12.3%
M7	60,212	62	2.9%	66,701	7.4%	14.0%	58,490	9.8%

Results on configuration Q

For the V and G instances, Table 3.4 compares the results of DM and FOM with the optimal solutions proven by CPLEX in the eponymous columns. The columns “Obj.” and “Time” are defined as above (but the time is given in seconds). For DM and FOM, the column “% opt.” provides the gap with respect to the optimal solution. The columns “% $O^{(nf)}$ ” and “% $I^{(nf)}$ ” give the average percentage of containers and trucks optimized at each iteration of FOM, respectively (this will be commented later). We observe that DM is faster than FOM, but the latter heuristic yields better solutions. Indeed, for FOM, the average gap to optimality never exceeds 2% for each instance. Interestingly, FOM requires on average 32% less execution time than CPLEX. The results of DM-FOM are not reported for the V and G instances. Indeed, FOM already shows a good performance for these smaller instances, and neither the objective nor the execution time are significantly improved by DM-FOM.

Table 3.4: Results of Q for CPLEX, DM and FOM (V and G instances).

Instance	CPLEX		DM			FOM				
	Obj.	Time[s]	Obj.	Time[s]	% opt.	Obj.	Time[s]	% $O^{(nf)}$	% $I^{(nf)}$	% opt.
V1	691	50	694	<1	0%	692	4	24.1%	34.4%	0%
V2	1,229	28	1,229	2	0%	1,229	1	8.3%	11.7%	0%
V3	1,481	1	1,481	3	0%	1,481	8	24.5%	37.1%	0%
V4	1,727	15	1,727	4	0%	1,727	9	22.4%	31.0%	0%
G1	2,489	104	2,643	26	6%	2,536	86	26.3%	39.9%	2%
G2	2,421	311	2,590	29	7%	2,450	129	26.4%	40.2%	1%
G3	2,621	56	2,632	31	0%	2,624	79	24.7%	31.1%	0%
G4	3,339	192	3,406	44	2%	3,346	85	23.0%	32.0%	0%
G5	2,651	1867	3,105	51	17%	2,705	338	26.1%	40.9%	2%
G6	3,246	691	3,404	65	5%	3,326	198	22.3%	35.0%	2%

Table 3.5 compares the results of DM, FOM and DM-FOM for the M instances. For DM, we report the execution time (in minutes) in the column “Time”. The time is not reported for FOM and DM-FOM since for these instances, the full 10-hour time budget is used. The column “% best” provides the gap with respect to the solution value found by DM-FOM (which is always the best solution). Finally, for both FOM and DM-FOM, we report the average percentage of trucks and containers optimized at each iteration in columns “% $O^{(nf)}$ ” and “% $I^{(nf)}$ ”, respectively.

Table 3.5 highlights that DM-FOM allows to efficiently use the available 10 hours of execution time and outperforms both DM and FOM. The average gap between DM and DM-FOM (resp. between FOM and DM-FOM) is 11.1% (resp. 5.3%). DM turns out to be a powerful first phase for the ECM problem: it demonstrates the importance of considering the problem characteristics to find appropriate decomposition techniques. Here, maximizing the volume sent daily is particularly efficient and is one of the strengths of DM. DM is able to quickly identify good solutions (within 18 minutes of execution time for the smallest instance M1, and 153 minutes for the largest instance M7) and is therefore recommended as a warm start for FOM, as opposed to initially feeding FOM with the ECM solution. It is interesting to note that considering simultaneously the optimization of the truck and of the container contents is necessary in order to be able to further improve the results returned by DM.

Additional experiments (not reported here) indicate that letting 10 hours of execution for DM only (i.e., iteratively solving Q_z and Q_x for 10 hours) does not improve the solution found after one single iteration of DM (i.e., solve Q_z then Q_x once). Indeed, decomposing the resolution with Q_z followed by Q_x is efficient to quickly find a rather good solution, but cannot, in contrast to FOM, further improve it. In this case, as the truck (or container) contents are always optimized to be suitable to the container (or truck) contents given as input, reoptimizing always yields similar truck and container contents.

The average gap between DM and DM-FOM is larger when solving Q than when solving Q_z (on average, it moves from 1.1% for Q_z to 14.1% for Q). This indicates that DM is more efficient on configuration Q_z than on Q . Hence, when integrating the decisions on both the truck and container contents with those on the container scheduling (i.e., configuration Q), FOM becomes an essential tool. The average gap between FOM and DM-FOM is equal to 4.56%, highlighting again the importance of considering the solution of DM as a warm start for FOM.

Tables 3.4 and 3.5 show how the values of the percentages of trucks and containers to be optimized at each iteration of FOM adapt to the characteristics of the instances. Typically, the percentage of optimized trucks is larger than the percentage of optimized containers. This stems from the increased complexity of Q_z compared with Q_x . Furthermore, we observe that the larger is the instance, the smaller is the percentage of trucks or containers to be optimized at each iteration of FOM. The strength of FOM lies more in the number of performed iterations within the allowed time rather than on the magnitude of the improvement achieved at each iteration.

Table 3.5: Results of Q for DM, FM and DM-FOM (M instances).

Instance	DM			FOM				DM-FOM		
	Obj.	% best	Time[min]	Obj.	% best	% $O^{(nf)}$	% $I^{(nf)}$	Obj.	% $O^{(nf)}$	% $I^{(nf)}$
M1	25,492	26.1%	18	20,335	0.6%	11.5%	29.5%	20,209	15.6%	24.2%
M2	36,669	16.2%	31	32,443	2.8%	10.6%	30.3%	31,559	11.8%	25.1%
M3	36,909	24.3%	104	33,847	14.0%	10.9%	23.5%	29,693	11.7%	20.8%
M4	36,459	8.5%	37	34,350	2.2%	9.2%	26.7%	33,614	12.9%	17.0%
M5	34,421	16.2%	42	29,661	0.1%	12.5%	22.9%	29,627	13.3%	20.2%
M6	41,670	2.5%	127	43,301	6.5%	8.0%	20.8%	40,670	10.1%	15.9%
M7	42,256	5.0%	153	44,206	9.8%	8.5%	22.8%	40,249	10.0%	17.1%

3.5.3 Managerial insights

We now evaluate the potential gain in terms of inventory penalty offered to ECM when simultaneously considering both the truck and container contents in the optimization, together with the loading day of the containers. In particular, we compare the results with those obtained in situations where only the decisions on the content of the trucks or the containers, or neither (which sets for the current practice at ECM), are integrated with those on the scheduling of container loading operations. We first compare the returned optimal solutions for the V and G instances. Next, we compare the best found solutions for the M instances. Last, we summarize the improvement potential achieved by our integrated approach.

V and G instances

Table 3.6 compares the obtained solutions for configuration Q (i.e., both the contents of the trucks and the containers are optimized) with those achieved when (1) the content of the containers is fixed (column “ Q_x ”); (2) the content of the trucks is fixed (column “ Q_z ”); (3) both the content of the trucks and that of the containers are fixed (column “ECM” which corresponds to the configuration $Q_{x,z}$). The columns “Obj.” report the value of the optimal solution and the columns “Time” give the time (in minutes) at which CPLEX

returned the optimal solution. Columns “% (ECM)”, “% Q_z ” and “%(Q_x)” give the improvement percentage with respect to configuration ECM, Q_z and Q_x , respectively. For example, the improvement achieved by configuration Q over configuration Q_z is displayed in column “% Q_z ” and is computed as $\frac{f(Q)-f(Q_z)}{f(Q_z)}$, where $f(Q_z)$ (resp. $f(Q)$) designates the obtained inventory penalty when considering configuration Q_z (resp. Q).

As already discussed in [Coindreau et al., 2019b], reconsidering the content of the containers during the optimization of the container loading operations allows to significantly improve the solution currently used at ECM (with an average improvement of 5% for the V instances and of 15% for the G instances). The gain achieved is of the same magnitude when integrating only the decisions on the content of the trucks in the optimization (with an average improvement of 17% for the V instances and of 14% for the G instances). The main improvement is achieved when we consider simultaneously the content of the trucks and containers together with the loading day of the containers. Compared with the results obtained in [Coindreau et al., 2019b], the additional average improvement brought by solving Q instead of Q_z amounts to 16% for the V instances, and to 19% for the G instances. Compared with the current practice at ECM, the average improvement achieved by optimizing both on the truck and container contents is 20% for the V instances, and 31% for the G instances. For these V and G instances, we recall that the largest execution time to find the optimal solutions with CPLEX is 31 minutes.

Table 3.6: Results of Q for the V and G instances.

Instance	ECM		Q_z			Q_x			Q				
	Obj.	Time[min]	Obj.	Time[min]	% (ECM)	Obj.	Time[min]	% (ECM)	Obj.	Time[min]	% (ECM)	% Q_z	% Q_x
V1	1,158	< 1	1,158	< 1	0%	695	< 1	-40%	691	1	-40%	-40%	-1%
V2	1,454	< 1	1,377	< 1	-5%	1,307	< 1	-10%	1,229	< 1	-15%	-11%	-6%
V3	1,710	< 1	1,631	1	-5%	1,484	< 1	-13%	1,481	1	-13%	-9%	0%
V4	2,117	< 1	1,956	< 1	-8%	1,888	< 1	-11%	1,727	< 1	-18%	-12%	-9%
G1	3,292	< 1	2,801	1	-15%	3,144	1	-4%	2,489	2	-24%	-11%	-21%
G2	3,690	< 1	3,074	3	-17%	3,130	1	-15%	2,421	5	-34%	-21%	-23%
G3	3,517	< 1	2,943	1	-16%	3,141	1	-11%	2,621	1	-25%	-11%	-17%
G4	4,318	< 1	3,672	3	-15%	4,005	2	-7%	3,339	4	-23%	-9%	-17%
G5	4,843	< 1	4,172	5	-14%	3,450	4	-29%	2,651	31	-45%	-36%	-23%
G6	4,706	< 1	4,059	4	-14%	4,037	3	-14%	3,246	12	-31%	-20%	-20%

M instances

Table 3.7 presents, for the M instances, the best solutions obtained. The columns of Table 3.7 correspond to those of Table 3.6. When the time is not reported, this means that the algorithm used the entire allowed 10 hours of execution time to obtain the achieved inventory penalty.

For these larger instances, and similarly to the smaller instances, Table 3.7 shows that optimizing only the truck or the container contents leads to similar average improvement when compared to the ECM current practice. The average improvement of Q_z (resp. Q_x) over ECM is 34% (resp. 29%). The main improvement comes when considering all the decisions simultaneously (both the truck, the container contents, and the container loading day). The average inventory-penalty reduction when solving Q instead of Q_z (as done in [Coindreau et al., 2019b]) is 37%. Compared with the ECM current practice, the gain is up to 72%, with an average of 58%. Such observations confirm the importance of integrating decisions on both the inbound and outbound sides at ECM's ILPs.

Table 3.7: Results of Q for the M instances.

Instance	ECM		Q_z		Q_x			Q			
	Obj.	Time[min]	Obj.	% (ECM)	Obj.	Time[min]	% (ECM)	Obj.	% (ECM)	% Q_z	% Q_x
M1	73,301	< 1	43,474	-41%	46,401	4	-37%	20,209	-72%	-54%	-56%
M2	56,547	< 1	45,897	-19%	44,144	3	-22%	31,559	-44%	-31%	-29%
M3	77,077	< 1	53,251	-31%	51,873	12	-33%	29,693	-61%	-44%	-43%
M4	71,518	< 1	48,435	-32%	52,942	5	-26%	33,614	-53%	-31%	-37%
M5	73,436	< 1	51,205	-30%	51,312	6	-30%	29,627	-60%	-42%	-42%
M6	93,518	< 1	56,144	-40%	72,663	16	-22%	40,670	-57%	-28%	-44%
M7	100,504	< 1	58,490	-42%	66,761	40	-34%	40,249	-60%	-31%	-40%

Improvement potential

For each instance, any achieved inventory penalty at the ILP lies between the inventory penalty of the ECM solution and the inventory penalty observed at the beginning of the

week (see Table 3.2 for the values V^{init} of the volume of products stored at the beginning of the week). For instance M1, the initial inventory volume V^{init} stored at the beginning of the week is 15,392 m³ and the largest inventory volume $f(Q_{x,z})$ observed in the current ECM solution is 73,301 m³. Any improving solution lies within these two bounds and the maximum theoretical improvement potential for this instance is 57,909 m³. Considering Q_z allows a reduction of the largest storage volume to 43,783 m³, and the savings for ECM is 29,518 m³, which represents 51% of the maximum theoretical improvement potential (when only direct product transfers would take place).

Figure 3.2 displays, for all instances and for both configurations Q_z and Q , the achieved percentage of the maximum theoretical improvement potential. Each bar represents an instance, and the bold bar indicates the average for the V, G, and M instances. For each ILP, Figure 3.2 highlights the significant additional gain achieved when considering Q over Q_z . It furthermore indicates that, on average and for both configurations Q_z and Q , the larger is the instance, the larger is the achieved improvement in terms of inventory penalty. This shows that our solution methods can take advantage of the increased potential for product exchange between trucks and containers in larger instances.

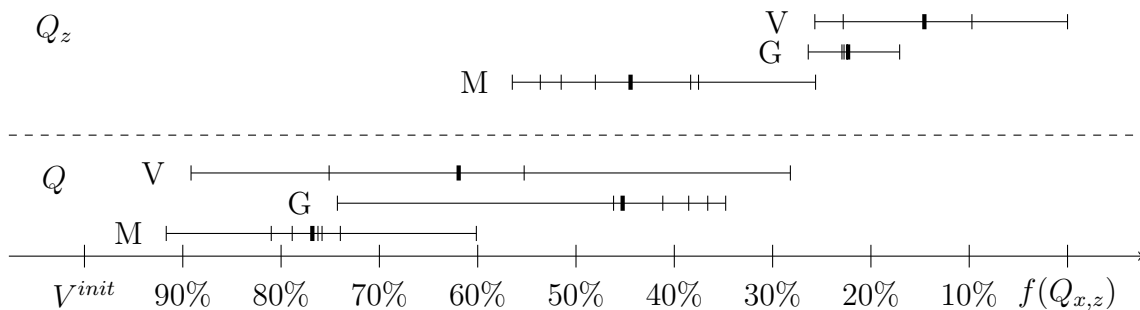


Figure 3.2: Percentage of the maximum theoretical improvement potential achieved by configurations Q_z and Q .

3.6 Conclusions

We have modeled and solved an industrial problem that considers the scheduling and the product assignment for both the inbound and outbound flows in a cross-docking platform. Whereas [Coindreau et al., 2019b] integrated the decisions on the outbound container content with the scheduling of their operations over the week, here we additionally included the decisions on the inbound truck contents in the same optimization framework. Concerning the complex loading constraints that affect both trucks and containers, we proposed an efficient formulation capable of quickly capturing the feasibility of different contents, as well as to evaluate their quality. We were able to realize the significant improvement potential offered by the proposed integrated optimization framework for all instances provided by the involved company ECM.

We have developed and compared two heuristics, namely a decomposition matheuristic (DM) and a fix-and-optimize matheuristic (FOM). DM is faster but less efficient than FOM. The results are better when both methods are combined. Computational experiments showed that, compared with current industrial practice, allowing product reassignment from one container to another and from a truck to another, can reduce the average required largest inventory volume by 58% for the large instances and by 27% for the small ones. Moreover, compared with the situation where only the content of the containers is included in the decision making, the integrated formulation allows an additional average reduction of 37% for the large instances, and of 18% for the small ones. From a managerial point of view, revoking the content of the trucks may be a more challenging task than acting on those of the containers, as it involves third parties. However, this study clearly shows that implementing such aspects has the potential of yielding a significant improvement with respect to current practice and should therefore be considered.

Acknowledgement

This work was partly supported by the Canadian Natural Sciences and Engineering Research Council under grant 2015-06189. This support is gratefully acknowledged. Thanks are due to the reviewers for their valuable comments.

Chapter 4

Synchronizing Trucks and Drones for a Real-World Parcel Delivery Problem with Time-Window Constraints

MARC-ANTOINE COINDREAU - *University of Lausanne, Switzerland*

OLIVIER GALLAY - *University of Lausanne, Switzerland*

NICOLAS ZUFFEREY - *University of Geneva, Switzerland*

Abstract

The negative impact of excessive traffic in urban areas requires innovative transportation concepts. One solution relies on autonomous delivery drones embedded in delivery trucks. We consider the case of a large *European Logistics Provider* (ELP) that aims to schedule parcel deliveries with a fleet of truck-and-drone vehicles. The truck routes start from a depot with a set of parcels to be delivered within given time windows. When appropriate, the drones can be loaded with a parcel, launched directly from the truck, and sent to a customer. Afterward, the drones autonomously return to the truck, where they will be replenished and recharged. We propose a mixed-integer linear programming formulation and an *Adaptive Large Neighborhood Search* (ALNS). Using the real cost structure of the ELP and the traditional truck-only delivery as a benchmark, we analyze the gain offered by this new transportation concept. The obtained bi-modal truck-and-drone solutions determine the most efficient allocation of customers among drones and trucks, as well as the locations at which the drones are launched and retrieved along the truck routes. Results show that truck-and-drone solutions can reduce costs by up to 35% when compared to traditional truck-only delivery. Managerial insights are also given: a minimum percentage of customer locations must be reachable by drones to find competitive truck-and-drone solutions (i.e., to allow the fixed costs of the drones to be compensated by the savings achieved on the truck routes) and the cost structures of truck-and-drone and truck-only solutions are compared.

Keywords: Vehicle Routing, Drones, Mixed-Integer Linear Program, Adaptive Large Neighborhood Search.

4.1 Introduction

Autonomous transportation, electric vehicles and multi-modality are often regarded as the most promising ways to resolve congestion problems, as well as reduce the environmental impact related to transportation activities [Speranza, 2018]. In particular, the use of autonomous drones to transport goods has recently drawn considerable attention from both industry (e.g., [DHL, 2014], [Amazon, 2016], [Daimler, 2017]) and scholars (e.g., [Otto et al., 2018] review more than 200 articles on optimization problems related to the use of drones for operations planning). The introduction of such a novel transportation mode provides managers with an alternative way to efficiently deliver parcels in large urban areas that are difficult or costly to access by trucks. However, whereas drone delivery turns out to be faster and cheaper than truck distribution [Wohlsen, 2014], a straightforward replacement of trucks by drones cannot be envisioned as drones suffer from limited flight range and restricted capacity.

We consider the case of a large *European Logistics Provider* (ELP) that cannot be named because of a non-disclosure agreement. The ELP proposes to use a mixed fleet of vehicles where drones are embedded into trucks. The mission of the trucks (and their assigned drivers) is twofold: (1) deliver parcels to customer locations and (2) manage (i.e., retrieve, transport, load, recharge, and launch) the drones along their routes. The synchronization of such a truck-and-drone fleet enables benefiting from the specific advantages of both of these vehicle types. Other examples of such mixed fleets of vehicles include trucks and autonomous robots [Boysen et al., 2018b], cars and non-motorized workers [Coindreau et al., 2019a], and trucks and trailers [Chao, 2002, Lin et al., 2009] (*Truck and Trailer Routing Problem*) or [Drex1, 2014] (the *Vehicle Routing Problem with Trailers and Transshipments*). In the latter case, a truck may or may not tow a trailer within the same solution).

Following the ELP requirements, we focus on the static day-ahead scheduling of parcel

deliveries, where all travel times are known in advance. As in the *Vehicle Routing Problem with Time Windows* (VRPTW) [Laporte, 2009], in which only trucks are available to deliver parcels, a time window (TW) is associated with each customer delivery. A fleet of truck-and-drone vehicles is available to transport the given set of parcels. Both trucks and drones have specific characteristics. Although trucks generate higher costs and travel more slowly, they benefit from a larger capacity. Drones are cost-effective and fly faster, but they suffer from a limited capacity and flight range. We formulate this problem as the *Minimum Cost Vehicle Routing Problem with Time Windows and Drones*, denoted as MC-VRPTW-D. Multiple trucks are considered, that can embed a drone or not. The drone's flight endurance, which is due to its battery size, and the maximal working-day duration for the drivers are both taken into account. The global cost function to be minimized is motivated by the ELP. It includes fixed costs (daily vehicle use) and variable costs (workforce wage, traveled distance, and driving and flying time) generated by the delivery operations of the truck-and-drone fleet.

The present work yields the following contributions.

1. We extend the *Vehicle Routing Problem with Drones* (VRP-D) by introducing the MC-VRPTW-D formulation, which aims to minimize the global cost function and considers TWs.
2. We propose a *Mixed-Integer Linear Program* (MILP) for the MC-VRPTW-D.
3. We design an insertion-based metaheuristic, namely an *Adaptive Large Neighborhood Search* (ALNS), to solve the MC-VRPTW-D. Our ALNS includes an algorithm to speed up the insertion phase. We compare the obtained ALNS solutions with those resulting from the commonly used *Route-First-Cluster-Second* (RFCS) procedures.
4. We solve an instance set that captures the various real situations encountered by the ELP.
5. Managerial insights are given for the efficient use of truck-and-drone fleets. We

quantify the achieved cost reduction and compare it to truck-only delivery. We identify instance characteristics that allow the use of drones to be competitive. We depict the cost structure of the obtained truck-and-drone solutions. The provided results and insights open the door for a novel management technique for delivery operations, where practitioners can optimize their costs efficiently using the specific characteristics of the two considered transportation modes.

The paper is organized as follows. A literature review is conducted in Section 4.2, with a focus on the operational aspects associated with mixed fleets of trucks and drones. In Section 4.3, a formal description of the MC-VRPTW-D is provided, and we introduce the corresponding MILP. The ALNS is presented in Section 4.4. It can efficiently handle the required synchronization of the two vehicle types. Section 4.5 proposes an algorithm that formulates the ALNS insertion procedure specifically for the considered truck-and-drone context. Computational experiments are presented in Section 4.6, where the potential gain offered by truck-and-drone fleets is quantified. Conclusions and extensions are given in Section 4.7.

4.2 Literature Review

Managing vehicle fleets composed of both trucks and drones has recently attracted substantial interest from the research community. Among the papers that consider trucks and drones for parcel delivery, we only review, here, those involving en-route synchronization (i.e., the situations where the drones meet the trucks on their way in order to refill their load and recharge their battery). [Drex1, 2012] identifies the different synchronization types that can take place in a routing context. Accordingly, papers considering cases where the drones deliver parcels directly from the depot without any en-route synchronization with trucks are not mentioned below (e.g., [Ham, 2018]). We have identified

16 papers, published between 2015 and 2019, that specifically address the en-route synchronization of truck-and-drone fleets. These contributions are compared in Table 4.1 according to the following seven operational characteristics.

1. *Configuration* (column ‘Config.’) synthesizes the considered number of trucks (‘Nb. Trucks’) and drones (‘Nb. Drones’). It also specifies the largest number of drones that can be carried by a truck (‘Max. Drones per Truck’). This information is listed as follows: ‘Nb. Trucks/Nb. Drones/Max. Drones per Truck’. For example ‘ $N/N/m$ ’ denotes a configuration involving more than one truck and more than one drone, and where each truck can embed more than one drone.
2. *Objective* (column ‘Obj.’) denotes the considered objective. We have found three different types of objectives. ‘Makespan’ identifies configurations that minimize the time at which the last truck returns to the depot (frequently denoted as makespan in the associated literature), ‘Op. Cost’ (resp. ‘Gl. Cost’) refers to formulations that minimize the variable operational costs (resp. the global costs, which include both variable and fixed costs). Operational costs include the expenses linked to operating trucks and drones and, for some configurations, the drivers salaries. The global costs take into account, in addition to the operational costs described above, the fixed costs incurred by engaging trucks and drones. In the present work, the number of trucks and drones are minimized while ensuring that all customer deliveries are performed within their associated TWs. The global costs turn out to be a generalization of the other objective functions. Indeed, driver salaries are proportional to the makespan. For specific values of the model’s parameters, the variable operational costs are found by removing the fixed costs from the objective function, and the makespan corresponds to cases where only drivers salaries remain in the objective function.
3. *Vehicle Routing Problem (VRP) Constraints* (column ‘VRP Cst.’) indicates which VRP constraints are taken into account. ‘Capa.’ stands for the capacity constraint

restricting the total parcel weight carried by each truck. ‘TW’ refers to time window constraints. ‘T-max’, which stands for the truck maximal day duration, limits the number of daily working hours for each truck driver.

4. *Synchronization Type* (column ‘Synch.’) specifies how drones synchronize with trucks to be refilled. All papers listed in Table 4.1 address the situation where a drone can be launched and retrieved at different locations, referred to as *acyclic* configuration. We identify the configurations that, in addition to the acyclic case, consider a ‘Cyclic’ configuration where the drones can be launched and retrieved at the same location. ‘Switch’ indicates formulations allowing a drone to be managed by different trucks (i.e., launched by one truck and retrieved by another).
5. *Transfer Point* (column ‘Transfer’) indicates at which locations drones can be launched and retrieved. ‘Cust.’ refers to situations where drones can be launched and retrieved at any customer location. ‘Dock’ denotes cases where drones can be operated from specifically defined docking nodes. ‘Wherever’ indicates that drones can be launched and retrieved everywhere.
6. *Drone Constraints* (column ‘Drone Cst.’) indicate the specific drone constraints. We have identified two types of constraints associated with limited battery size: ‘Endu.’ when the drone’s endurance is taken into account (i.e., the time difference between the launch and the retrieval of the drone is upper-bounded); ‘Dist.’ when the drone’s flying distance is constrained. In the latter, it is assumed that drones can wait indefinitely for the retrieving truck to arrive. Note that all papers but [Wang and Sheu, 2019] consider that drones can only transport one parcel at a time.
7. *Solution Methods* (column ‘Method’) lists the considered solution methods. Several papers propose exact approaches to tackle truck-and-drone problems. Exact methods are classified with the following acronyms ‘MILP’, ‘BP’ (*Branch and Price*), ‘BB’ (*Branch and Bound*), and ‘DP’ (*Dynamic Programming*). These exact methods do not allow for solving instances larger than 20 customer deliveries. For larger instance sizes, heuristics have been proposed. ‘RFCS’ stands for the *Route-First-*

Cluster-Second heuristic. It starts from optimized routes obtained for a fleet made of trucks only. Next, it creates clusters for the drones by assigning customers to drones until no more saving can be achieved. Local search (LS) metaheuristics have also been extensively used to tackle larger instances of truck-and-drone problems: ‘GRASP’ (*Greedy Randomized Adaptive Search Procedure*); ‘SA’ (*Simulated Annealing*); ‘VNS’ (*Variable Neighborhood Search*); ‘ALNS’ (*Adaptive Large Neighborhood Search*). The last three metaheuristics iteratively improve one single solution during the execution of the algorithm. In SA, the generated neighbor solution replaces the current one according to an acceptance mechanism that allows for some deterioration at early stages of the metaheuristic. VNS and ALNS differ from SA by the fact that they consider large neighborhood search (e.g., LNS, proposed by [Shaw, 1998]), which means that large neighborhood structures are used in the search process (i.e., more significant modifications of the current solution structure can be performed to generate a neighbor solution). Whereas VNS only accepts improving solutions, ALNS uses the SA acceptance criteria to decide whether to move the search to the newly generated neighbor solution. At the intersection of exact methods and metaheuristics, ‘MH’ stands for *matheuristic*. Matheuristics combine the use of metaheuristics and of exact algorithms. ‘Cont Opt.’ indicates that continuous optimization is considered. Denoted by ‘Worst Case’, [Wang et al., 2017] and [Poikonen et al., 2017] compute theoretical bounds for various problems involving the synchronization of multiple trucks and multiple drones, but they do not propose a related solution method.

8. *Size* (column ‘Size’) specifies the largest number of customers handled by the different formulations.

Regarding the considered problem, Table 4.1 shows that no paper explicitly considers the specific characteristics addressed in the present work. More precisely, we integrate a global cost function that includes fixed and operational costs, TWs, and complex synchronization constraints between multiple trucks and drones (i.e., a truck can either wait at a customer’s

location to refill a drone multiple times or move on to the next customer's location while the drone is flying). Additionally, our formulation allows removing trucks and replacing them with drones. Moreover, no paper explicitly considers the trade-off between the numbers of trucks and drones employed.

Regarding the solution methods, Table 4.1 indicates that exact approaches can solve instances with up to 20 customers. Metaheuristics can tackle larger instances (i.e., involving more than 100 customers). In particular, ALNS has recently been used to efficiently solve even larger truck-and-drone problems [Sacramento et al., 2019] and has proven to be a powerful method for solving problems involving the synchronization of different types of vehicles (e.g., [Masson et al., 2014], [Grangier et al., 2016]). Heuristics based on the RFCS principle have been extensively used to solve truck-and-drone problems. These methods rely on an initial phase of efficient and well-established algorithms to solve the associated VRP with trucks only. The resulting solutions are then improved in the second phase by introducing drone sub-tours. Although RFCS can efficiently use VRP solutions created in the first phase, it suffers from being easily trapped in local minima. Indeed, the truck routes cannot be completely reshaped in the second phase of the procedure. Moreover, RFCS cannot find solutions involving fewer trucks than in the initial VRP solution. When optimizing a global cost function, it appears crucial to address the existing potential of replacing some trucks by drones with respect to the corresponding VRP solution. In this paper, we propose both a MILP and a dedicated ALNS for the MC-VRPTW-D, and we compare them with a standard RFCS procedure.

Paper	Config.	Obj.	VRP Cst.	Synch.	Transfer	Drone Cst.	Method	Size
[Murray and Chu, 2015]	1/1/1	Makespan	-	-	Cust.	Endu.	MILP, RFCS	20
[Ponza, 2016]	1/1/1	Makespan	-	-	Cust.	Endu.	SA	200
[Carlsson and Song, 2017]	1/1/1	Makespan	-	-	Wherever	-	Cont. Opt.	100
[Wang et al., 2017], [Poikonen et al., 2017]	$N/N/m$	Makespan	-	Cyclic	Cust.	Endu.	Worst Case	-
[Pugliese and Guerriero, 2017]	$N/N/m$	Op. Cost	TW	-	Cust.	Endu.	MILP	10
[Ha et al., 2018]	1/1/1	Op. Cost	-	-	Cust.	Endu.	MILP, GRASP	100
[Yurek and Ozmutlu, 2018]	1/1/1	Makespan	-	-	Cust.	-	RFCS	20
[Agatz et al., 2018], [Bouman et al., 2018]	1/1/1	Gl. Cost	-	-	Cust.	Endu.	RFCS, DP	10
[Boysen et al., 2018a]	$1/N/m$	Makespan	-	Cyclic	Cust.	-	MILP, SA	100
[Poikonen et al., 2019]	1/1/1	Makespan	-	Cyclic	Cust.	Endu.	BB	10
[Sacramento et al., 2019]	$N/N/1$	Op. Cost	Capa., T-max	-	Cust.	Endu.	MILP, ALNS	250
[Wang and Sheu, 2019]	$N/N/m$	Gl. Cost	Capa.	Switch	Dock	Endu.	BP	15
[Schermer et al., 2019a]	$N/N/m$	Makespan	-	-	Cust., Dock	Dist.	MILP, VNS	50
[Schermer et al., 2019b]	$N/N/m$	Makespan	-	Cyclic	Cust.	Dist.	MILP, MH	100
This work	$N/N/1$	Gl. Cost	TW T-max	Cyclic	Cust.	Endu.	MILP, RFCS, ALNS	100

Table 4.1: Comparison of related truck-and-drone formulations.

4.3 Problem Formulation

The ELP's practical assumptions are presented in Section 4.3.1. The considered type of synchronization of trucks and drones is described in Section 4.3.2. The considered sets, parameters, and variables are defined in Section 4.3.3. Finally, Section 4.3.4 proposes the MILP model for the MC-VRPTW-D.

4.3.1 Practical Assumptions

We consider the following assumptions associated with the ELP context.

- Each job (i.e., customer delivery) is served, exactly once, by either a truck or by a

drone.

- Some jobs are not eligible to be served by drones (e.g., parcels that are too heavy to be transported by a drone or customer locations at which drone landing is impossible).
- A truck can embed one drone.
- Each drone is assigned to a truck and returns to its assigned truck after each flight. To avoid robustness issues, the ELP does not consider the case where a drone can be retrieved by another truck.
- The drones can deliver a single parcel during each flight.
- The drones cannot accept jobs directly from the depot.
- The maximum working-day duration is fixed, and the drivers are paid for the whole duration spent outside the depot. When workers come back earlier at the depot, they can be employed for other tasks (e.g., prepare the parcels for the next day, do some maintenance work). Hence, we only pay for the time spent outside the depot.
- The endurance of the drone is limited and depends on its battery life.
- The drone launching and retrieving times are ignored as they are negligible, the batteries are supposed to be instantaneously swapped between flights.
- The delivery time depends on the vehicle involved (i.e., either a truck or a drone).

4.3.2 Truck-and-Drone Synchronization

Figure 4.1 displays the different types of truck-and-drone synchronization that are allowed at a node (i.e., the job location). Plain (resp. dashed) arcs represent truck (resp. drone) routes. The operations allowed at a node are as follows.

- (1) *Drone Retrieval*: after arriving at node j_3 , the truck retrieves the drone that has just delivered a parcel at j_2 .
- (2) *Drone Launch and Retrieval*: the truck launches the drone from j_3 , serves the corresponding job, and waits until the drone comes back to the same node.
- (3) *Drone Launch, then Leave*: the truck launches the drone from j_3 and continues on its route to j_6 .

At any node, a truck can perform one or more of these operations or none of them. To prevent the driver from experiencing an excessive waiting time at a node, the ELP forbids a truck from performing more than one type (2) operation at the same node.

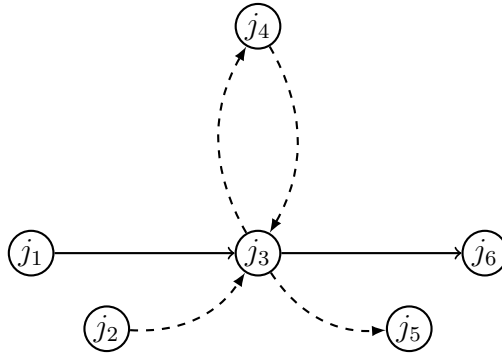


Figure 4.1: Different types of truck-and-drone synchronizations allowed at a job location.

4.3.3 Sets, Parameters, and Variables

Let $J = \{1, \dots, n\}$ be the set of jobs (i.e., delivery locations) and H the set of trucks. $\mathcal{T} \in \mathbb{N}$ denotes the time horizon (in minutes). Drivers can start their working day at time e_h and must return to the depot before l_h (hence, $\mathcal{T} = l_h - e_h$). From this point, the use of $\tilde{\cdot}$ differentiates parameters related to trucks and drones. For job $j \in J$: $p_j \in \mathbb{N}$ (resp. $\tilde{p}_j \in \mathbb{N}$) is its processing time (in minutes) when served by a truck (resp. by a drone); $(e_j, l_j) \in [0, \mathcal{T}]$ is its TW (the service must start within it); $a_j = 1$ indicates that the job can be served by a drone ($a_j = 0$ otherwise). Between two jobs (i, j) , the travelling (resp.

flying) distance is given by $d_{ij} \in \mathbb{R}$ (resp. $\tilde{d}_{ij} \in \mathbb{R}$), and the driving (resp. flying) time is denoted by $\tau_{ij} \in \mathbb{R}$ (resp. $\tilde{\tau}_{ij} \in \mathbb{R}$). The drone's endurance is given by \mathcal{E} .

The cost parameters are as follows. $c^f \in \mathbb{R}$ (resp. $\tilde{c}^f \in \mathbb{R}$) is the truck's (resp. drone's) daily fixed cost (in €). $c_{ij}^d \in \mathbb{R}$ is the cost of driving from $i \in J$ to $j \in J$ (in €). $c^t \in \mathbb{R}$ is the driver's wage (in €/h), and $\tilde{c}^t \in \mathbb{R}$ is the flying cost (in €/h). For drones, the flying time is the only variable cost, whereas for trucks and their assigned drivers, the variable costs include both the truck's fuel consumption, which is proportional to the driving distance, and the driver wage, which is proportional to the time spent outside the depot.

To handle the synchronization of trucks and drones, we create virtual nodes for each job to distinguish the time at which the drone is launched, the time at which it is retrieved, and the time at which the job is served. Accordingly, drones are launched and retrieved at virtual nodes. Indeed, considering the example displayed in Figure 4.1, the time at which the parcel is delivered by the truck at j_3 differs from the times at which the drone is retrieved from j_2 and relaunched towards j_5 . We introduce the following extended sets and variables. $J^- = \{n + 1, \dots, 2 \cdot n\}$ is the set of virtual entry nodes, whereas $J^+ = \{2 \cdot n + 1, \dots, 3 \cdot n\}$ is a set of virtual exit nodes. In our modeling, a truck that serves job j first visits $j + n$ (a drone can be launched and/or retrieved at this node), then visits j (to set the time at which the parcel is delivered at j) and finally visits $j + 2 \cdot n$ (a drone can be launched and/or retrieved at this node). Moreover, $\{0\}$ is the node representing the starting depot, and $\{3 \cdot n + 1\}$ represents the terminal depot. We introduce $V = J^- \cup J^+$, the set of virtual nodes, and $V^- = V \cup \{0\}$ and $V^+ = V \cup \{3 \cdot n + 1\}$. $A_1 = \{(i, j) \in (V \cup \{0\}) \times (V \cup \{3 \cdot n + 1\}), \text{ such as } i \neq j, \text{ if } i = 0 \text{ then } j \in J^-, \text{ if } i \in J^- \text{ then } j = i + n, \text{ if } i \in J^+ \text{ then } j \in J^- \cup \{3 \cdot n + 1\}\}$ is set of arcs that can be used by the trucks, it gathers all paths between virtual nodes except those that cannot be used in any solution (e.g., between two nodes in J^+). Respectively, $A_2 = \{(i, j, k) \in V^- \times J \times V^+, \text{ such as } i \neq j + n, k \neq j + 2 \cdot n, a_j = 1, \text{ and } \tilde{\tau}_{ij} + \tilde{\tau}_{jk} + \tilde{p}_j \leq \mathcal{E}\}$

is the set of all possible flying tours for drones (we removed some tours, e.g., a drone starting from a node in J^+ cannot be retrieved by the associated node in J^-).

We define the decision and the intermediate variables :

- $x_{ij}^h \in \{0, 1\}$ is equal to 1 if truck $h \in H$ travels from $i \in V^-$ to $j \in V^+$, 0 otherwise,
- $y_{ijk}^h \in \{0, 1\}$ is equal to 1 if the drone assigned to truck $h \in H$ visits $j \in J$ in a sub-route starting from $i \in V^-$ and arriving at $k \in V^+$, 0 otherwise,
- $z_{ij}^h \in \{0, 1\}$ is equal to 1 if truck $h \in H$ transports a drone from $i \in V^-$ to $j \in V^+$, 0 otherwise,
- $u_j^h \in \mathbb{R}$: time at which truck $h \in H$ leaves $j \in V$,
- $s_j \in \mathbb{R}$: the service time of job $j \in J$,
- $\tilde{w}_{ijk} \in \mathbb{R}$: flying time corresponding to the flight where the drone is launched at $i \in V^-$, serves customer $j \in J$, and is retrieved at $k \in V^+$ ($\tilde{w}_{ijk} = 0$ if such a flight does not exist),
- $r_h \in \{0, 1\}$ is equal to 1 if a drone is assigned to truck $h \in H$, 0 otherwise.

4.3.4 Mixed-Integer Linear Program

The MILP is solved for a fixed number of trucks. To minimize the number of trucks, the MILP is launched iteratively, reducing the number of trucks by one each time. For a given number of trucks, Objective (4.1) minimizes the number of drones, the distance traveled by trucks, the driver's completion times (i.e., salaries), and the drones' flying times.

$$\min \sum_{h \in H} \tilde{c}^f \cdot r_h + \sum_{h \in H} \sum_{(i,j) \in A_1} c_{ij}^d \cdot x_{ij}^h + \sum_{h \in H} c^t \cdot (u_{3,n+1}^h) + \sum_{(i,j,k) \in A_2} \tilde{c}^t \cdot \tilde{w}_{ijk} \quad (4.1)$$

Constraints (4.2) ensure that each job is completed exactly once by either a truck or a drone. The left component indicates that if a truck enters a virtual entry node ($j+n \in J^-$), it serves the corresponding job. The right component checks whether the considered job belongs to a drone's flight route.

$$\sum_{h \in H} \left(\sum_{(i,j+n) \in A_1} x_{i,j+n}^h + \sum_{(i,j,k) \in A_2} y_{ijk}^h \right) = 1 \quad j \in J \quad (4.2)$$

Various vehicle-flow constraints must be satisfied. Figure 4.2 illustrates the paths that can be followed by a drone. If a drone is retrieved or launched at node $i \in V$, a truck route must pass by i . Constraints (4.3) ensure that a drone arriving at $i \in V$ by flying or being transported in a truck must then exit the node by either flying or being transported. Similarly, Constraints (4.4) and (4.5) prevent a drone from arriving at (or exiting from) a node $i \in V$ by two different paths. Constraints (4.6) force a truck arriving at a virtual node to ultimately leave this node. Constraints (4.7) force a truck that leaves the depot to return to the depot at the end of its tour. Constraints (4.8) state that a truck can only exit the depot by one arc. Constraints (4.9) ensure the en-route synchronization of drones and trucks. Constraints (4.10) indicate whether the drone assigned to truck $h \in H$ is engaged.

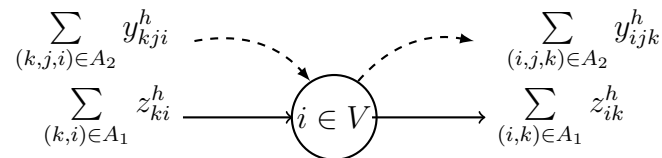


Figure 4.2: Path consistency for a drone.

$$\sum_{(k,i) \in A_1} z_{ki}^h + \sum_{(k,j,i) \in A_2} y_{kji}^h = \sum_{(i,j,k) \in A_2} y_{ijk}^h + \sum_{(i,k) \in A_1} z_{ik}^h \quad i \in V, h \in H \quad (4.3)$$

$$\sum_{(k,i) \in A_1} z_{ki}^h + \sum_{(k,j,i) \in A_2} y_{kji}^h \leq 1 \quad i \in V, h \in H \quad (4.4)$$

$$\sum_{(i,j,k) \in A_2} y_{ijk}^h + \sum_{(i,k) \in A_1} z_{ik}^h \leq 1 \quad i \in V \cup \{0\}, h \in H \quad (4.5)$$

$$\sum_{(j,i) \in A_1} x_{ji}^h = \sum_{(i,j) \in A_1} x_{ij}^h \quad i \in V, h \in H \quad (4.6)$$

$$\sum_{(0,i) \in A_1} x_{0i}^h = \sum_{(i,3 \cdot n+1) \in A_1} x_{i,3 \cdot n+1}^h \quad h \in H \quad (4.7)$$

$$\sum_{(0,i) \in A_1} x_{0i}^h \leq 1 \quad h \in H \quad (4.8)$$

$$z_{ij}^h \leq x_{ij}^h \quad (i,j) \in A_1, h \in H \quad (4.9)$$

$$\tilde{r}_h \geq \sum_{(i,j,k) \in A_2} y_{i,j,k} \quad j \in J, h \in H \quad (4.10)$$

Various temporal constraints must be satisfied. At a node in $J^- \cup J^+$, a truck can retrieve or launch a drone, perform both of these tasks, or do nothing. Furthermore, a truck must leave the corresponding node after all these operations take place. We set $M_1 = \tau_{max} + l_{max}$, $M_2 = \tilde{\tau}_{max} + l_{max}$, and $M_3 = \tilde{\tau}_{max} + \tilde{p}_{max} + l_{max}$. Constraints (4.11) require truck $h \in H$ to leave node $j \in V$ after its arrival. Constraints (4.12) (resp. (4.13)) ensure that service occurs after the truck's (resp. the drone) arrival. Constraints (4.14) force truck $h \in H$ to leave the node after completing the associated service. Constraints (4.15) allow truck $h \in H$ to leave the node after its assigned drone arrives. Constraints (4.16) compute the drone's flying time, i.e., the time length between the launch and the retrieval. Constraints (4.17) forbid any flight from having a longer duration than the drone's endurance. Constraints (4.18) require that the service times correspond to their

associated TWs.

$$u_j^h \geq u_i^h + \tau_{ij} - M_1 \cdot (1 - x_{ij}^h) \quad (i, j) \in A_1, h \in H \quad (4.11)$$

$$s_j \geq u_{j+n}^h \quad j \in J, h \in H \quad (4.12)$$

$$s_j \geq u_i^h + \tilde{\tau}_{ij} - M_2 \cdot \left(1 - \sum_{(i,j,k) \in A_1} y_{ijk}^h\right) \quad j \in J, i \in V^- \quad (4.13)$$

$$u_{j+2-n}^h \geq s_j + p_j \quad j \in J, h \in H \quad (4.14)$$

$$u_k^h \geq s_j + \tilde{p}_j + \tilde{\tau}_{jk} - M_3 \cdot \left(1 - \sum_{(i,j,k) \in A_2} y_{ijk}^h\right) \quad j \in J, k \in V^+, h \in H \quad (4.15)$$

$$\tilde{w}_{ijk} \geq u_k^h - u_i^h - l_{max} \cdot \left(1 - \sum_{h \in H} y_{ijk}^h\right) \quad i \in V^-, j \in J, k \in V^+ \quad (4.16)$$

$$\tilde{w}_{ijk} \leq \mathcal{E} \quad i \in V^-, k \in V^+, j \in J \quad (4.17)$$

$$e_j \leq s_j \leq l_j \quad j \in J \quad (4.18)$$

4.4 Solution Methods for the MC-VRPTW-D

No papers have proposed a solution method for the MC-VRPTW-D in its full complexity (i.e., multiple trucks with embedded drones, parcel delivery under TW constraints, and a global cost function to be minimized). The ALNS is introduced in Section 4.4.1. Next, a RFCS heuristic is proposed in Section 4.4.2. Finally, Section 4.4.3 presents the algorithm used to build an initial solution.

4.4.1 Adaptive Large Neighborhood Search (ALNS)

The ALNS proposed for the MC-VRPTW-D is given in Algorithm 5. Starting from a feasible solution, at each iteration, a neighbor solution s' is generated from the current solution s by removing and re-inserting several jobs (LNS, [Shaw, 1998]). The search moves from s to s' if s' improves s , or with a probability $e^{-(c(s')-c(s))/T}$ that depends on

the deterioration of solution s' when compared to s : $c(s)$ is the cost of solution s , and T is an exogenous parameter, called the temperature, that decreases with the execution time. Step (5) of Algorithm 5 describes the mechanism used to update the temperature, where T_0 (resp. T_f) is a parameter that specifies the initial (resp. final) temperature. This corresponds to the well-known *Metropolis criterion* employed in *Simulated Annealing* (SA) [Kirkpatrick et al., 1983].

ALNS combines multiple LNS heuristics by considering a pool \mathcal{R} (resp. \mathcal{I}) of removal (resp. insertion) heuristics. For each heuristic $i \in \mathcal{I} \cup \mathcal{R}$, π_i denotes the score obtained during the last η iterations (the score increases each time the heuristic has been used to find a solution accepted by the metropolis criteria); ω_i is the weight assigned to the heuristic, the greater this weight is, the higher the probability for the heuristic to be selected for the next step of Algorithm 5. The weight ω_i of a heuristic is updated each η iterations according to the formula: $\omega_i = (1 - r) \cdot \omega_i + r \cdot \pi_i$, where $r \in [0, 1]$ is a parameter called the learning rate. At each iteration, the probability of selecting the insertion heuristic $i \in \mathcal{I}$ (resp. $i \in \mathcal{R}$) is given by: $\omega_i / \sum_{i' \in \mathcal{I}} \omega_{i'}$ (resp. $\omega_i / \sum_{i' \in \mathcal{R}} \omega_{i'}$). At each step of ALNS, the number q of jobs to be removed and reinserted is randomly chosen. ALNS stops after a given maximum execution time t_{max} .

Algorithm 5 takes as input the largest execution time t_{max} , the initial and final temperature T_0 and T_f , the learning rate r , the size of a segment η (i.e., the number of iterations before updating the heuristics' weights), the largest absolute number of jobs (resp. percentage of jobs) q_{max} (resp. p_{max}) that can be removed at each iteration, and the reward attributed to successful heuristics σ . Preliminary experiments have led to the following parameter setting: ($q_{max} = 35$, $\eta = 20$, $r = 0.3$, $\sigma = 100$); T_0 (resp. T_f) is chosen so that at the beginning of ALNS, a deterioration of 10% (resp. 0.01%) is accepted with a probability of 50% (resp. 0.01%).

Algorithm 5 Adaptive Large Neighborhood Search (ALNS)

Input: initial solution s , t_{max} , (T_0, T_f) , r , η , q_{max} , p_{max} , σ .

Initialization: set $q_m = \min\{p_{max} \cdot |J|, q_{max}\}$; set $\pi_i = 0$, $\forall i \in \mathcal{I} \cup \mathcal{R}$; set $\omega_i = \sigma$, $\forall i \in \mathcal{I} \cup \mathcal{R}$; set $T = T_0$;

While the execution time is lower than t_{max} , **do:**

- (1) Select one removal heuristic and one insertion heuristic. The heuristic $i \in \mathcal{R}$ has a probability $\omega_i / \sum_{i' \in \mathcal{R}} \omega_{i'}$ of being selected, similarly, heuristic $i \in \mathcal{I}$ has a probability $\omega_i / \sum_{i' \in \mathcal{I}} \omega_{i'}$ of being selected.
 - (2) Select randomly the number $q \in [2, q_m]$ of jobs to be removed.
 - (3) Generate s' from s according to the selected removal and insertion heuristics (LNS procedure).
 - (4) Move or not the search from s to s' with respect to the Metropolis criterion.
 - (5) Update the search parameters:
 - update the score π_i of the heuristic used, set $\pi_i = \pi_i + \sigma$ if heuristic $i \in \mathcal{I} \cup \mathcal{R}$ has been used and if solution s' is accepted;
 - update the temperature: set $T = T_f + (T_0 - T_f) \cdot \left(\frac{t_e - t_{max}}{t_{max}}\right)^{10}$, where t_e is the current execution time;
 - update the score of each heuristic: each η iterations, update the weight ω_i of each heuristic i , set $\omega_i = (1 - r) \cdot \omega_i + r \cdot \pi_i$. Set $\pi_i = 0$, $\forall i \in \mathcal{I} \cup \mathcal{R}$.
-

Removal Heuristics

The following removal heuristics are proposed.

- (1) *Random Removal*. The jobs to be removed are randomly selected.
- (2) *Related Removal*. As stated in [Shaw, 1998], it is likely to be easier to reinsert jobs that are somehow related to each other. Accordingly, the jobs being removed are sequentially selected, and the probability of the next job to be removed directly depends on its relatedness to one of the already removed jobs. Equation (4.19) describes the function $R(j, j')$ used to define the relatedness of two jobs, j and j' . As $R(j, j')$ decreases, the jobs become more closely related. $R(j, j')$ takes into account geographical aspects (i.e., distance), temporal dimensions (i.e., service time and TW), and the current solution dispatch ($\mathbb{1}_{jj'} = 1$, if j and j' are served by the same vehicle, 0 otherwise). At each step, the *Related Removal* heuristic randomly selects one of the already removed jobs, say j_{rm} . All the jobs still belonging to the solution are then ranked in a list, L , from the most to the least related to j_{rm} . Finally, we select the job at position $L[y^\rho \cdot |L|]$, where y is randomly chosen in $[0, 1]$, and ρ is a parameter. Preliminary experiments have led to the following parameter tuning: $(\alpha, \beta, \gamma, \delta, \rho) = \left(1/\max_{(j,j')} d_{jj'}, 1/(l_h - e_h), 1/(l_h - e_h), 1, 0.89 \right)$.

$$R(j, j') = \alpha \cdot d_{jj'} + \beta \cdot |s_j - s_{j'}| + \gamma \cdot |l_j - l_{j'}| + \delta \cdot \mathbb{1}_{jj'} \quad (4.19)$$

- (3) *Worst Removal*. The jobs that have the largest contribution to the global cost function have a higher probability of being removed. As for the *Related Removal* heuristic, the jobs are ranked according to their contribution in the global cost function and selected accordingly.

Insertion Heuristics

Let J^{out} be the set of jobs to be reinserted into a truck-and-drone solution. Let c_{jh}^H (resp. c_{jh}^L) be the cost of inserting $j \in J^{out}$ at its cheapest position in the route of truck $h \in H$ (resp. in the schedule of the drone assigned to truck $h \in H$). $c_{jh} = \min\{c_{jh}^H, c_{jh}^L\}$ is, hence, the cheapest option when inserting j into the solution; we set $c_{jh} = \infty$ if no feasible insertion position can be found. \bar{c}_{jl} designates the l^{th} lowest cost for the insertion of job $j \in J^{out}$ on a route. We consider the following insertion heuristics.

- (1) *Best Insertion*. Jobs that minimize the insertion cost (i.e., have the lowest \bar{c}_{j1}) are inserted first.
- (2) *k-Regret Insertion*. This heuristic minimizes the regret of not inserting a job in the early stages of the procedure by first inserting jobs with the highest k -regret. The k -regret of job $j \in J^{out}$ is the difference between the k^{th} smallest insertion cost (\bar{c}_{jk}) and the smallest insertion cost (\bar{c}_{j1}). We have used $k \in \{1, 2\}$ in our experiments.
- (3) *Drone-First Insertion*. This heuristic is similar to the *Best Insertion* heuristic, but we only consider the drone insertion cost. More precisely, we only compute c_{jh}^L for each $j \in J$ and $h \in H$, and we select the best insertion based on these values. We compute c_{jh}^H for jobs that could be inserted into a drone schedule and select the best insertion as in the *Best Insertion* heuristic.

4.4.2 Route-First-Cluster-Second (RFCS)

The pseudocode of RFCS is given in Algorithm 6. It acts as a decomposition algorithm that first builds truck routes and then assigns jobs to drones. We investigate this procedure as it has been extensively used in the related literature [Murray and Chu, 2015, Agatz et al., 2018, Ham, 2018]. The proposed RFCS builds a VRP solution with an ALNS

relying on [Pisinger and Ropke, 2007]. For VRP, ALNS has been acknowledged to be one of the most efficient algorithms. In the second phase, RFCS attempts to improve the solution by sequentially assigning some jobs to drones within a descent local search framework (i.e., a modification is accepted only if it can reduce the costs). The solution is modified by removing one job and reinserting it at its best position (i.e., best insertion heuristic for one single job). Using the most efficient VRP algorithms to build the initial truck routes, the RFCS procedure is, hence, able to quickly deliver competitive solutions that are at least as good as the solutions involving trucks only. The algorithm stops when no further savings can be achieved by reassigning a job in the solution.

Algorithm 6 Route-First-Cluster-Second (RFCS) algorithm

Input: set J of jobs, considered network.

- (1) **Route First:** use ALNS to serve all customers with trucks only. Let s be the resulting VRP solution.
- (2) **Cluster Second:** assign customers to drones if the objective function can be improved. **For** each truck $h \in H$ of solution s that does not transport a drone, **do:**
 - (a) Assign a drone D to truck h , and let s^h denote the resulting solution.
 - (b) **While** a saving is encountered thanks to the use of D , **do:**
perform the best *reassignment move*, where a reassignment move consists in relocating a job $j \in J$ at its best position in s^h (either using a drone or a truck).
 - (c) **If** all the savings achieved thanks to D are larger than the fixed cost of the drone, set $s = s^h$.

Return s

4.4.3 Initial Solution

Starting from an empty solution where no job is performed, the initial solution is generated in two steps. First, the *Best Insertion* heuristic is used until all the jobs have been inserted. At each iteration, if a job cannot be inserted into existing routes, a new truck and drone tandem is added. To reduce the number of trucks used in the obtained VRP solution s_0 , a modified version of ALNS is launched starting from s_0 . In this version of ALNS, a drone or truck is removed from the solution each time a feasible solution is found. All the jobs

performed by the removed vehicles are stored in a request pool. A solution is said to be feasible when the request pool is empty (i.e., all jobs are served). Conversely, when the request pool is not empty, the solution is said to be unfeasible. The insertion mechanism aims to reinsert the jobs stored in the request pool at the same time as those removed in the ALNS process.

4.5 Speed up the Insertion Mechanism

As described in Section 4.4, ALNS strongly relies on the insertion mechanisms to iteratively improve the solution. In the context of MC-VRPTW-D, insertion mechanisms exhibit much greater complexity than standard VRP situations. To tackle realistic instances, ALNS must employ fast procedures to evaluate the feasibility and cost of a solution obtained after an insertion. This section proposes heuristics to significantly speed up the insertion mechanisms.

4.5.1 Modeling Aspects and Notation

Let \mathcal{R}_h be a truck-and-drone route. \mathcal{R}_h is an ordered set of nodes in $J \cup V \cup \{0, 3 \cdot n + 1\}$. With each node $i \in \mathcal{R}_h$, we associate the time X_i at which the truck or the drone visits it. A solution is feasible if Constraints (4.20 – 4.29) are satisfied. Constraints (4.20) frame the traveling time from one node to another in \mathcal{R}_h . Constraints (4.21 – 4.22) create dependencies between node j and its assigned virtual nodes when it is served by a truck. Constraints (4.23) refer to the time constraints related to drone endurance. Constraints (4.24 – 4.25) are active when a drone serves j , accounting for the travel and job processing time. Constraints (4.26 – 4.27) refer to the job’s TW satisfaction. Constraints (4.28 –

4.29) refer to the worker's longest duration spent outside of the depot.

$$X_j - X_i \geq \tau_{ij}, \quad \text{if the truck goes from } i \in V^- \text{ to } j \in V^+ \text{ in } \mathcal{R}_h, \quad (4.20)$$

$$X_j - X_{j+n} \geq 0, \quad \text{if the truck serves } j \in J, \quad (4.21)$$

$$X_{j+2n} - X_j \geq p_j, \quad \text{if the truck visits } j \in J, \quad (4.22)$$

$$X_i - X_k \geq -E, \quad \text{if the drone is launched at } i \in V^- \text{ and retrieved at } k \in V^+, \quad (4.23)$$

$$X_i - X_j \geq \tilde{\tau}_{ij}, \quad \text{if the drone is launched at } i \in V^- \text{ to visit } j \in J, \quad (4.24)$$

$$X_k - X_i \geq p_j + \tilde{\tau}_{jk}, \quad \text{if the drone is retrieved at } k \in V^+ \text{ after serving } j \in J, \quad (4.25)$$

$$X_j - 0 \geq e_j, \quad \forall j \in J, \quad (4.26)$$

$$0 - X_j \geq -l_j, \quad \forall j \in J, \quad (4.27)$$

$$0 - X_0 \geq e_h, \quad \forall j \in J, \quad (4.28)$$

$$0 - X_{3n+1} \geq -l_h, \quad \forall j \in J. \quad (4.29)$$

To efficiently tackle these temporal constraints, we model \mathcal{R}_h with a precedence graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. $\mathcal{V} = J \cup V \cup \{0, 3 \cdot n + 1\} \cup O$ designates all the physical and virtual nodes. O denotes the origin node required to model TW in Constraints (4.26 – 4.29). \mathcal{E} represents all the temporal constraints detailed in Constraints (4.20 – 4.29). More precisely, for any constraint of type $X_j - X_i \geq t_{ij}$, where $t_{ij} \in \mathbb{R}$, an arc from i to j with a weight t_{ij} is added to \mathcal{E} .

In the context involving the synchronization of transportation resources when passenger transfers can occur on the routes, [Masson et al., 2013] has shown that determining the feasibility of a solution is equivalent to checking for the presence of a cycle of positive length in the precedence graph. To do so, the Bellman-Ford-Tarjan algorithm (BFCT) [Cherkassky et al., 2009] is acknowledged to be the most efficient. In our context, it has a computational complexity of $\mathcal{O}(n^2)$.

For a feasible truck-and-drone route \mathcal{R}_h , ϵ_i and λ_i denote the bounds for X_i (i.e., $X_i \in$

$[\epsilon_i, \lambda_i]$). Specifically, i cannot be visited before ϵ_i , and leaving later than λ_i leads to the violation of a future TW. w_{ik} denotes the waiting time for the truck between two nodes $i \in \mathcal{V}$ and $k \in \mathcal{V}$ in the precedence graph. Relying on [Masson et al., 2013], computing ϵ_i , λ_i and w_{ik} is done in $\mathcal{O}(n^2)$.

4.5.2 Greedy Algorithm to Insert a Job at its Best Position

Algorithm 7 describes the greedy algorithm used to compute the best insertion position for job j on a truck-and-drone route \mathcal{R}_h . For all insertion positions in a truck or drone schedule, it first checks if the resulting solution is feasible (steps (1b) and (2b)), then it computes the additional costs associated with the generated solution (steps (1c) and (2c)).

In the context of MC-VRPTW-D, checking the feasibility of an updated solution is rather complicated as it requires checking that the drone’s endurance constraint is satisfied, in addition to the non-violation of TW constraints. For example, the drone’s retrieval could be delayed after its insertion into a truck route. Such a delay could violate the drone endurance constraint. In this case, the drone’s launch might be delayed to ensure the feasibility of the endurance constraint, but this delay precludes the constants of the graph (ϵ, λ, w) being used further. This results in heavier computational requirements to check the feasibility of an updated solution and evaluate its cost. Moreover, compared with standard VRP situations, the number of insertions to be tested increases in the MC-VRPTW-D case. Indeed, in addition to insertions into truck schedules, we also have to consider insertions into drone schedules (step (2)).

Recomputing the whole solution for all insertion positions precludes ALNS from visiting a sufficiently large number of neighbor solutions per iteration. To avoid this drawback, Section 4.5.3 (resp. Section 4.5.4) proposes an algorithm that: (1) checks in constant time

whether an insertion in a drone (resp. a truck) schedule is feasible and (2) evaluates the resulting costs.

Algorithm 7 Finding the best insertion position for job j in truck-and-drone route \mathcal{R}_h

Input: feasible truck-and-drone route \mathcal{R}_h ; job $j \in J$ that is not served in \mathcal{R}_h .

Initialization: set $c_{jh}^H = \infty$; set $c_{jh}^L = \infty$.

(1) Compute the cost c_{jh}^H of assigning j to the truck.

For each node $i \in J^+ \cup \{O\}$ visited in \mathcal{R}_h , **do the following:**

- (a) Insert j after i in \mathcal{R}_h .
- (b) Check the feasibility of the generated route.
- (c) Compute the cost Δ_{ih}^H of the generated route (set $\Delta_{ih}^H = \infty$ if the solution is not feasible).
- (d) Set $c_{jh}^H = \min\{c_{jh}^H, \Delta_{ih}^H\}$.

(2) If $a_j = 1$, compute the cost c_{jh}^L of assigning j to the drone.

For each node $(i, k) \in (V \cup \{O\}) \times (V \cup \{3 \cdot n + 1\})$ **such that** (i is visited before k in \mathcal{R}_h) **and** (the drone is on-board from i to k), **do the following:**

- (a) Insert j into the drone's scheduling such that the drone is launched from i and retrieved at k in \mathcal{R}_h .
- (b) Check the feasibility of the generated route.
- (c) Compute the cost Δ_{ikh}^L of the generated route (set $\Delta_{ikh}^L = \infty$ if the solution is not feasible).
- (d) Set $c_{jh}^L = \min\{c_{jh}^L, \Delta_{ikh}^L\}$.

Return $c_{jh} = \min\{c_{jh}^H, c_{jh}^L\}$ **and** the corresponding position in \mathcal{R}_h .

4.5.3 Insertion of a Job into a Drone's Schedule

Algorithm 8 allows avoiding the systematic use of BFCT at step (2b) of Algorithm 7. It evaluates whether launching a drone from node i and then retrieving it at node k after serving client j results in a feasible solution. Furthermore, it computes the cost difference of the associated solution. Note that the number of pairs (i, k) to test grows in $\mathcal{O}(n^2)$.

The precedence graph is used to check the feasibility of the insertion (step (1)). More precisely, we check whether the arcs added when launching a drone at node i and retrieving

it at node k induce a cycle of positive length in the precedence graph. Figure 4.3 shows a subgraph extracted from the precedence graph after inserting job j into the solution when a drone is launched at node $i \in V^-$ and retrieved at node $k \in V^+$. The plain arcs with weight ϵ_i and λ_i (resp. ϵ_k and λ_k) denote the earliest and latest arrival times at node i (resp. at node k) and show the paths existing in the solution before the insertion. Dashed lines represent the arcs added in the precedence graph after proceeding to the insertion of client j between nodes i and k . Figure 4.3 is helpful to understanding where a cycle of positive length could appear in the precedence graph after the insertion.

Algorithm 8 also computes the service time ϵ_j at client j , and the induced delay δ_i (resp. δ_k) at node i (resp. k). Steps (1a) and (1b) are initialization steps. Step (1b) verifies whether i is already a retrieval point for the drone. In this case, the delay at i must be bounded so that the endurance of the drone arriving at i is not violated (step (1d)). The service time at j and the delay at k are computed in steps (1c) and (1d), respectively. These computations are similar to the *Forward Time Slack* (FTS) presented in [Savelsbergh, 1992]. The remaining part of step (1d) focuses on the drone flight endurance constraint. If the delay at k forces the drone to remain in flight longer than endurance allows, we delay its launch time (δ_i). If this delay does not lead to a constraint violation, we return to step (1c) to recompute the service time at j and the delay at k .

Next, if an insertion is found feasible (i.e., if there exists $\epsilon_j \geq 0$, $\delta_i \geq 0$ and $\delta_k \geq 0$), Algorithm 8 computes the increased cost associated with the newly generated solution. There are two sources of additional costs: costs related to powering the drone (step (2b)) and costs associated with the driver salary in the case the driver arrives later at the depot (step (2c)) (when inserting a job in a drone schedule, the driver's route does not change, hence the fuel expenses do not change neither). In this case, we compare the delay induced at k with the total waiting time for the driver between k and the depot.

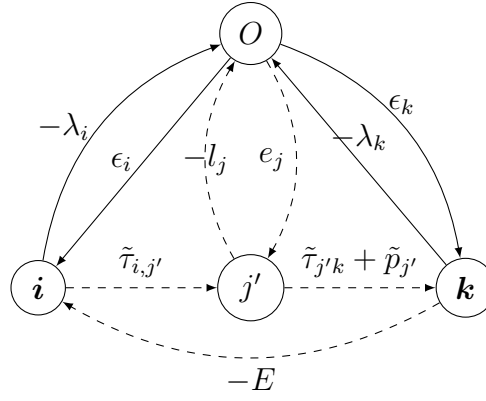


Figure 4.3: Subgraph of the precedence graph after the insertion of job j into the schedule of a drone launched at i and retrieved at k .

Algorithm 8 Check the feasibility and evaluate the cost of serving a client with a drone.

Input: truck-and-drone route \mathcal{R}_h ; job $j \in J$; launch and retrieval points for the drone: $(i, k) \in \mathcal{R}_h$.

(1) Check the feasibility:

- (a) Set $\delta_i = 0$ (delay at node i);
- (b) **If** a drone flight arrives at node i (let $\sigma_s \in \mathcal{R}_h$ be the launch node corresponding to this flight), set ϵ_{σ_s} to be the launching time at σ_s ; **Else:** set $\epsilon_{\sigma_s} = \infty$, if no drone flight arrives at i .
- (c) Set the service time at j : $\epsilon_j = \max\{\epsilon_i + \delta_i + \tilde{\tau}_{ij}, e_j\}$; **return** ‘infeasibility’ if $\epsilon_j > l_j$.
- (d) Set the delay at k : $\delta_k = \max\{0, \epsilon_j + p_j + \tilde{\tau}_{jk} - \epsilon_k\}$.
If $\epsilon_k + \delta_k > \lambda_k$, **return** ‘infeasibility’.
If $\epsilon_k + \delta_k - (\epsilon_i + \delta_i) > E$, **do the following:**
 - If** $\delta_i = 0$, set $\delta_i = \epsilon_k + \delta_k - E - \epsilon_i$.
 - If** $(\epsilon_i + \delta_i > \lambda_i)$ **or** $(\epsilon_i + \delta_i - \epsilon_{\sigma_s} > E)$, **return** ‘infeasibility’.
 - Else**, go to step (1c).**Else return** ‘infeasibility’.

(2) Evaluate the cost using the obtained values $(\epsilon_i, \delta_i, \delta_k)$:

- (a) Compute the flying cost: $\Delta^{flying} = (\epsilon_k + \delta_k - (\epsilon_i + \delta_i)) \cdot \tilde{c}^t$.
 - (b) Compute the required augmentation of the driver’s salary: $\Delta^{salary} = \max(0, \delta_k - w_{k,3 \cdot n + 1}) \cdot c^t$.
 - (c) **Return** $\Delta_{ikh}^L = \Delta^{flying} + \Delta^{salary}$.
-

4.5.4 Inserting a Job into a Truck's Schedule

Algorithm 9 checks the feasibility of the solution after inserting job j into a truck's route and computes the associated additional costs. Step (1a) checks whether a drone is in flight when the insertion is performed. Let σ_s and σ_e be the launch and retrieval points of a such flight on the route, respectively. Step (1d) verifies whether the drone's endurance constraint is violated. Note that we are considering a conservative case here. Indeed, a delay may have to be introduced at σ_s to ensure the drone's endurance constraint is satisfied (from σ_s to σ_e). In this situation, we would need to further propagate the delay through the previous drone flights as σ_s could be the retrieval node of a previous drone flight. Considering such cases could lead to having to update the whole solution, which would drastically slow down the insertion mechanisms and decrease the overall efficiency of the ALNS. Preliminary experiments have shown that precluding these situations is overcompensated by the increased number of iterations achieved by the ALNS.

In step (2), Algorithm 9 computes the additional costs associated with the newly generated solution. It takes into account the increased driving distance (step (2a)), the increased time en-route for the driver (step (2b)), and the increased flight time for the drone (step (2c)) when it is also affected by the performed insertion.

4.5.5 Complexity of an Insertion

Table 4.2 compares the complexity associated with finding the best insertion position for a job in a VRP solution and in a truck-and-drone solution. 'Nb. Insertions' counts the total number of insertions to be tested and 'Complexity' gives the computational complexity resulting from checking the feasibility of an insertion and evaluating the associated solution cost. n denotes the number of jobs in the solution. 'BFCT' stands for the previously mentioned straightforward approach that recomputes the whole solution

Algorithm 9 Check the feasibility and evaluate the cost of serving a client with a truck.

Input: truck-and-drone route \mathcal{R}_h ; job $j \in J$; launch and retrieval points for the drone: $(i, k) \in \mathcal{R}_h$.

(1) Check the feasibility:

- (a) **Identify** if a drone on flight: let $(\sigma_s, \sigma_e) \in \mathcal{R}_h$ the launch and retrieve points of a drone such that: σ_s is visited before i , σ_e is visited after i in \mathcal{R}_h .
Set the launch time ϵ_{σ_s} of the drone in \mathcal{R}_h ; **Set** $\epsilon_{\sigma_s} = \infty$ if such a flight does not exist.
Set the retrieval ϵ_{σ_e} time of the drone in \mathcal{R}_h ; **Set** $\epsilon_{\sigma_e} = 0$ if such a flight does not exist.
Set the wait time w_{i+1, σ_e} between $i+1$ and σ_e in \mathcal{R}_h ; **Set** $w_{i+1, \sigma_e} = \infty$ if such flight does not exist.
- (b) **Set** the service time at j : $\epsilon_j = \max\{\epsilon_i + \delta_i + \tau_{ij}, e_j\}$; **return** ‘infeasibility’ if $\epsilon_j > l_j$.
- (c) **Set** the delay at $i+1$: $\delta_{i+1} = \max\{0, \epsilon_j + p_j + \tilde{\tau}_{j, i+1} - \epsilon_{i+1}\}$; **return** ‘infeasibility’ if $\epsilon_{i+1} + \delta_{i+1} > \lambda_{i+1}$.
- (d) **Set** delay at σ_e : $\delta_{\sigma_e} = \max\{0, \delta_{i+1} - w_{i+1, \sigma_e}\}$; **return** ‘infeasibility’ if $\epsilon_{\sigma_e} + \delta_{\sigma_e} - \epsilon_{\sigma_s} > E$.

(2) Evaluate the cost with the obtained values $(\epsilon_i, \delta_i, \delta_{i+1}, \delta_e)$:

- (a) Compute the driving cost: $\Delta^{driving} = (c_{i,j}^d + c_{j,i+1}^d - c_{i,i+1}^d)$.
 - (b) Compute the increased driver salary: $\Delta^{salary} = \max(0, \delta_{i+1} - w_{i+1, 3 \cdot n+1}) \cdot c^t$.
 - (b) Compute the flying cost: $\Delta^{flying} = \delta_{\sigma_e} \cdot \tilde{c}^t$.
 - (d) **Return** $\Delta_{ih}^H = \Delta^{driving} + \Delta^{salary} + \Delta^{flying}$.
-

to check its feasibility. When inserting a job into a truck route, the number of insertions to be tested grows linearly with the number of jobs served by the truck. The number of possible insertions into a drone schedule grows with the square of the number of jobs to be inserted as we need to test all pairs (i, k) (i served before k) in the truck routes. In the VRP case, checking the feasibility can be done in constant time due to the FTS principle.

Table 4.2 shows that in the VRP case, the computational complexity required to find the best insertion position for a job grows with $\mathcal{O}(n)$. In the truck-and-drone case, our heuristics allow decreasing the computational complexity from $\mathcal{O}(n^4)$ (i.e., the straightforward approach that recomputes the whole solution for each tested insertion) to $\mathcal{O}(n^2)$. However, whereas our heuristics also allow evaluating the feasibility and the cost of an insertion in constant time, the number of insertion positions to be tested is larger in the truck-and-drone case than in the VRP case. As a result, for the same execution time, ALNS will be able to visit fewer neighbor solutions than in the VRP case.

Vehicle	Truck-only		Truck-and-drone		
	Nb. Insertions	Comp. Complexity	Nb. Insertions	Complexity BFCT	Complexity our algorithms
Truck	$\mathcal{O}(n)$	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(1)$
Drone	-	-	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(1)$

Table 4.2: Complexity comparison for insertion procedures.

4.6 Computational Experiments and Managerial Insights

The MILP and all algorithms have been coded in C++. The MILP is solved with CPLEX 12.4 (called using the Concert Technology). Computations were performed on a 2.2 GHz Intel Core i7 with 16 GB 1600 MHz DDR3 RAM. For the considered static day-ahead problem, the ELP limits the total execution time to 5 hours.

4.6.1 Instances

The instance set was built based on input from the ELP, with the aim of covering the various situations occurring in practice (i.e., situations involving urban and suburban territories). The jobs are randomly generated in a 25×25 km squared grid. In the considered transportation network, the Manhattan (resp. Euclidean) distance is considered for the trucks (resp. drones). The depot is located at the center of the grid. The working day begins at 8am and must end before 5pm. Expenses related to the drivers' salaries are proportional to the amount of time they spend away from the depot. The possible 2-hour TWs for the parcel deliveries are [8am, 10am], [9am, 11am], \dots , [3pm, 5pm]). Each truck (resp. drone) travels at an average speed of 30km/h (resp. 60km/h). When reaching the involved node, the parcel delivery has a duration of 3 (resp. 5) minutes when processed by a driver (resp. drone). More time is needed for the drone case as the customer has to unlock the parcel. All jobs cannot be performed by a drone. Hence, we consider the accessibility of the jobs by drone, which acknowledges the fact that some parcels are too heavy to be transported by a drone, or some customers are not eligible to receive a parcel by drone. For each instance, 5 configurations of job accessibility by drone are generated: 0% (i.e., when no job can be reached by drone, which corresponds to the VRP case), 25%, 50%, 75%, and 100% (i.e., when all the jobs can be performed by a drone).

We have generated several instances from $n = 10$ to $n = 100$ jobs. The smallest instances are used to compare the results of ALNS with those of the MILP, whereas the larger instances better capture real situations. Each instance is denoted as follows: Nn_{Ap_a-i} , where n is the number of jobs, p_a indicates the percentage of customers that can be served by drones, and i discriminates between the instances that share the same number of customers and the same percentage of jobs reachable by drones. Following this notation, $N50_{A75-1}$ is the first instance involving 50 customers, 75% of which can be served by drone. We have generated 71 instances: 21 small instances with $n \in [10, 25]$, 25 medium-sized instances with $n = 50$, and 25 larger instances with $n = 100$. TWs are only employed

for the realistic instances, involving 50 or 100 customers.

We consider the real cost structure provided by the ELP. While the absolute values of the various costs (i.e., fixed and variable; see Section 4.3.3) cannot be given because of confidentiality issues, we indicate the ratio of these costs between trucks and drones: the daily fixed cost for a truck is 5.6 times larger than the fixed cost for a drone; employing a driver for one hour is 9.2 times more expensive than using a drone for one hour; the cost per km driven per truck is 4.3 times cheaper than the cost of powering a drone for one hour.

4.6.2 Results

In this section, we first compare ALNS with CPLEX on the smallest instances, and we then compare ALNS with RFCS on the larger but realistic instances.

ALNS versus CPLEX (smaller instances)

Table 4.3 compares the performance of ALNS and CPLEX for the instances involving up to 25 jobs (without TWs). Columns ‘Obj.’ (resp. ‘Time [s]’) gives the value of the objective function in € (resp. the execution time, in seconds). The execution time is bounded to one hour for CPLEX. It is not reported in Table 4.3 as CPLEX was, for all instances, never able to prove the optimal solution within this time limit. For the MILP, column ‘LB’ indicates the value of the lower bound returned by CPLEX when optimality was not proven. We can observe that CPLEX turns out to be competitive for instances involving up to $n = 20$ jobs. Indeed, for instances with $n \in [10, 20]$ jobs, CPLEX (resp. ALNS) finds the best solution for 8 (resp. 15) instances over 18. However, for all 18 instances with $n \in [10, 20]$, the ALNS produced solutions with an average improvement

of 0.8% compared to CPLEX. Furthermore, the ALNS finds its best solution in less than 10 minutes for these instances, whereas CPLEX uses the entire one hour time budget. For larger instances ($n > 20$), ALNS significantly outperforms CPLEX on both the solution quality and speed. For instances involving $n = 25$ jobs, the ALNS improves CPLEX results on average by 11%. It is interesting to note that the more jobs are eligible to be served by drone, the less efficient is CPLEX (i.e., the gap between the solution returned by CPLEX and the lower bound increases with the percentage of jobs reachable by drone). Note that we have tested CPLEX for the 50-job instances, but no feasible solution was found.

Instance	CPLEX		ALNS	
	Obj.	LB	Obj.	Time [s]
N10_A50.1	118.6	115.6	118.6	30
N10_A75.1	118.6	95.9	118.6	30
N10_A100.1	101.5	84.3	101.7	20
N10_A50.2	117.3	114.8	117.3	10
N10_A75.2	112.4	91.3	112.2	15
N10_A100.2	112.4	85.6	112.1	25
N15_A50.1	136.0	84.9	136.0	12
N15_A75.1	128.8	76.8	128.2	30
N15_A100.1	114.4	68.2	117.0	47
N15_A50.2	130.3	80.9	131.6	55
N15_A75.2	119.2	72.6	118.8	165
N15_A100.2	119.2	70.1	117.1	180
N20_A50.1	148.4	83.3	147.5	365
N20_A75.1	131.7	75.1	132.1	182
N20_A100.1	133.9	65.8	132.1	242
N20_A50.2	134.3	79.0	132.9	235
N20_A75.2	130.6	72.13	128.6	156
N20_A100.2	140.6	67.2	127.7	370
N25_A50.1	162.7	87.1	161.8	401
N25_A75.1	154.0	74.1	140.2	750
N25_A100.1	169.2	65	139.5	559

Table 4.3: Comparison of ALNS and CPLEX for the smaller instances.

ALNS versus RFCS (larger instances)

Table 4.4 compares the results of RFCS and ALNS. ‘Obj.’ refer to the cost found by the corresponding method. ‘Time [h]’ indicates the average time (in hours) at which the best solution was found. Finally, column ‘% (RFCS)’ indicates the average percentage gap of the ALNS solution with respect to the RFCS solution. It is computed as $(f_{RFCS} - f_{ALNS})/f_{RFCS}$, where f_{RFCS} (resp. f_{ALNS}) denotes the cost of the solution returned by RFCS (resp. ALNS). The stopping criterion for the ALNS is when the execution time reaches 5 hours, and for the RFCS is when no more improvements can be made. Whereas the RFCS acts as a decomposition solution method that first builds the truck routes and then incorporates drone subtours into these routes, the ALNS acts as an integrated method that simultaneously builds trucks and drones’ routes. One can observe that although RFCS generates its best solutions more rapidly, its results can be significantly improved by ALNS as the average percentage gap is 8.8%.

Visualization of a Truck-and-Drone Solution

For instance *N10_A100_1* involving 10 customers, Figure 4.4 compares a truck-and-drone solution (right side) with the optimal truck-only solution (left side). To simplify the visualization, the truck route is denoted by a straight line from one customer to another, whereas it uses the Manhattan grid in reality. The right side of Figure 4.4 displays all types of synchronization that can happen between a truck and its assigned drone. In the truck-and-drone solution, the drone leaves the truck when it is located at the depot and flies to drop a parcel at job 9 before being refilled by the truck at job 2. Next, the truck transports the drone to job 0. At jobs 1 and 4, the truck acts as a delocalized depot. Indeed, at these nodes, all operations described in Section 4.3.2 take place: ‘Drone Retrieval’, ‘Drone Launch and Retrieval’, and ‘Drone Launch, then Leave’.

Instance	RFCS		ALNS		
	Obj.	Time [h]	Obj.	Time [h]	% (RFCS)
N50_A25_1	420.8	1.1	420.8	3.2	0.0%
N50_A25_2	423.5	1.4	423.5	1.8	0.0%
N50_A25_3	418.9	1.5	412.8	2.4	-1.5%
N50_A25_4	433.6	1.9	392.3	3.0	-9.5%
N50_A25_5	427.2	1.3	427.2	2.9	0.0%
N50_A50_1	406.1	1.2	380.8	3.4	-6.2%
N50_A50_2	423.5	1.5	408.5	2.1	-3.5%
N50_A50_3	420.9	1.7	400.6	2.5	-4.8%
N50_A50_4	433.6	2.0	371.0	3.1	-14.4%
N50_A50_5	424.3	1.4	407.5	3.4	-4.0%
N50_A75_1	385.1	1.4	344.3	3.0	-10.6%
N50_A75_2	387.8	1.5	345.7	3.5	-10.9%
N50_A75_3	396.6	1.7	359.2	3.4	-9.4%
N50_A75_4	381.3	2.2	357.2	3.8	-6.3%
N50_A75_5	374.8	1.5	339.6	5.2	-9.4%
N50_A100_1	354.0	1.4	310.8	3.9	-12.2%
N50_A100_2	346.7	1.6	319.7	4.6	-7.8%
N50_A100_3	369.4	1.8	327.8	3.8	-11.3%
N50_A100_4	343.6	2.3	312.9	3.9	-8.9%
N50_A100_5	342.2	1.6	302.9	4.9	-11.5%
N100_A25_1	627.2	3.3	622.3	3.8	-0.8%
N100_A25_2	650.6	3.0	643.1	3.0	-1.2%
N100_A25_3	623.9	3.8	622.9	4.0	-0.2%
N100_A25_4	646.7	2.9	633.2	3.5	-2.1%
N100_A25_5	633.3	2.8	625.3	2.2	-1.3%
N100_A50_1	627.2	3.4	586.5	4.4	-6.5%
N100_A50_2	647.4	3.2	586.6	4.4	-9.4%
N100_A50_3	623.9	3.9	594.4	4.6	-4.7%
N100_A50_4	645.2	3.2	583.7	4.3	-9.5%
N100_A50_5	633.3	2.9	597.3	4.6	-5.7%
N100_A75_1	582.3	3.7	537.4	4.4	-7.7%
N100_A75_2	548.3	3.3	446.0	5.0	-18.7%
N100_A75_3	530.3	4.1	436.6	5.0	-17.7%
N100_A75_4	602.5	3.3	558.2	4.8	-7.3%
N100_A75_5	577.9	3.0	522.5	4.8	-9.6%
N100_A100_1	520.6	3.8	409.6	3.8	-21.3%
N100_A100_2	520.5	3.6	413.5	4.5	-20.6%
N100_A100_3	530.3	4.2	385.7	5.0	-27.3%
N100_A100_4	543.2	3.5	429.7	2.9	-20.9%
N100_A100_5	500.3	3.1	411.5	4.3	-17.8%

Table 4.4: Comparison of ALNS and RFCS for the larger instances.

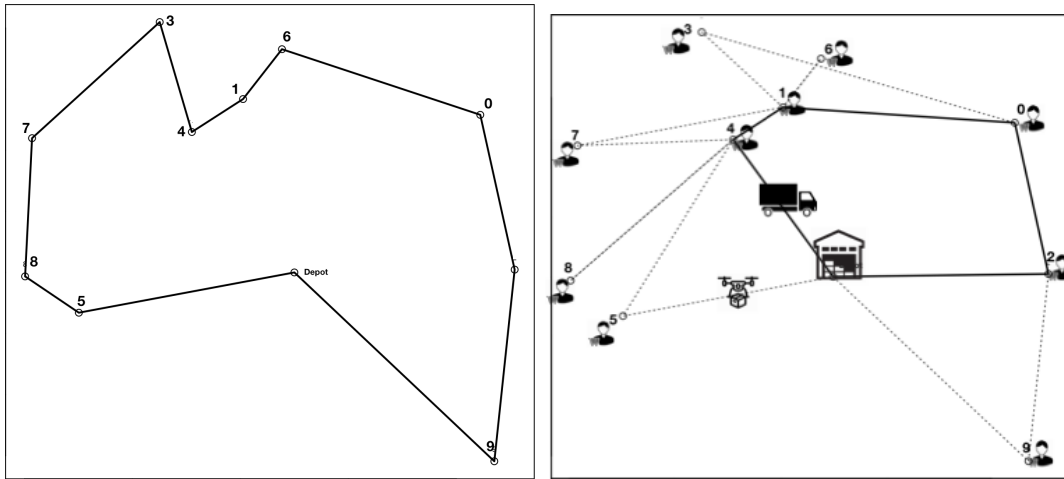


Figure 4.4: Truck-only (left side) versus truck-and-drone (right side) solutions. Plain (resp. dashed) lines are truck (resp. drone) trips.

4.6.3 Sensitivity Analysis of the Percentage of Jobs Reachable by Drone

Table 4.5 compares the best truck-and-drone solution found by ALNS for all instances involving 50 jobs or more and for different percentages of jobs reachable by drone (%A). If no job can be reached by drone, we present a VRP solution. Column ‘Cost’ gives the cost of the solution. Column ‘%(VRP)’ quantifies the percentage cost improvement achieved by the truck-and-drone solutions with respect to the associated truck-only solutions. This percentage gap is computed as $(c_d - c_{VRP})/c_{VRP}$, where c_d (resp. c_{VRP}) is the total cost of the truck-and-drone (resp. truck-only) solution. Column ‘T-Nb.’ (resp. ‘D-Nb.’) indicates the number of trucks (resp. drones) employed in the solutions. Column ‘T-Dist.’ (resp. ‘D-Dist.’) indicates the distance (in km) traveled by trucks (resp. drones). ‘T-Time’ (resp. ‘D-Time’) gives the time (in minutes) traveled by trucks (resp. drones). Finally, column ‘%J’ provides the percentage of jobs that are actually served by drone.

First, we can measure the significant cost reduction achieved when drones are eligible to serve jobs. Moreover, the gain increases with %A: it grows from 1.1% when %A = 25% to 35.6% when %A = 100%. Interestingly, for instances involving 100 jobs, some trucks can be replaced by drones when more than 75% of the jobs can be reached by drone. Such

replacement has a positive impact on the overall cost of the solution. As an example, for the instances $N100_A75_2$ and $N100_A100_2$, one truck is replaced by two drones.

However, there is a critical threshold for $\%A$ below which the use of drones is not cost effective. When $\%A = 25\%$, allowing the use of drones does not lead to better solutions for 60% of the considered instances. The returned solution by ALNS is, therefore, the VRP solution (see lines with $\%J = 0\%$). In such cases, the fixed costs associated with the drone use cannot be compensated by either the savings achieved for the truck's traveling distance or the driver's working hours. Interestingly, following the ELP's cost structure, the drone's fixed costs start to be compensated with $\%J = 5\%$. In line with our results, for $n \in \{5, 10\}$, [Pugliese and Guerriero, 2017] also show that truck-and-drone solutions are efficient for some instance characteristics (i.e., when drone flying costs are at least 9 times cheaper than truck driving costs). This work extends their findings as we consider instances involving up to $n = 100$ jobs, and we account for the drone's and the truck's fixed costs, as well as driver wages.

We can also observe that it is not always beneficial to assign a drone to each truck. Indeed, for $N50_A25_3$ and $N50_A50_3$, the solution uses 2 trucks but only 1 drone. In such a case, the fixed costs incurred by using a second drone would not be overcompensated by the cost reduction for the trucks (fixed and variable costs). In contrast with the literature (e.g., [Sacramento et al., 2019] consider that either each truck is equipped with one drone or none), our findings highlight the importance of considering a flexible fleet of vehicles, where trucks only and trucks equipped with drones coexist in the solution.

4.6.4 Cost Structure of Truck-and-Drone Solutions

Figure 4.5 displays the different aggregated costs for $n \in \{50, 100\}$. From the bottom to the top of each bar, the first two values are the fixed costs associated with the use of trucks

Instance	Cost	%(VRP)	T-Nb.	D-Nb.	T-Dist.	D-Dist.	T-Time	D-Time	%J
N50_A0_1	420.8	0.0%	2	0	349.7	0.0	15.9	0.0	0%
N50_A25_1	420.8	0.0%	2	0	349.7	0.0	15.9	0.0	0%
N50_A50_1	380.8	-9.5%	2	2	245.8	188.4	14.3	3.4	26%
N50_A75_1	344.3	-18.2%	2	2	199.3	286.9	12.6	5.6	42%
N50_A100_1	310.8	-26.1%	2	2	148.2	525.6	11.0	9.3	64%
N50_A0_2	423.5	0.0%	2	0	347.5	0.0	16.2	0.0	0%
N50_A25_2	423.5	0.0%	2	0	347.5	0.0	16.2	0.0	0%
N50_A50_2	408.5	-3.5%	2	1	276.9	112.5	16.2	1.8	10%
N50_A75_2	345.7	-18.4%	2	2	158.8	438.1	13.4	7.8	46%
N50_A100_2	319.7	-24.5%	2	2	161.8	403.1	11.5	8.1	58%
N50_A0_3	424.2	0.0%	2	0	341.6	0.0	16.4	0.0	0%
N50_A25_3	412.8	-2.7%	2	1	311.3	31.3	15.7	1.0	4%
N50_A50_3	400.6	-5.6%	2	1	294.0	90.2	15.2	1.5	12%
N50_A75_3	359.2	-15.3%	2	2	218.4	217.1	13.2	4.8	32%
N50_A100_3	327.8	-22.7%	2	2	166.9	339.6	12.1	6.8	52%
N50_A0_4	433.6	0.0%	2	0	312.8	0.0	17.8	0.0	0%
N50_A25_4	392.3	-9.5%	2	1	294.0	63.1	14.6	1.3	6%
N50_A50_4	371.0	-14.4%	2	2	212.7	229.8	14.3	3.9	28%
N50_A75_4	357.2	-17.6%	2	2	188.6	338.3	13.6	6.9	40%
N50_A100_4	312.9	-27.8%	2	2	156.6	340.2	11.3	6.0	54%
N50_A0_5	427.2	0.0%	2	0	339.8	0.0	16.6	0.0	0%
N50_A25_5	427.2	0.0%	2	0	339.8	0.0	16.6	0.0	0%
N50_A50_5	407.5	-4.6%	2	1	298.8	105.7	15.5	2.2	14%
N50_A75_5	339.6	-20.5%	2	1	221.2	310.6	12.2	5.5	38%
N50_A100_5	302.9	-29.1%	2	2	144.6	496.9	10.5	9.5	60%
N100_A0_1	627.2	0.0%	3	0	486.2	0.0	24.5	0.0	0%
N100_A25_1	622.3	-0.8%	3	1	468.7	122.6	23.8	2.6	7%
N100_A50_1	586.5	-6.5%	3	2	401.5	279.6	22.1	6.0	21%
N100_A75_1	537.4	-14.3%	3	2	356.8	391.4	19.5	7.0	33%
N100_A100_1	409.6	-34.7%	2	2	220.4	768.7	15.9	13.5	61%
N100_A0_2	650.6	0.0%	3	0	526.0	0.0	25.2	0.0	0%
N100_A25_2	643.1	-1.2%	3	0	518.9	0.0	24.9	0.0	0%
N100_A50_2	586.6	-9.8%	3	3	387.1	308.9	22.0	5.2	24%
N100_A75_2	446.0	-31.4%	2	2	309.1	490.6	16.9	8.2	40%
N100_A100_2	413.5	-36.4%	2	2	207.9	813.8	16.5	13.5	59%
N100_A0_3	623.9	0.0%	3	0	498.5	0.0	24.0	0.0	0%
N100_A25_3	622.9	-0.2%	3	0	472.5	0.0	24.6	0.0	0%
N100_A50_3	594.4	-4.7%	3	3	388.5	217.4	22.6	4.9	24%
N100_A75_3	436.6	-30.0%	2	2	305.1	525.5	16.2	9.3	47%
N100_A100_3	385.7	-38.2%	2	2	190.8	673.5	15.0	12.6	58%
N100_A0_4	646.7	0.0%	3	0	485.4	0.0	26.0	0.0	0%
N100_A25_4	633.2	-2.1%	3	1	460.5	118.5	24.8	2.6	8%
N100_A50_4	583.7	-9.7%	3	2	383.3	195.9	22.6	3.6	15%
N100_A75_4	558.2	-13.7%	3	3	303.3	389.9	21.7	8.4	40%
N100_A100_4	429.7	-33.6%	2	2	246.5	683.2	16.8	12.1	60%
N100_A0_5	633.3	0.0%	3	0	513.4	0.0	24.3	0.0	0%
N100_A25_5	625.3	-1.3%	3	1	508.3	91.0	23.1	2.1	5%
N100_A50_5	597.3	-5.7%	3	2	428.7	291.0	22.3	5.1	22%
N100_A75_5	522.5	-17.5%	3	3	305.8	517.3	18.9	10.3	42%
N100_A100_5	411.5	-35.0%	2	2	228.9	695.3	15.9	12.3	58%

Table 4.5: Result variation for different values of %A.

(‘T-fixed’) and drones (‘D-fixed’). Next, the variable costs are plotted: ‘Wage’ and ‘Fuel’ refer to the variable costs associated with the use of trucks or, more precisely, to the drivers salaries, and to fuel cost, respectively. Finally, ‘Elect.’ refers to the drone’s variable costs, which correspond to the electricity needed to power the drones while in flight. One can see that a major portion of the cost reduction comes from savings in fuel consumption. For instances involving 50 jobs, on average, fuel consumption is reduced by 2.4%, 21.5%, 41.7%, and 54.0% for $\%A = 25\%$, $\%A = 50\%$, $\%A = 75\%$ and $\%A = 100\%$, respectively. Correspondingly, when $n = 100$, fuel consumption is reduced by 3.2%, 20.7%, 37.0%, and 56.4%. Next, for $\%A = 25\%$, $\%A = 50\%$, $\%A = 75\%$, and $\%A = 100\%$, drone use reduces the driver en-route time by 4.8%, 8.9%, 21.5%, and 31.9% when $n = 50$, and by 2.3%, 10.1%, 24.0%, and 35.4% when $n = 100$. This is important as at the depot, the drivers can be employed for other tasks, and hence, the cost of delivering parcels can be reduced if workers are available earlier. Although not reported in Figure 4.5, the distance traveled by trucks and drones in truck-and-drone solutions turns out to be greater than the distance traveled by trucks in truck-only solutions. Indeed, drones frequently have to go back and forth from the trucks to load parcels and recharge their battery. However, in terms of costs, this augmentation of the overall traveled distance is clearly overcompensated by the increased efficiency offered by using drones as the drones’ operational costs are almost 10 times smaller than those of the trucks.

Table 4.6 summarizes the total costs incurred by using trucks (‘T-cost’) and drones (‘D-cost’). $T\text{-cost}/n$ (resp. $D\text{-cost}/n$) represents the average cost of serving one customer via truck (resp. with a drone). Finally, column ‘ J_D ’ gives the average number of jobs served by a drone. One can see that for instances with a sufficiently high percentage of jobs reachable by drone, the cost of delivering to one customer via drone is below 1€/customer, which is almost ten times smaller than the cost of serving one customer via truck (between 6.4€ and 8.5€ per customer).

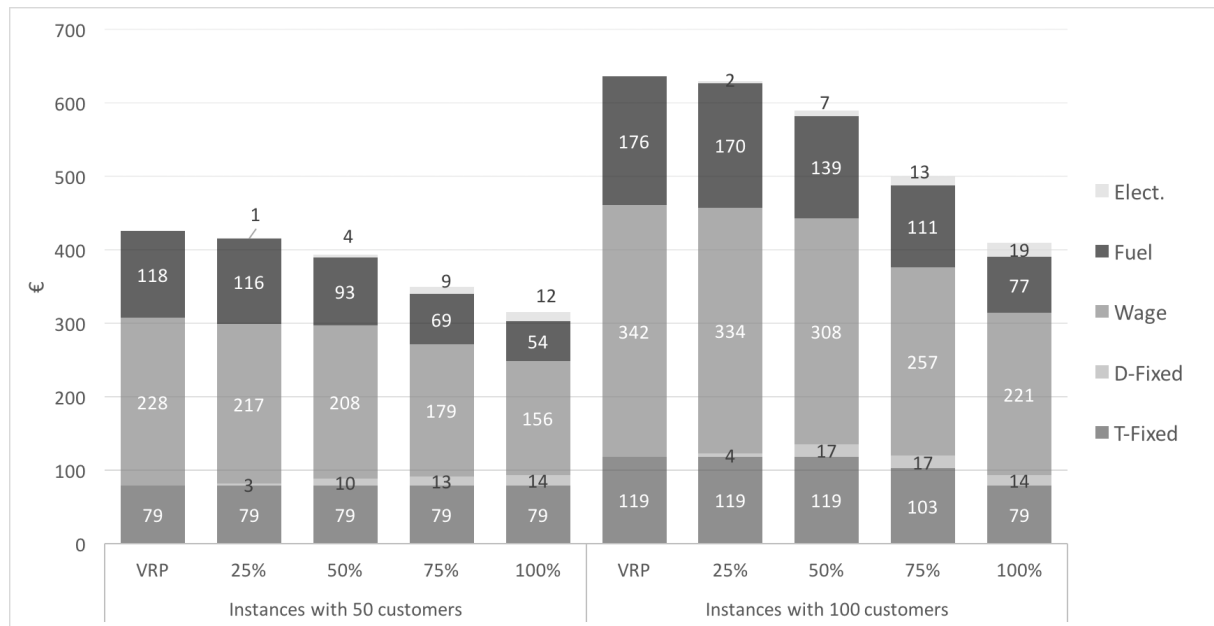


Figure 4.5: Aggregated cost structure for instances involving 50 and 100 jobs.

Instance	%A	T-cost	D-cost	Total Cost	T-cost/ n	D-cost/ n	J_D
50 Jobs	VRP	425.9	0.0	425.9	8.5	-	0
	25%	412.0	3.5	415.5	8.4	3.6	1
	50%	380.1	13.6	393.7	9.3	1.6	9
	75%	327.4	21.7	349.2	10.9	1.1	19.8
	100%	289.0	25.8	314.8	13.7	0.9	28.8
100 Jobs	VRP	636.4	0.0	636.4	6.4	-	0
	25%	623.0	6.4	629.4	6.5	1.8	4
	50%	565.5	24.2	589.7	7.2	1.2	21.2
	75%	470.4	29.7	500.1	7.9	0.7	40.4
	100%	376.8	33.2	410.0	9.2	0.6	59.2

Table 4.6: Cost structure of truck-and-drone solutions for different percentages of jobs reachable by drone.

4.7 Conclusion

Motivated by an industrial partner, this study considers the synchronization of trucks and drones to deliver parcels to customers while accounting for time-windows and drone constraints (i.e., endurance and capacity). The drones are seen as transportable resources that can be carried by trucks along their route. Combining trucks and drones allow for refilling and recharging the drones at the truck. A cost function has to be minimized, capturing both the truck's and the drone's features.

We propose a customized ALNS to solve the problem under study. This ALNS outperforms standard route-first cluster-second algorithms, which is an approach commonly used to solve truck-and-drone problems. Furthermore, we show the limitation of using a MILP approach to solve realistic instances. Compared with truck-only configurations (i.e., corresponding to the classic *Vehicle Routing Problem with Time Windows*), we highlight the significant cost reduction that can be achieved when customers can be served by drones. For instances involving 50 and 100 customers, and when all customers can be served by drones, the cost reduction lies between 20% and 35%. We show that a minimum percentage of customers reachable by drone is required to overcompensate the fixed costs associated with drone use. When the percentage of customers reachable by drone is above 50%, expenses related to fuel consumption can be reduced by 20%, and this reduction grows to 56% when all customers can be served by drone. Hence, truck-and-drone fleets possess great potential for reducing greenhouse gas emissions.

As future research directions, it would be interesting to evaluate the robustness of the proposed truck-and-drone solutions in a data-changing context. In practice, it is likely that some parameters are uncertain (e.g., delays on the road due to unexpected events [Heilporn et al., 2011, Respen et al., 2019]). Indeed, as trucks are synchronized with drones, an unexpected event on a truck route could lead to violating the drone's endurance constraint. Moreover, we plan to evaluate how truck-and-drone solutions could be improved

if the drones could fly from one truck to another or transport multiple parcels at a time, as well as if a truck could transport multiple drones.

Chapter 5

Vehicle Routing with Transportable Resources: Using Carpooling and Walking for On-Site Services

MARC-ANTOINE COINDREAU - *University of Lausanne, Switzerland*

OLIVIER GALLAY - *University of Lausanne, Switzerland*

NICOLAS ZUFFEREY - *University of Geneva, Switzerland*

*Chapter published in **European Journal of Operational Research** [Coindreau et al., 2019a]*

Abstract

In the classical *Vehicle Routing Problem* (VRP), it is assumed that each worker moves using an individually assigned vehicle. Removing this core hypothesis opens the door for new solutions, where workers are seen as transportable resources that can also move without the help of a vehicle. In this context, motivated by a major European energy provider, we consider a situation where workers can either walk or drive to reach a job and where carpooling is enabled. In order to quantify the potential benefits offered by this new framework, a dedicated *Variable Neighborhood Search* is proposed to efficiently tackle the underlying synchronization and precedence constraints that arise in this extension of the VRP. Considering a set of instances in an urban context, extensive computational experiments show that, despite conservative scenarios favoring car mobility, significant savings are achieved when compared to the solutions currently used by the involved company. This innovative formulation allows managers to reduce the size of the vehicle fleet while keeping the number of workers stable and, surprisingly, decreasing the overall driving distance simultaneously.

Keywords: Routing, On-Site Services, Synchronization, Carpooling, Variable Neighborhood Search.

5.1 Introduction

5.1.1 Industrial context

Transportation in urban areas is increasingly facing new challenges. On the one hand, the systematic use of cars produces hazardous impacts on the environment, such as noise, toxic emissions, and the effects induced by greenhouse gases [Knörr, 2008]. On the other hand, as highlighted by [Jabali et al., 2012], city centers suffer from congestion and limited parking space. These phenomena, which are magnified by low vehicle occupancy rates, decrease the intrinsic efficiency of car-based transportation. Consequently, current legislation tends to constrain the use of cars within city centers either by limiting the number of authorized vehicles or completely banning vehicles in specific areas, such as pedestrian zones, as highlighted by [Parragh and Cordeau, 2017]. For all these reasons, reducing the systematic use of cars in urban areas is becoming increasingly important. Firms that provide on-site services or parcel deliveries are directly concerned by these issues, as a substantial part of their activities takes place in metropolitan areas.

We focus on the case of a large European energy provider, denoted by EEP (it cannot be named because of a non-disclosure agreement), that routes technicians to provide on-site services (e.g., small maintenance work, consumption evaluations, and consumer-setting upgrades). Every day, technicians who are not assigned to clients are employed for heavy works on the electricity network. However, once assigned to on-site services, the workers cannot be re-assigned thereafter to heavy works, even if they terminate their working day earlier. Indeed, for the heavy works, teams of technicians are selected for the full day's work, and the jobs are frequently located outside of the cities. As a result, idle time arises in the workers' planning, either at the depot or on their route, due to the presence of time windows to serve the jobs. As each worker assigned to on-site services must be employed for the whole working day, EEP's current practice is to first minimize

the number of technicians necessary to serve all jobs. In a second phase, EEP minimizes the remaining costs implied by the technicians' routes (i.e., vehicle fixed costs and total driving distance).

EEP manages thousands of workers in urban areas, who drive more than a million kilometers every year. In that respect, EEP aims to evaluate the savings potential generated by the use of walking to reduce the total costs of its routes while also meeting the workers' expectations. EEP observed that its technicians often leave their vehicles to perform clustered jobs on foot, even if their planning would indicate driving to the next job. EEP also wants to go one step further by evaluating the savings potential of carpooling (i.e., using the same car to transport multiple workers), to scale down the size of its fleet and to possibly further reduce the overall driving costs.

Introducing these alternative transportation options obviously presents significant challenges. It is necessary to build and manage routes that are highly synchronized. Possible waiting times must be efficiently managed, as drivers might have to wait for workers to be picked up, and non-motorized workers might have to wait for drivers to be transported. Competitive solutions must also ensure that the workers' productivity remains stable, which could be decreased by the slower walking speed and the detours imposed by carpooling to drop off and pick up non-motorized technicians.

5.1.2 Problem description

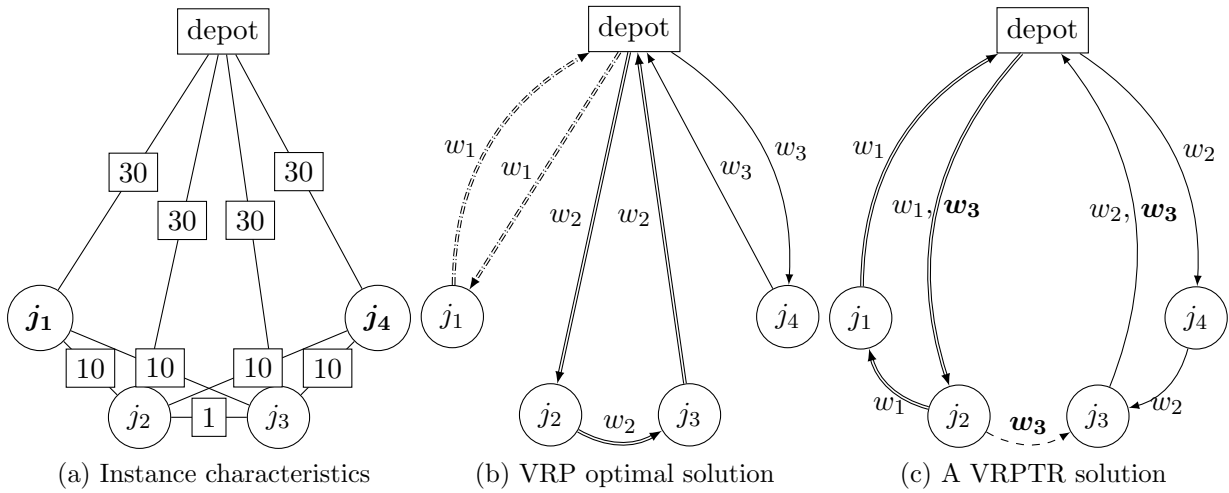
We consider the problem of routing a set of workers through different client locations in order to provide on-site jobs. Each job has a given duration and must be performed in a specific time window that is agreed upon with the involved client. This problem has garnered considerable interest in the research community in recent decades and is referred to as the *Vehicle Routing Problem* (VRP), or more specifically, as the *Vehicle*

Routing Problem with Time Windows (VRPTW). In the VRPTW, each worker moves from one job to another by driving an individually assigned car. We propose a modeling framework that relaxes this assumption, and we consider an extension of the VRPTW in which workers are allowed to share a vehicle and to choose between walking or driving to reach their next job. The technicians can be separated from their vehicles and are seen as transportable resources that can move autonomously. We refer to this extension as the *Vehicle Routing Problem with Transportable Resources* (VRPTR), for which a full description of the considered assumptions is given in Section 5.3.1. While keeping the number of workers stable compared with EEP's current practice (i.e., VRP solutions), we aim at reducing the size of the vehicle fleet and/or the total driving distance. We allow for the modeling of every situation in which workers have to visit clients without any delivery or transportation of heavy equipment, making walking a viable option. This particularly occurs with various types of home services, such as health and elder care, IT support, household appliance repairs, and security checks.

A toy example is given in Figure 5.1, which illustrates how a VRPTR solution works. The characteristics of the instance are given in the left part of the figure. Compared with the VRP solution (middle part of the figure), the VRPTR solution (right part of the figure) provides improved efficiency: the same number of workers, one car saved, and the total driving distance is reduced by 22.6%.

5.1.3 Contributions and outline

We develop both a mixed integer linear program (MILP) and a metaheuristic to solve the VRPTR. The latter uses a dedicated neighborhood structure and a fast insertion mechanism to tackle the increased complexity resulting from the introduction of walking and carpooling. Whereas the MILP is able to tackle instances up to 18 customers, the metaheuristic can solve all other instances, which involve up to 50 jobs. Compared with EEP's



(a) Values on the arcs denote the driving time (in minutes); walking is 10 times slower than driving; the planning horizon is 130 minutes; job durations are 60 minutes for j_1 and j_4 , and 30 minutes for j_2 and j_3 .

(b) and (c) The vehicle path is drawn with a specific line style; walking is represented with a dashed line; the label of an arc specifies which workers are using it.

(c) Worker w_3 is dropped off at j_2 by w_1 and then walks to j_3 , where s/he is picked up by w_2 . w_1 (resp w_2) works on j_1 (resp. j_4) after (resp. before) dropping off (resp. picking up) w_3 .

Figure 5.1: Comparison between a VRPTR solution and the corresponding VRP optimal solution.

current practice (i.e., one vehicle assigned to each worker, no walking) on a representative set of instances capturing urban characteristics, the computational experiments yield an average improvement of 6.5% for the driving distance and 18.4% for the reduction of the vehicle fleet. We show that the introduction of carpooling and walking is able to generate a simultaneous gain, both in terms of fleet size and total driving distance, in 25% of the considered instances. We highlight and quantify the trade-off that might arise between removing cars and the resulting total driving distance. Finally, we study the existing relationship between the achieved gain and the specific instance characteristics.

The remainder of the paper is organized as follows. A literature review is presented in Section 5.2. We formally describe the VRPTR and develop the associated MILP formulation in Section 5.3. In Section 5.4, we describe the proposed metaheuristic. Section 5.5 presents the computational experiments and the results. Section 5.6 proposes managerial insights (e.g., quantifying the gains compared with current practice and understanding the promising configurations for carpooling). Finally, concluding remarks and future research

opportunities are presented in Section 5.7.

5.2 Literature review

The literature review is structured as follows. We first position the VRPTR with respect to the existing VRP formulations that also synchronize different resources (a formal review of VRP with synchronization constraints can be found in [Drexl, 2012]). Next, we describe the solution methodologies that have proven to be efficient for such related problems.

Several studies consider the situation in which drivers and vehicles are allowed to disassemble along their route. [Domínguez-Martín et al., 2018] examine a case where vehicles must start and terminate their route at different depots, whereas drivers have to come back to their starting depot. As a consequence, drivers must change vehicles during their route. Similarly, in the *Vehicle and Crew Routing Problem* [Lam et al., 2015], a vehicle is driven by different drivers to maximize its use. Although these contributions explicitly consider the synchronization between vehicles and workers, they do not address aspects related to carpooling and walking.

[Levy and Bodin, 1989] originally introduced the combined use of walking and driving for mail delivery purposes. It is referred to as *Park-and-Loop* and was generalized by [Ghiani and Laporte, 2001] as the *Location-Arc Routing Problem*. The postman parks her/his car, visits a subset of jobs, comes back to the car, and drives to the next customers. In the related contributions, the modeling differs from ours, as an arc-oriented approach is considered (i.e., workers must visit arcs and not nodes). A node-oriented approach was later considered by [Gussmagg-Pfieggl et al., 2011] for a similar mail delivery application. Whereas these works acknowledge the advantages of combining walking and driving to serve on-site jobs, they do not address a potential reduction of the fleet size through the use of carpooling.

Other extensions of the VRP share a similar structure as *Park-and-Loop*, in particular when trucks and trailers can uncouple at specific locations to serve clients that cannot receive a truck paired with a trailer (thus, a lone truck stands for an on-foot worker, and a truck paired with a trailer stands for a worker equipped with a vehicle). Such problems have been introduced as the *Partially Accessible Constrained VRP* by [Rochat and Semet, 1994] and [Semet, 1995], [Semet and Taillard, 1993]. More recently, this research axis has received substantial attention under the *Truck and Trailer Routing Problem* (TTRP) [Chao, 2002, Lin et al., 2009] or the *Vehicle Routing Problem with Trailers and Transshipments* (VRPTT) [Drexler, 2014] formulations. As our contribution is not limited to the introduction of *Park-and-Loop* sub-tours in vehicle routes but also involves carpooling, these works cannot be directly applied to the present situation. Additionally, these formulations differ from ours since the motivation for *Park-and-Loop* sub-tours differs. In our case, this is due to customer restrictions in the TTRP and cost reductions in our case. Whereas the locations where the uncoupling of trailers can take place are limited to specific areas, a car can be parked at any client location in the present case.

In addition to *Park-and-Loop* aspects, the VRPTR involves the transportation of on-foot workers. Among these, [Lin, 2008] considers the synchronization of on-foot couriers with vans to deliver mail. However, and contrary to our formulation, [Lin, 2008] does not consider a complete synchronization of the resources, as on-foot couriers can only walk from the depot to a van or from a van to the depot. [Fikar and Hirsch, 2015], in a problem referred to as *Home Health Care Staff Scheduling*, addressed the situation where nurses have to visit patients in their homes. Nurses are allowed to walk but cannot drive a car. When walking is not possible, drivers, who are not permitted to visit patients, are employed to transport nurses by cars. Hence, the total number of workers (namely, nurses and drivers) is strictly greater than in the situation where nurses would drive their own cars (n.b., in the VRPTR, the technicians are both able to drive and perform jobs; hence, a reduction in the size of the vehicle fleet is achieved without increasing the number of employed workers).

In the context of parcel delivery (more generally, when the on-site presence of technicians is not mandatory), unmanned vehicles, such as drones [Wohlsen, 2014] or robots [Daimler, 2017], can be synchronized with vans to decrease the routing costs. Whereas, in the present case, unmanned vehicles would not be eligible to perform on-site services, the associated formulations share some similarities with the VRPTR. Indeed, both situations yield a similar modeling framework, where autonomous and transportable resources are dropped off and retrieved at different locations along the van routes. Although several recent contributions (e.g., [Murray and Chu, 2015, Ferrandez et al., 2016, Poikonen et al., 2017, Agatz et al., 2018, Boysen et al., 2018b]) have considered such types of synchronization, various limitations and specific constraints prevent adapting the associated solution approaches to the present case. [Murray and Chu, 2015] introduced a formulation called *Flying Sidekick Traveling Salesman Problem*, in which drones can be transported by vans to deliver parcels at client locations for some parts of their routes. In this situation and typically for contributions in this specific research domain, several major discrepancies with the present formulation can be underlined. First, only one location can be visited by the drone between its drop-off and its pick-up. Second, a single van is considered; hence, the global synchronization aspects that follow from the possibility of a drone being dropped off and picked up by different vans is not addressed. Third, the objective differs as its focus is on minimizing the completion time (i.e., time windows are not considered). Finally, [Boysen et al., 2018b] assume that robots can wait indefinitely at the depot or at client locations. Such an assumption precludes its application in the present context, since the number of workers is limited and their employment is costly.

In the *Active Passive Vehicle Routing Problem* (APVRP) (see [Meisel and Kopfer, 2014] or [Tilk et al., 2017]) as well as in the *Rollon-Rolloff Vehicle Routing Problem* (see e.g., [Bodin et al., 2000]), a set of trailers has to be transported with trucks from loading to unloading locations (i.e., pick-up and delivery requests). The duration of these operations is long enough to allow the trucks (i.e., the active transportation resources) to move other trailers (i.e., the passive transportation resources) in the meantime. Moreover, trailers

can be carried by different vehicles (see [Smilowitz, 2006] for an example of such a practice in drayage operations in the Chicago region). Creating bridges with the present study, a trailer can be seen as a non-motorized worker that requires transportation between different locations. However, the complexity is increased in our problem because the passenger transportation requests are not fixed a priori and are part of the decision-making process.

Most of the above-cited papers propose an exact formulation for the problem under study, which is able to solve instances of limited size (e.g., the MILP developed in [Murray and Chu, 2015] is able to tackle instances involving up to 10 customers in a 10-square-mile region). The exact approaches are often complemented with a two-stage heuristic to find solutions for larger instances, either in a *cluster-first-route-second* or in a *route-first-cluster-second* fashion. The first alternative is aimed at initially building job clusters that will be visited by the transportable resources alone and then creating routes for the carrying vehicles to connect the clusters together [Levy and Bodin, 1989, Fikar and Hirsch, 2015]. The second alternative proposes to first build routes for the carrying vehicles and then to assign some clients to the transportable resources (see e.g., [Ghiani and Laporte, 2001], [Gussmagg-Pfieggl et al., 2011], [Murray and Chu, 2015]). Even though these two approaches are able to efficiently improve the quality of the initially generated solutions, they suffer from being easily trapped in a local minimum since the decision at the first stage strongly impacts the quality of the decisions in the second one.

General metaheuristics based on the *ruin and recreate* principle have proven to be successful for various related VRP formulations [Schrimpf et al., 2000]. These solution approaches do not suffer from the drawbacks of a two-stage methodology, as the decisions on the job clusters and the routing are made simultaneously. In a routing context, the *ruin and recreate* principle aims to improve a solution by iteratively removing and reinserting some jobs (one of the numerous fruitful implementations is presented

in [Pisinger and Ropke, 2007]). Known as *Large Neighborhood Search* (LNS) and introduced by [Shaw, 1997], this principle has been the basis of multiple successful contributions in various domains. In particular, two related metaheuristics are developed in [Derigs et al., 2013] for the TTRP. They both combine the strength of a descent algorithm for the intensification with the exploration ability of LNS for the diversification. The authors highlight the benefit of combining a local search and a collection of neighborhood structures of different amplitudes, as in LNS.

5.3 Problem formulation

As the VRPTW is a special case of the VRPTR (where walking is forbidden and the number of vehicles is equal to the number of workers), the VRPTR can be classified as an \mathcal{NP} -Hard problem (see [Cordeau et al., 2007] for overviews of the various VRP characteristics, their associated models, and their efficient solution approaches).

5.3.1 Definition and assumptions

A walking path between a set of jobs is called a walking route (WR). Idle time is the total time that a worker waits in a solution (either en route or at the depot). Returning to the depot earlier at the end of the day is considered idle time, as workers are employed for the whole day, and they cannot be assigned to other tasks once they are back at the depot. For the EEP context, the following features are taken into account:

- The planning horizon is a day (i.e., the daily working time is upper bounded), for which all the jobs and travel information are accurately known (static data).
- For each worker, the walking limitations are the maximum daily walking distance

(d_M^f) and the maximum allowed walking time between two jobs (τ_M^f) .

- Vehicles and workers can disassemble and reassemble at any job location (the duration of this operation is assumed to be null).
- Each vehicle has a single assigned worker, meaning that the workers are separated into two categories: the drivers and the passengers (drivers have to perform their assigned jobs and to fulfill the transportation requests of passengers).
- Workers and vehicles start and end their routes at the depot.
- Both driver and passenger workers can walk to reach the next job on their routes. In the driver case, the return path to her/his car is mandatory (i.e., departure and arrival points of a WR must coincide), whereas in the passenger case, departure and arrival points of a WR can be different.
- Idling is allowed for both the drivers and the passengers at job locations.

5.3.2 Graph modeling and variables

Let $J = \{1, \dots, n\}$ be the set of jobs, K the set of motorized workers (i.e., drivers), and L the set of non-motorized workers (i.e., passengers). $W = K \cup L$ denotes the set of all workers. For job $j \in J$, $p_j \in \mathbb{R}^+$ is its processing time, and $(e_j, l_j) \in \mathbb{R}^{+2}$ is its time window, consisting of the earliest and latest possible service times. Between two jobs $(i, j) \in J^2$, the distance (in km) is given by $d_{ij} \in \mathbb{R}^+$, and the driving (resp. walking) time (in minutes) is denoted by $\tau_{ij} \in \mathbb{R}^+$ (resp. $\tilde{\tau}_{ij} \in \mathbb{R}^+$). $c \in \mathbb{N}$ indicates the maximum number of non-motorized workers allowed in a car (in addition to the motorized worker). Finally, $M_1 = \max_{j \in J} l_j + \max_{i \in J, j \in J} \tau_{ij}$ and $M_2 = \max_{j \in J} l_j + \max_{j \in J} p_j + \max_{i \in J, j \in J} \tilde{\tau}_{ij}$ are sufficiently large numbers, which are required for the MILP.

The node set J is duplicated using $J^+ = \{n+1, \dots, 2n\}$, where $i \in J$ and $i+n \in J^+$ represent the same physical location, with $i \in \{1, \dots, n\}$. These two sets allow

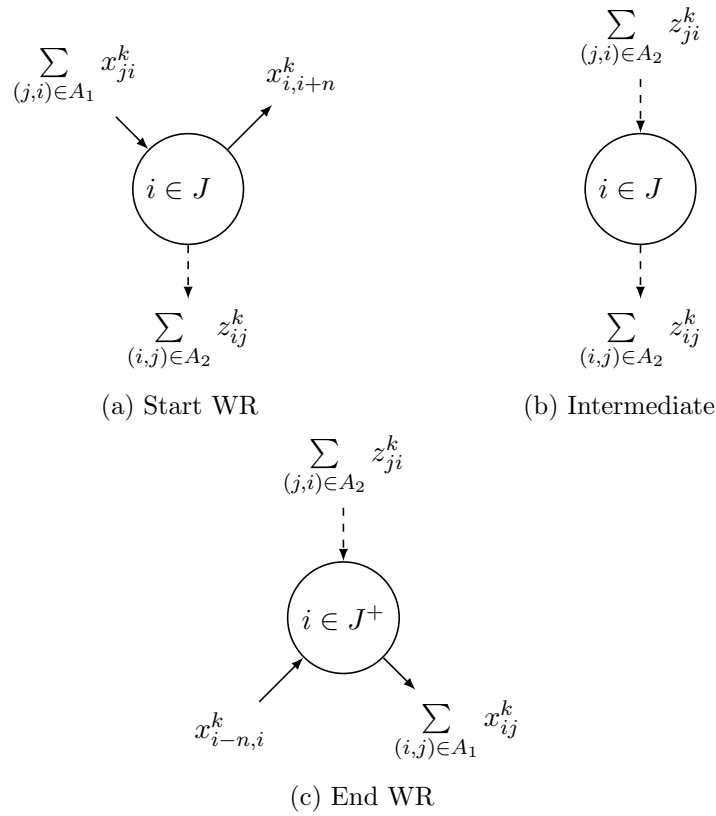
distinguishing the different operations taking place at the same physical node (e.g., a motorized worker parks her/his car, starts a WR, and retrieves her/his car). J (resp. J^+) stands for the set of starting and intermediate points (resp. terminating points) of a WR. As a result, in the optimization model, a motorized worker starts a WR at $j \in J$ and finishes it at $j + n \in J^+$ (three different times are thus managed: arrival time, service time, and departure time). $V = J \cup J^+ \cup \{0, 2n + 1\}$ is the set of all nodes, where 0 represents the starting depot and $2n + 1$ the ending depot. Based on this notation, $A_1 = \{(i, j) \in V \setminus \{2n + 1\} \times V \setminus \{0\}, \text{ such that } i \neq j \text{ and } j \neq i - n \text{ for } i \in J^+\}$ is the driving arc set and $A_2 = \{(i, j) \in J \times (J \cup J^+), \text{ such that } i \neq j \text{ and } \tilde{\tau}_{ij} < \tilde{\tau}_M^f\}$ is the walking arc set.

We define the variables:

- $x_{ij}^k \in \{0, 1\}$ is equal to 1 if motorized worker $k \in K$ uses arc $(i, j) \in A_1$; $x_{ij}^k = 0$, otherwise,
- $y_{ij}^{kl} \in \{0, 1\}$ is equal to 1 if non-motorized worker $l \in L$ is transported by the vehicle associated to motorized worker $k \in K$ on arc $(i, j) \in A_1$; $y_{ij}^{kl} = 0$, otherwise,
- $z_{ij}^w \in \{0, 1\}$ is equal to 1 if a worker $w \in W$ walks on arc $(i, j) \in A_2$; $z_{ij}^w = 0$, otherwise,
- $t_i^w \in \mathbb{R}$ denotes the time at which worker $w \in W$ leaves node $i \in V$,
- $s_i \in \mathbb{R}$ stands for the time at which the service starts at node $i \in J$.

Figure 5.2 illustrates the flow of a motorized worker when walking is involved. It is helpful to understand the coordination between a motorized worker and her/his vehicle as detailed in the constraints below.

Due to carpooling, and contrary to the standard VRP formulations, a worker can visit a node without performing the associated job. In standard VRP formulations, when a worker visits a node, s/he performs the associated job. When carpooling is allowed,



Dashed (resp. plain) lines denote an edge traveled on foot (resp. with a car).

(a) Shows the arcs activated if motorized worker $k \in K$ starts a WR in $i \in J$.

(b) Shows the arcs activated if $i \in J$ is an intermediary point of a WR performed by a motorized worker $k \in K$.

(c) Shows the arcs activated if motorized worker $k \in K$ terminates a WR in $i \in J^+$.

Figure 5.2: Different flow configurations for a WR performed by a motorized worker.

several workers can stop at a node but only the dropped worker performs the associated job. Indeed, when a non-motorized worker is dropped off at a node $i \in J$, the motorized worker (and potentially other non-motorized workers on-board) also stops at the node i , but s/he continues her/his route and does not perform the associated job. Consequently, with the introduced notation, a worker $w \in W$ completes the job at $i \in J$ if and only if w exits the node on foot (i.e., $\sum_{(i,j) \in A_2} z_{ij}^w = 1$). Figure 5.3 illustrates the different node sets and variables for the VRPTR solution displayed in Figure 5.1. In this example, $J = \{1, 2, 3, 4\}$ and $J^+ = \{5, 6, 7, 8\}$ (e.g., nodes 1 and 5 represent the same physical node j_1), and node 0 (resp. node 9) represents the starting (resp. ending) depot.

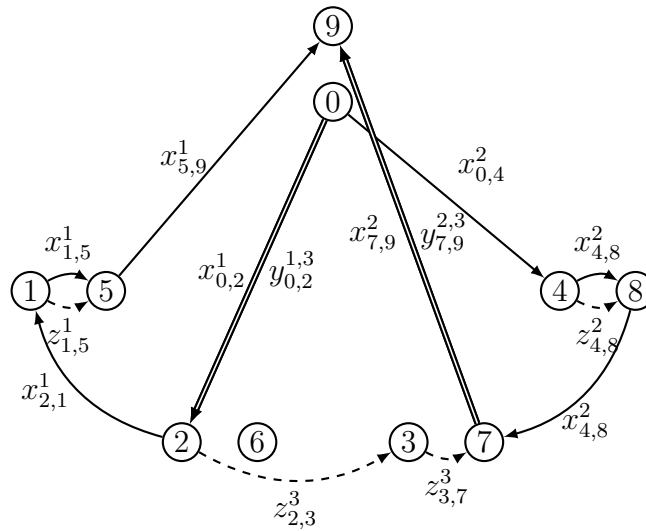


Figure 5.3: Modeling of the VRPTR solution displayed in Figure 5.1 using the introduced sets and variables

5.3.3 Mathematical formulation

We propose a MILP model for the VRPTR, where the numbers of both the involved workers and vehicles are given as inputs. More precisely, we fix the number of workers to the optimal value found when all workers are motorized. The costs associated with worker daily wages hence remain the same in the VRPTR solutions and in the corresponding optimal VRP solutions. To reduce the number of vehicles, the MILP is then launched sequentially, every time with one vehicle less, until no feasible solution can be found (i.e., a sequence of instances is thus generated in this way). The trade-off that appears between the size of the employed vehicle fleet and the total driving distance is discussed in Section 5.6.1.

For a fixed number of workers and a given vehicle fleet, Objective (5.1) minimizes the total driving distance (which constitutes the remaining transportation costs):

$$\text{minimize } \sum_{(i,j) \in A_1} \sum_{k \in K} d_{ij} \cdot x_{ij}^k \quad (5.1)$$

Constraints for workers and vehicles flows.

$$\sum_{(i,j) \in A_2} \sum_{w \in W} z_{ij}^w = 1, \quad i \in J \quad (5.2)$$

$$\sum_{(i,j) \in A_2} d_{ij} \cdot z_{ij}^w \leq d_M^f, \quad w \in W \quad (5.3)$$

$$\sum_{(j,i) \in A_1} x_{ji}^k = \sum_{(i,j) \in A_1} x_{ij}^k, \quad i \in J \cup J^+, k \in K \quad (5.4)$$

$$\sum_{(i,j) \in A_2} z_{ij}^k \leq \sum_{(j,i) \in A_2} z_{ji}^k + \frac{1}{2} \left(\sum_{(j,i) \in A_1} x_{ji}^k + x_{i,i+n}^k \right), \quad i \in J, k \in K \quad (5.5)$$

$$\sum_{(j,i) \in A_2} z_{ji}^k + \sum_{(j,i) \in A_1} x_{ji}^k \leq 1, \quad i \in J, k \in K \quad (5.6)$$

$$\sum_{(j,i) \in A_2} z_{ji}^k = x_{i-n,i}^k, \quad i \in J^+, k \in K \quad (5.7)$$

$$\sum_{(i,j) \in A_1} x_{ij}^k \leq 1, \quad i \in V, k \in K \quad (5.8)$$

$$\sum_{(i,j) \in A_2} z_{ij}^w \leq 1, \quad i \in J, w \in W \quad (5.9)$$

$$\sum_{i \in V} x_{0i}^k = \sum_{i \in V} x_{i,2n+1}^k, \quad k \in K \quad (5.10)$$

Constraints (5.2) ensure that all the jobs are processed. Note that some nodes in J^+ might not be visited, as all jobs do not terminate a WR (e.g., see node 6 in Figure 5.3). Constraints (5.3) define an upper bound on the total daily walking distance of a worker (valid for both motorized and non-motorized workers). Constraints (5.4) ensure that vehicles arriving at a node ultimately exit the node. Constraints (5.5) impose that a motorized worker leaving $i \in J$ by walking must formerly arrive at this node either by walking or by car (see Figure 5.2 (a, b)). When a motorized worker performs a job $i \in J$, s/he exits this node by walking (according to the assumptions detailed above). Ultimately, s/he moves from i to $i + n$ (the end of the walking route) without his/her car, therefore, we allow in our model that his/her assigned car moves from i to $i + n$ without the driver on-board, by doing so, the car flow coincides with the worker flow.

Constraints (5.6) forbid a motorized worker from arriving both by car and by walking at $i \in J$. Constraints (5.7) state that a motorized worker walking to $i \in J^+$ has her/his car waiting for her/him at i (see Figure 5.2 (c)). Constraints (5.8) and (5.9) forbid a worker (motorized or non-motorized) from using two different arcs simultaneously. Constraints (5.10) state that every motorized worker who leaves the depot has to come back to it in a single trip.

Specific constraints for non-motorized workers:

$$\sum_{k \in K} \sum_{(j,i) \in A_1} y_{ji}^{kl} + \sum_{(j,i) \in A_2} z_{ji}^l = \sum_{k \in K} \sum_{(i,j) \in A_1} y_{ij}^{kl} + \sum_{(i,j) \in A_2} z_{ij}^l, \quad i \in J, l \in L \quad (5.11)$$

$$\sum_{k \in K} \sum_{(i,j) \in A_1} y_{ij}^{kl} + \sum_{(i,j) \in A_2} z_{ij}^l \leq 1, \quad i \in J, l \in L \quad (5.12)$$

$$\sum_{k \in K} \sum_{(j,i) \in A_1} y_{ji}^{kl} + \sum_{(j,i) \in A_2} z_{ji}^l = \sum_{k \in J} \sum_{(i,j) \in A_1} y_{ij}^{kl}, \quad i \in J^+, l \in L \quad (5.13)$$

$$\sum_{k \in K} \sum_{i \in V} y_{0i}^{kl} = \sum_{k \in K} \sum_{i \in V} y_{i,2n+1}^{kl}, \quad l \in L \quad (5.14)$$

$$\sum_{(j,i) \in A_2} z_{ji}^w \leq \sum_{(i,j) \in A_2} z_{ij}^w, \quad i \in J, w \in W \quad (5.15)$$

$$\sum_{l \in L} y_{ij}^{kl} \leq c \cdot x_{ij}^k, \quad (i,j) \in A_1, k \in K \quad (5.16)$$

$$\sum_{(j,i) \in A_1} y_{ji}^{kl} \leq 1, \quad i \in V, l \in L, k \in K \quad (5.17)$$

Constraints (5.11) ensure that a non-motorized worker arriving at node $i \in J$ (either by walking or by car) ultimately exits the node. Constraints (5.12) state that a non-motorized worker cannot use the two different transportation modes (i.e., walking and driving) to leave a node. Constraints (5.13) force any non-motorized worker arriving at a node $i \in J^+$ to exit the node by car. Constraints (5.14) state that every non-motorized worker who leaves the depot has to come back to it in a single trip. Constraints (5.15) ensure that a worker arriving by walking at $i \in J$ exits by walking. Constraints (5.16) couple non-motorized worker transportation and motorized worker routes, it also limits the total number of workers that can be transported in the same car. Such constraints are also considered in the APVRP (see [Meisel and Kopfer, 2014]). Constraints (5.17) forbid

a worker from using two different arcs arriving at the same node.

Time constraints:

$$t_j^l \geq t_i^k + \tau_{ij} - M_1 \cdot (1 - y_{ij}^{kl}), \quad (i, j) \in A_1, k \in K, l \in L \quad (5.18)$$

$$t_j^k \geq t_i^k + \tau_{ij} - M_1 \cdot (1 - x_{ij}^k), \quad (i, j) \in A_1, k \in K \quad (5.19)$$

$$t_j^w \geq s_i + p_i + \tilde{\tau}_{ij} - M_2 \cdot (1 - z_{ij}^w), \quad (i, j) \in A_2, w \in W \quad (5.20)$$

$$s_i \geq t_i^w - M_3 \cdot \left(1 - \sum_{(i,j) \in A_2} z_{ij}^w\right), \quad i \in J, w \in W \quad (5.21)$$

$$t_i^k \geq t_i^l - M_3 \cdot \left(1 - \sum_{(i,j) \in A_1} y_{ij}^{kl}\right), \quad i \in J^+, k \in K, l \in L \quad (5.22)$$

$$l_i \geq s_i \geq e_i, \quad i \in J \quad (5.23)$$

$$l_0 \geq t_0^w \geq e_0, \quad w \in W \quad (5.24)$$

For each non-motorized worker $l \in L$, constraints (5.18) set the arrival time at $j \in V$ after being transported by motorized worker $k \in K$ using arc $(i, j) \in A_1$. Constraints (5.19) define the arrival time at $j \in V$ of motorized worker $k \in K$ using arc $(i, j) \in A_1$. Constraints (5.20) set the arrival time at $j \in J \cup J^+$ for worker $w \in W$ after processing job $i \in J$ and walking thereafter. Constraints (5.21) impose that the service time of job $i \in J$ takes place after the arrival time of the worker at that node. Constraints (5.22) impose that a motorized worker can only leave node $i \in J^+$ if all the non-motorized workers to be transported by her/him have arrived at the node. Constraints (5.23) impose that the service time of each job must belong to the associated time window. Constraints (5.24) impose that all workers leave and come back to the depot within the regulatory hours.

5.4 Methodology

This section starts by describing the general principles of the proposed *Variable Neighborhood Search* (VNS) and the reasons why it is expected to provide good results for the VRPTR. Next, we present a dedicated insertion heuristic that helps to specifically manage walking and carpooling. Finally, after highlighting the complexity associated with searching for the best insertion position for a job in a VRPTR solution, we introduce an algorithm to speed up this procedure, which is then employed as a key procedure of our VNS.

5.4.1 VNS: motivation and general principles

To tackle the considered problem, we propose a VNS [Mladenović and Hansen, 1997] that combines a large neighborhood structure \mathcal{N}_q (it first removes $q > 1$ jobs from the solution and then reinserts them sequentially) and a local search (LS). On the one hand, the role of the LNS component is to diversify the search (i.e., explore new parts of the solution space). For this purpose, it is mandatory to consider large neighborhoods to tackle the VRPTR, as the presence of WRs is responsible for trapping the search in local minima. More precisely, a WR synchronizes multiple workers (i.e., the one that is dropped, the driver that brings her/him at the beginning of the WR, and the driver that picks her/him up at the end of it). Unless all the jobs composing such a WR are removed from the solution, the removed jobs tend to be reinserted at the same position in the same WR. The exploration capability of \mathcal{N}_q would thus be poor for small values of q . On the other hand, and in contrast with the LNS component of VNS, the role of LS is to intensify the search in promising regions of the solution space. For this purpose, small (in terms of the modification of the solution structure) but efficient (i.e., it should be able to favorably modify the solution value) moves should be performed iteratively on the incumbent solution. Unsurprisingly, [Derigs et al., 2013] (for the TTRP) and [Meisel and Kopfer, 2014] (for the APVRP) have

shown that combining a LNS and a LS outperforms the use of a LNS only. We have confirmed this observation by performing preliminary experiments on our instances.

A generic version of the VNS is given in Algorithm 10. It starts from an initial solution s and considers a collection of neighborhood structures $\mathcal{N} = \{\mathcal{N}_1, \dots, \mathcal{N}_{q_{max}}\}$ that are ranked according to their strength for modifying a solution (i.e., \mathcal{N}_q modifies the structure of the involved solution more than \mathcal{N}_{q-1}). Section 5.4.2 details the dedicated algorithm that is used to explore the neighborhood \mathcal{N}_q .

The implemented LS is a descent algorithm based on *relocate* moves. Formally, at each step, a job is removed from the solution and reinserted at the best possible location, with or without involving additional walking. As long as the generated neighbor solution $s^{neighbor}$ outperforms the current solution s , the new current solution immediately becomes $s^{neighbor}$. The process stops when the current solution cannot be improved further (i.e., all jobs of s have been tested). More refined LS algorithms (e.g., a tabu search for which it is forbidden to insert a job in some positions) were tested, but they did not yield better results.

In order to facilitate the exploration of the solution space associated with the VRPTR, constraints (5.2), which guarantee that all jobs are visited, are relaxed (i.e., removed from the constraints and penalized in the objective function with a penalty parameter ψ). Following this, for each solution $s = \{x, y, z\}$, $d(s) = \sum_{k \in K} \sum_{(i,j) \in A_1} d_{ij} \cdot x_{ij}^k$ is the overall driving distance. $J^{in}(s) = \{j \in J \mid \sum_{(j,i) \in A_2} z_{ji} = 1\}$ is the set of jobs that are served in s , and $J^{out}(s) = J \setminus J^{in}(s)$ is the set of unserved jobs in s . As done in other VRP contributions (see e.g., [Pisinger and Ropke, 2007]), Objective (5.1) thus becomes as shown below, where ψ is chosen sufficiently large to ensure that for the two solutions s and s' , if $|J^{out}(s)| < |J^{out}(s')|$, then $c(s) < c(s')$, with:

$$c(s) = d(s) + \psi \cdot |J^{out}(s)| \quad (5.25)$$

Algorithm 10 Variable Neighborhood Search (VNS)

Input: n_L, q_{max}, K, W .

Generate an initial solution s with $|K|$ vehicles and $|W|$ workers.

Set $q = 1$.

While no stopping condition is met, **do**

- (1) *Shaking*: randomly generate n_L (parameter) solutions in $\mathcal{N}_q(s)$, and let s' be the best of these solutions.
 - (2) *Local search (LS)*: apply the local search on s' , and let s'' be the resulting solution.
 - (3) *Move or not*: if s'' is better than s , move there (i.e., set $s = s''$), and continue the search with \mathcal{N}_1 (i.e., set $q = 1$); otherwise set $q = q + 1$, but if $q > q_{max}$, set $q = 1$ (i.e., start a new research cycle).
-

5.4.2 VNS: shaking phase

Neighborhood \mathcal{N}_q is explored by means of an LNS-type procedure, based on the sequential use of a *removal* heuristic and an *insertion* heuristic. These heuristics are detailed hereafter.

Removal heuristic

The removal heuristic aims at dropping jobs that currently block the search process in a local minimum. We consider here the *related removal heuristic* (RRH) proposed by [Shaw, 1998] and adapt it for the VRPTR. The general idea is that it is likely to be easier to reinsert removed jobs that share some similarities. The relatedness function $R(i, j)$ indicates how two jobs i and j are similar. To that aim, some parameters are introduced. $(\alpha, \beta, \gamma, \delta, \epsilon)$ are positive weights, and $\text{WR}(i, j) = 1$ if i and j are served in the same WR; $\text{WR}(i, j) = 0$ otherwise. $\mathbb{1}_{k_i=k_j} = 1$ if i and j are served in the same route; $\mathbb{1}_{k_i=k_j} = 0$ otherwise. This relatedness function takes into account the geographical proximity ($\alpha \cdot d_{ij}$), the similarity in service time ($\beta \cdot |h_i - h_j|$), the similarity in time windows ($\gamma \cdot |l_i - l_j|$), and the presence in common WRs ($\delta \cdot (1 - \text{WR}(i, j))$) and common routes ($\epsilon \cdot (1 - \mathbb{1}_{k_i=k_j})$).

The smaller $R(i, j)$ is, the greater i and j are related:

$$R(i, j) = \alpha \cdot d_{ij} + \beta \cdot |h_i - h_j| + \gamma \cdot |l_i - l_j| + \delta \cdot (1 - \text{WR}(i, j)) + \epsilon \cdot (1 - \mathbb{1}_{k_i=k_j}) \quad (5.26)$$

The first removed job is randomly selected in $J^{in}(s)$. Then, L_{NR} designates the ranked list of non-removed jobs. The relatedness of a non-removed job i is computed according to one of the removed jobs j (that is randomly selected, as proposed in other studies considering related removal, e.g., [Ropke and Pisinger, 2006]). Next, as long as q removals have not been performed, the job $L_{NR}[\lfloor y^\rho \cdot |J^{in}(s)| \rfloor]$ is removed from the solution. y is randomly generated in $[0, 1]$, and $\rho \in [0, 1]$ is a parameter that calibrates the degree of randomness of the removal heuristic ($\rho = 1$, jobs are randomly removed; $\rho = 0$, the most related job is removed at each step).

Insertion heuristic

Before introducing the proposed insertion heuristic, we start by describing the drawbacks that best-insertion heuristics (BIHs), which is one of the most frequently used insertion components of the LNS (e.g., [Ropke and Pisinger, 2006, Masson et al., 2014, Grangier et al., 2016]), have in the present situation. The BIH inserts first the job that minimizes the insertion cost (i.e., the additional driving distance in the present case). The BIH is inefficient in the VRPTR context because it either favors (a) insertions involving walking only or (b) insertions in a driver's planning. For (a), it follows from the fact that walking does not increase the driving distance. For (b), moving a driver to a job only requires one detour with her/his assigned car, whereas assigning this job to a passenger requires two detours: one for the drop-off and one for the pick-up. First, these drawbacks limit the diversification ability of the insertion heuristic. Second, unbalanced schedules are created for the workers because more jobs are assigned to drivers than to passengers, which finally results in assigning the latest considered jobs to the passengers (which are

the most difficult resources to move). For all these reasons, we propose below an insertion heuristic capable of removing these two drawbacks due to carpooling and walking.

We propose a *Random Worker Best-Insertion* (RWBI) heuristic, the pseudo-code of which is given in Algorithm 11. At each step of the RWBI, a worker is first randomly chosen, and then a non-dominated insertion is performed. An insertion is said to be non-dominated if no other insertion has a better performance, according to both the walking distance and the insertion cost. Randomly selecting the worker that will serve the next job helps in overcoming the problem of over-insertions in drivers' planning. Furthermore, choosing a non-dominated insertion position allows for efficiently managing the amount of walking time in the solution. On the one hand, walking seems favorable, as it does not contribute cost-wise to the objective function. Additionally, the more the passengers are walking, the less the drivers are used for transporting the passengers, and the more time they can allocate to perform jobs themselves. However, on the other hand, walking directly reduces the workers' availability and thus augments the likelihood of having unserved jobs that will remain at the end of the shaking phase. During the RWBI, the jobs are inserted without walking in the drivers' routes. This maximizes the likelihood of getting a feasible and non-saturated solution at the end of the diversification step. Walking is added to the drivers' planning during the intensification step (i.e., LS).

Algorithm 11 Random Worker Best Insertion heuristic (RWBI)

Input: s

Set $J^{out}(s)$ as the set of unserved jobs in s .

While $J^{out}(s) \neq \emptyset$, **do**

- (1) Compute the set W^{av} of available workers.
- (2) Select a worker $w \in W^{av}$ randomly.
- (3) Randomly choose a non-dominated insertion from J^{out} in w .

If no feasible insertion is found, *change* w **or** stop RWBI if all workers have been tested.

Else: proceed the insertion **and** update $J^{out}(s)$.

5.4.3 Complexity of an insertion

The best-insertion move is the core component of both RWBI and LS. This section shows that finding the best insertion position for a job in a solution involving carpooling requires $\mathcal{O}(n^5)$ feasibility tests. For the VRP, it only requires $\mathcal{O}(n)$ of such tests.

To find the cheapest insertion position (i.e., the one that least increases the driving distance), all the insertion positions are greedily tested. We consider the insertion of job $j \in J^{out}(s)$ to a non-motorized worker route (n jobs are inserted in the solution). In this case, we need to transport this worker from its previous WR to j , and from j to its next WR (after the job has been processed). This leads to the creation of two new transportation requests (each one composed of a pick-up and a delivery) that have to be inserted in the driver routes. As a result, four nodes must be inserted into the solution, and accordingly, the number of feasibility tests is in $\mathcal{O}(n^4)$. The number of insertion positions (between two WRs) for this non-motorized worker is proportional to the number of jobs in the solution, and therefore the total number of required tests grows to $\mathcal{O}(n^5)$. Note that removing a job from the route of a non-motorized worker is also a complex task, as a transportation request from her/his previous WR to her/his next WR has to be created.

5.4.4 Accelerating up the insertion phase

In addition to the significantly larger number of tests to be performed, proving the feasibility of an insertion is a more complex task for the VRPTR than for the VRP. First, we need to check that the induced delay after the insertion does not violate future time windows. As all routes can be interconnected, it is not sufficient to only recompute the concerned route; rather the whole solution may have to be updated. Second, when assigning a job to a non-motorized worker, four nodes must be inserted into the solution

(see Section 5.4.3). Therefore, the solution must be correctly updated four times (once after each node insertion) when checking the overall feasibility of the insertion.

To avoid having to recompute the whole solution after each of the four insertions, we propose a fast insertion algorithm based on a precedence graph that captures all the temporal constraints. [Masson et al., 2014] introduced the same type of graph structure for a vehicle routing problem in which transfers are allowed to transport persons. Each physical node in the real network has its associated node in the precedence graph. For each node v in this graph, it is possible to compute the earliest arrival time h_v and the latest departure time λ_v (i.e., leaving v after this time would lead to a violation of a future time window) and the waiting time matrix Φ between all pair of nodes. For a driver, waiting at a node is either due to an early arrival before the start of a time window or to the later arrival of a non-motorized worker that needs to be transported further. The computation of these values is done in $\mathcal{O}(n^2)$. A.1 shows the construction of the precedence graph and provides details on the computation of h_v , λ_v and Φ [Cherkassky et al., 2009].

$i_1 + 1$ denotes the successor node of i_1 in route k_1 . e_{D_1} (resp. l_{D_1}) is the earliest start time of the WR starting at node D_1 (resp. latest arrival time at the WR starting at D_1 to serve all jobs of the WR on time). p_{D_1} is the processing time of the WR starting at D_1 . After proceeding with an insertion in the graph, for node v in the precedence graph, \bar{h}_v is the new arrival time and $\delta_v = \max\{\bar{h}_v - h_v, 0\}$ denotes the delay induced in v .

Based on these notations, Algorithm 12 tests in constant time whether assigning a job to a non-motorized worker at a given position is feasible. More precisely, the proposed algorithm contains a specific feasibility check that corresponds to the precedence constraints arising between the WRs performed by a same worker. Algorithm 12 details the most complex situation when four nodes (namely (P_1, D_1) and (P_2, D_2) , corresponding to the two new transportation requests created for the non-motorized worker) are inserted into positions (i_1, j_1) (resp. (i_2, j_2)) in route k_1 (resp. k_2). After inserting any of the four

nodes, the induced delay at any other node is computed in constant time. For this purpose, the delay after each insertion is reduced by the smallest waiting time between the predecessor and successor nodes. If the delay does not exceed the latest departure time of the successor node, all other jobs will still be served within their time window, and the four nodes remaining to be inserted are tested. If this is not so, the insertion position is determined to be unfeasible. Experiments have shown that using this fast feasibility check procedure can reduce the computation time of the proposed VNS by 95%.

Algorithm 12 Algorithm for testing the feasibility of the insertion of (P_1, D_1) and (P_2, D_2) after i_1, j_1, i_2, j_2 , respectively.

Evaluate the insertion of P_1 :

- **set** $\bar{h}_{P_1} = \max\{e_{P_1}, h_{i_1} + \tau_{i_1, P_1}\}$, **if** $\bar{h}_{P_1} > l_{P_1}$ **return FALSE**.
- **set** $\bar{h}_{i_1+1} = h_{P_1} + \tau_{P_1, i_1+1}^d$, **if** $\bar{h}_{i_1+1} > \lambda_{i_1+1}$ **return FALSE**.

Evaluate the insertion of D_1 :

- **set** $\bar{h}_{j_1} = h_{j_1} + \max\{\delta_{i_1+1} - ST_{i_1+1, j_1}, 0\}$, **set** $\bar{h}_{D_1} = \max\{h_{j_1} + \tau_{j_1, D_1}, e_{D_1}\}$, , **if** $\bar{h}_{D_1} > l_{D_1}$ **return FALSE**.
- **set** $\bar{h}_{j_1+1} = \bar{h}_{D_1} + \tau_{D_1, j_1+1}$, **if** $\bar{h}_{j_1+1} > \lambda_{j_1+1}$ **return FALSE**.

Evaluate the insertion of P_2 :

- **set** $\bar{h}_{i_2} = h_{i_2} + \max\{\delta_{i_1+1} - ST_{i_1+1, i_2}, \delta_{j_1+1} - ST_{j_1+1, i_2}, 0\}$ and **set** $\bar{h}_{P_2} = \max\{h_{i_2} + \tau_{i_2, P_2}, h_{D_1} + p_{D_1}\}$.
- **set** $\bar{h}_{\sigma(i_2)} = h_{P_2} + \tau_{P_2, i_2+1}$, **if** $\bar{h}_{\sigma(i_2)} > \lambda_{\sigma(i_2)}$ **return FALSE**.

Evaluate the insertion of D_2 :

- **set** $\bar{h}_{j_2} = h_{j_2} + \max\{\delta_{i_1+1} - ST_{i_1+1, j_2}, \delta_{j_1+1} - ST_{j_1+1, j_2}, \delta_{i_2+1} - ST_{i_2+1, j_2}, 0\}$, **set** $\bar{h}_{D_2} = \bar{h}_{j_2} + \tau_{j_2, D_2}^d$, **if** $\bar{h}_{D_2} > l_{D_2}$ **return FALSE**.
- **set** P_3 as the pick-up at the end of WR D_2 , **if** $\max\{\bar{h}_{D_2}, e_{D_2}\} + p_{D_2} > \lambda_{P_3}$ **return FALSE**.

return TRUE.

5.5 Computational experiments

We start by describing the considered set of benchmark instances in Section 5.5.1. Section 5.5.2 introduces some notation needed to present the numerical experiments as well as the considered routing configurations. Section 5.5.3 presents the results of the MILP, whereas Section 5.5.4 analyzes the performance of the proposed VNS. Finally, Section 5.5.5 gives the results of the VNS for the introduced set of instances.

The MILP and the VNS have been coded in C++. The MILP is solved with CPLEX 12.4 (called with the Concert Technology). Computations were performed on a 2.2 GHz Intel Core i7 with 16 Go 1600 MHz DDR3 RAM. In Algorithm 1, the parameters q_{max} and n_L were tuned to 30% and 10 respectively. In preliminary experiments, values were tested in [10%, 50%] for q_{max} and in [1, 15] for n_L .

5.5.1 Instances

The VRPTR is a new problem proposed by company EEP for which no benchmark instance exists in the literature. Focusing on urban contexts, a set of instances has been generated according to the real parameter distributions provided by EEP. The job locations are uniformly distributed in a square grid of 10 km by 10 km. The Euclidean metric is used to compute the distance between the jobs. As highlighted by [Boysen et al., 2018b], using Euclidean distances ensures that the triangle inequality is satisfied for both walking and driving. The driving speed is 30 km/h and the walking speed is 4 km/h. The job duration ranges between 20 and 35 minutes (uniformly distributed). The maximum walking time τ_M^f to reach a job on foot is 15 minutes (i.e., 1 km), and the maximum walking distance d_M^f per day and per worker is 8 km (i.e., 2 hours). The duration of the working day is 7 hours, from 8 am to 3 pm. The depot is located at the center of the considered

urban area.

Instances with $n \in \{20, 30, 40, 50\}$ are considered. Such instance sizes allow for comparing our results with VRP optimal solutions and are in line with the existing literature considering the en route synchronization of transportable resources (e.g., [Boysen et al., 2018b] solve real-world instances for up to 40 customers). In the present study, the same distance matrix is used for both walking and driving. This yields the obtained results to be a lower bound on the ones that would be obtained when walking implies shorter distances than driving (e.g., when vehicles would be constrained by one-way streets or in the presence of pedestrian walkways). Although other distance matrices could be alternatively considered, preliminary experiments have shown that similar results are found when the Manhattan metric is used, but for a slightly reduced grid size compared to the Euclidean case considered here. While the Manhattan distance could decrease the walking potential of the instances (as the distances between the jobs would increase by a factor lying in $[1, \sqrt{2}]$), the walking potential (as introduced in Section 5.6.2) of the considered instances is in line with EEP’s field observations when using the Euclidean metric.

Three service levels are envisioned by EEP. The smaller the time window, the shorter the mandatory availability for the involved client and, hence, the better the service level. Three types of time window are considered: *all day* (i.e., each job can be served in the [8 am, 3 pm] time window), *half day* (i.e., each job is associated to either the [8 am, 11:30 am] or the [11:30 am, 3 pm] time windows), and *quarter day* (i.e., each job is associated with one of the following time windows: [8 am, 9:45 am], [9:45 am, 11:30 am], [11:30 am, 1:15 pm], [1:15 pm, 3 pm]). We consider three types of instances, each of them representing one single envisioned service level. The time window assigned with each job is uniformly chosen among the possible alternatives, describing the clients’ preferences.

An instance is referred to as “ n_TW_i ”, where n stands for the number of jobs, TW represents the size of the used time window (A, H, and Q correspond, respectively, to

all day, *half day*, and *quarter day*), i characterizes the instance identifier, and n_TW denotes the set of all instances of size n and time window size TW . 10 instances have been generated for each n and each TW , leading to a total of 120 instances.

5.5.2 Notation and considered configurations

All of the 120 instances have been solved to optimality for the VRP configuration. For this purpose, we have used the algorithm proposed by [Desaulniers et al., 2008], which is acknowledged to be one of the most efficient algorithms for solving the VRP [Baldacci et al., 2012]. By assigning appropriate weights to the number of workers used in the solution, [Desaulniers et al., 2008] first minimize the number of employed workers and then minimize the total traveled distance. Accordingly, we know the associated smallest number of workers $|W^*|$ required to serve all jobs. In the following, we consider different configurations $(P_a^{|K|})$, where $|K|$ designates the number of used vehicles and $a \in \{walk, no\ walk\}$ indicates whether or not walking is allowed. $d(P_a^{|K|})$ (resp. $d^*(P_a^{|K|})$) gives the total driving distance for configuration $(P_a^{|K|})$ found by VNS (resp. the total driving distance for the optimal solution). All configurations are solved with $|W^*|$ workers (fixed by the VRP optimal solution).

The following five $(P_a^{|K|})$ are considered:

- $(P_{no\ walk}^{|W^*|})$: all workers are motorized, but they are not allowed to walk (i.e., VRP);
- $(P_{walk}^{|W^*|})$: all workers are motorized, and they are allowed to walk (i.e., *Park-and-Loop*);
- $(P_{no\ walk}^{|W^*|-1})$: carpooling is allowed, one worker is not motorized, but walking is forbidden;

- $(P_{walk}^{|W^*|-1})$: both carpooling and walking are allowed, and one worker is not motorized;
- $(P_{walk}^{|W^*|-2})$: both carpooling and walking are allowed, and two workers are not motorized.

For the considered instances and when walking was not permitted, it was never possible to remove more than one car with respect to the optimal VRP solution. When walking was allowed, it was never possible to remove more than two cars.

5.5.3 MILP results for the VRPTR

As already mentioned, a time limit of 10 hours is used. When only walking is allowed (but no carpooling), the MILP can find solutions for instances involving 20 jobs. However, when both walking and carpooling are considered, the MILP can only find solutions for instances with up to $n = 18$ jobs. A solution obtained by the MILP is shown in Figure 5.4. It exhibits both carpooling and walking, and points out the efficient synchronization that arises between a driver and a passenger. Considering the instances for which the MILP can be used to find optimal solutions, the proposed VNS finds results with a percentage gap never exceeding 1%.

5.5.4 Performance of VNS on the VRP configuration

Focusing on configuration $P_{no\ walk}^{|W^*|}$ (i.e., VRP), for all generated instance types (number of jobs and time window sizes), Table 5.1 gives the average percentage of unserved jobs in the solutions found by the VNS (column “% unserved.”), the average percentage gap of VNS with respect to the optimal values (column “% gap*”) and the percentage of

instances that could be solved to optimality (column “% opt”). “% gap*” is computed as follows: $\frac{d(P_{no\ walk}^{W^*}) - d^*(P_{no\ walk}^{W^*})}{d^*(P_{no\ walk}^{W^*})} \cdot 100$. Columns “All Day”, “Half Day”, and “Quarter Day” refer to the size of the considered time window, and each line considers the 10 instances for a given value of n . Table 5.1 shows that the VNS finds optimal solutions for 95% of the instances. For the remaining 5%, either the VNS did not find a feasible solution (i.e., some jobs remain unserved) with a number of vehicles fixed at its optimal value (see column “% unserved”), or a small gap is observed with respect to the optimal driving distance (see column “% gap*”). When some jobs are not inserted (see column “% unserved, Quarter Day”), this indicates that the VNS returned a solution with a greater number of cars used than in the VRP optimal solution. We indicate in this column the number of jobs that are not inserted when the number of cars used is constrained at its optimal value. Although the proposed VNS has been specifically designed for the VRPTR, these results contribute to validating its efficiency and consistency.

Table 5.1: Performance of VNS on configuration $P_{no\ walk}^{W^*}$.

Time Window Size	All Day			Half Day			Quarter Day		
n	% unserved	% gap*	% opt	% unserved	% gap*	% opt	% unserved	% gap*	% opt
20	0%	0%	100%	0%	0%	100%	0%	0%	100%
30	0%	0%	100%	0%	0%	100%	0.3%	0%	90%
40	0%	0%	100%	0%	0.04%	90%	0.2%	0.08%	90%
50	0%	0%	100%	0%	1.34%	90%	0.4%	0.82%	80%

5.5.5 VNS results for the VRPTR

Proportion of feasible instances for configurations involving less cars than workers.

Contrary to the *Park-and-Loop* configuration for which a VRP solution can be initially built and then improved through the introduction of walking sub-tours, the configurations involving carpooling (i.e., less cars than workers: $(P_{walk}^{W^*|-1})$, $(P_{no\ walk}^{W^*|-1})$, and

$(P_{walk}^{|W^*|-2})$) are structurally more complex. Finding a feasible solution cannot be taken for granted. Indeed, both walking (slower than driving) and carpooling (need for detours to drop off and pick up non-motorized workers) involve potential inefficiencies, and it is therefore not surprising that some instances end up unfeasible when some workers are non-motorized. While [Fikar and Hirsch, 2015] generate solutions involving less cars than workers, it comes at the price of increasing the total number of employed workers (compared with the VRP optimal solution). Here, we keep this total number of workers stable.

Figure 5.5 qualitatively highlights the potential values associated with the feasible solutions of the configurations involving less cars than workers. More precisely, three situations might arise, ranging from an improvement with respect to the *Park-and-Loop* configuration, an amelioration of the VRP solution, to not improving the one-man-one-car models (VRP and *Park-and-Loop*). Indeed, reducing the driving distance poses an additional challenge since detours to transport non-motorized workers must be efficiently compensated by merging the right paths with carpooling.

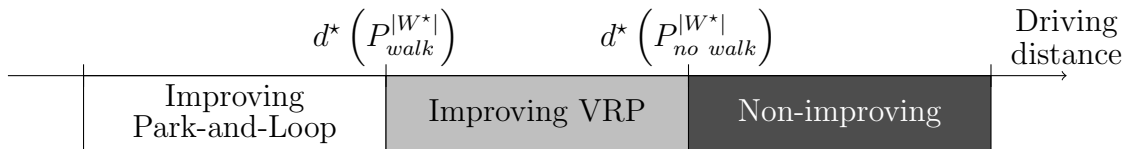


Figure 5.5: Potential driving distance of the solutions found feasible with less cars than workers.

For the three configurations involving carpooling, Table 5.2 gives the number of instances (and the associated percentage), over the 120 considered instances, that belong to each category identified in Figure 5.5. It indicates that for 55% of the instances, the VNS is able to reduce the fleet size when only carpooling is allowed (configuration $(P_{no\ walk}^{|W^*|-1})$). Moreover, the VNS finds smaller total driving distances than in the VRP solution for 6.7% of the instances, meaning that the need for detours generated by carpooling can be overcompensated by efficiently merging worker journeys. When walking is permitted in addition to the implementation of carpooling (configuration $(P_{walk}^{|W^*|-1})$), the number of

feasible instances found by the VNS grows from 55% to 56.7%, the number of instances for which VRPTR dominates VRP increases from 6.7% to 19.2%, and the proportion of instances for which VRPTR is able to improve the *Park-and-Loop* configuration ($P_{walk}^{|W^*|}$) increases from 0% to 6.7%. When two cars are removed with respect to the VRP optimal solution, the VNS finds feasible solution for 7.5% of the instances. This relatively low number can be explained by the fact that the generated instances require a maximum of 5 workers (with an average of 3.3 workers) to be solved. Thus, removing two cars represents a drastic reduction of the vehicle fleet.

Table 5.2: Proportion of feasible instances for the different configurations involving less cars than workers.

Solution characteristics	$(P_{no\ walk}^{ W^* -1})$		$(P_{walk}^{ W^* -1})$		$(P_{walk}^{ W^* -2})$	
	% Inst.	Nb. Inst.	% Inst.	Nb. Inst.	% Inst.	Nb. Inst.
Improving <i>Park-and-Loop</i>	0.0%	(0 / 120)	6.7%	(8 / 120)	0.0%	(0 / 120)
Improving VRP	6.7%	(8 / 120)	19.2%	(23 / 120)	0.0%	(0 / 120)
Non-improving	48.3%	(58 / 120)	30.8%	(37 / 120)	7.5%	(7 / 120)
Total feasible	55.0%	(66 / 120)	56.7%	(68 / 120)	7.5%	(7 / 120)

Detailed results

A.2 details the results found by VNS for all instances and all configurations. An extract of three representative instances (corresponding to the three boxes displayed in Figure 5.5) is given in Table 5.3. The “VRP” columns reflect the characteristics of the optimal VRP solutions: the number of workers required ($|W^*|$), the total driving distance (d^*), and the corresponding idle time (either en route or at the depot). The “Park-and-Loop” column gives the total driving distance found for configuration $(P_{walk}^{|W^*|})$. The “Carpooling” columns denote the associated driving distance (d) and the number of jobs that cannot be served in the solution ($|J^{out}|$) for all configurations involving carpooling (n.b., the driving distance is not displayed for unfeasible solutions). Focusing on configuration $(P_{walk}^{|W^*|-1})$, it shows that for instance 40_H_2, it improves the *Park-and-Loop* solution (driving distance reduced by 2.8%), which itself already improves the optimal VRP solution. The solution of instance 50_A_6 improves the VRP optimal solution (driving distance reduced

by 1.8%) but exhibits a driving distance 7.2% greater than in the *Park-and-Loop* solution. Instance 50_Q_5 is found feasible, but its solution returns a driving distance larger than in the optimal VRP solution.

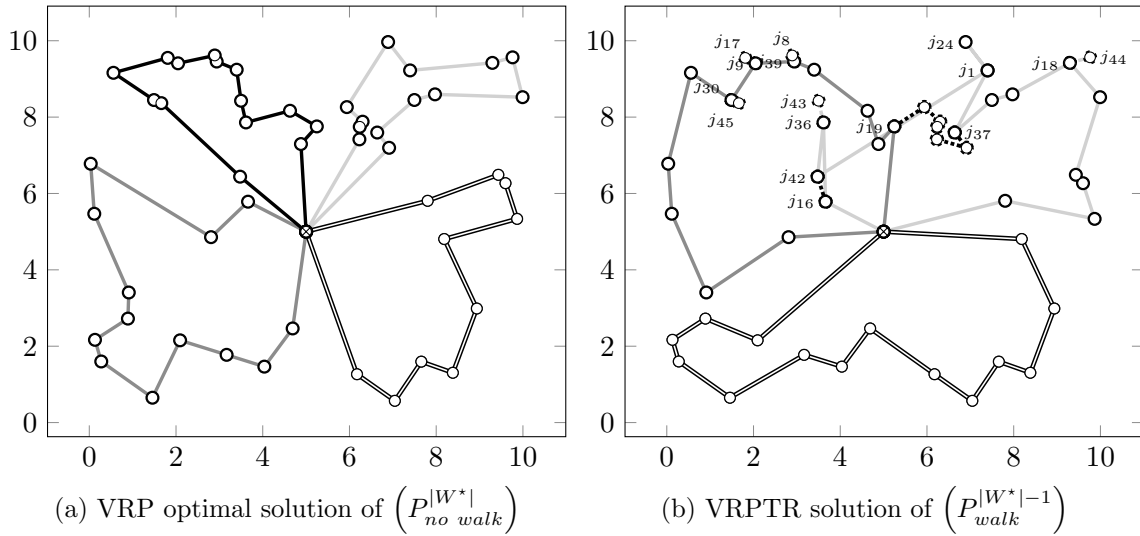
Table 5.3: Detailed results for the representative instances.

Instance	VRP $(P_{no\ walk}^{ W^* })$				Park-and-Loop $(P_{walk}^{ W^* })$ d	Carpooling					
	$ W^* $	d^*	Idle Time			$(P_{no\ walk}^{ W^* -1})$		$(P_{walk}^{ W^* -1})$		$(P_{walk}^{ W^* -2})$	
			Route	Depot		d	$ J^{out} $	d	$ J^{out} $	d	$ J^{out} $
40_H_2	4	73.1	0.0%	25.8%	63.9	75.1	0	62.1	0	94.6	0
50_A_6	4	71.8	0.0%	11.7%	66.1	78.7	0	70.9	0	-	4
50_Q_5	5	113.2	17.9%	2.7%	104.4	130.7	0	117.0	0	-	2

Figure 5.6 exemplifies a VRPTR solution (right part of the figure) for configuration $(P_{walk}^{|W^*|-1})$ in instance 50_A_6 and compares it with the optimal VRP solution (left part of the figure). In this example, carpooling and walking allow for improving the optimal VRP solution, as for the same number of employed workers, the driving distance is reduced by 1.3% and one car is saved. In this VRPTR solution, the non-motorized worker walks for 71 minutes. Note that for all performed experiments, walking never exceeds 90 minutes per worker.

5.5.6 Execution time

The stopping criterion of the VNS has been set to 10 hours (i.e., one night of computation, from 8 pm to 6 am), which follows EEP's requirements in the present one-day-ahead optimization context. Table 5.4 gives the average execution times (for each n and TW values) to obtain the best found solutions. It highlights the fact that solving $(P_{walk}^{|W^*|-1})$ is more complex than solving $(P_{walk}^{|W^*|})$. Indeed, the additional complexity of finding the best insertion position for a job when carpooling is allowed (as shown in Section 5.4.3) is reflected in these execution times. Interestingly, Table 5.4 furthermore indicates that the time window size only has a marginal impact on the execution time.



Plain (resp. dashed) lines represent the vehicle paths (resp. the walking routes). Each line type corresponds to a worker: double line for worker w_1 , light gray for w_2 , gray for w_3 , and black for w_4 (non-motorized worker in (b)). The jobs explicitly labeled are those included in a WR. In (b), w_4 is dropped off at j_{16} and walks to j_{42} . w_2 fulfills j_{36} and j_{43} in a WR before picking up w_4 at j_{42} . Before being dropped off at j_{37} , w_4 works on j_1 while w_2 serves j_{24} . The tours continue as described, and w_4 returns to the depot with w_3 .

Figure 5.6: Illustration of a VRPTR solution for which one car is saved and the total driving distance is reduced by 1.3%. The optimal VRP solution and a VRPTR solution are presented for the same instance.

Table 5.4: Average execution time of the VNS (in seconds) for each instance and time window size.

n	$(P_{walk}^{ W^* })$			$(P_{walk}^{ W^* -1})$		
	All day	Half day	Quarter day	All day	Half day	Quarter day
20	71	62	30	1,133	1,470	611
30	381	860	1,174	4,552	5,498	3,834
40	1,993	1,988	3,747	10,939	10,030	9,825
50	6,779	8,143	11,219	24,454	18,972	21,856

5.6 Managerial insights

In Section 5.6.1, we position the obtained VRPTR solutions with respect to the existing industrial practices. Next, in Section 5.6.2, we highlight the instance characteristics that influence the gain obtained with the VRPTR formulation. Finally, in Section 5.6.3, we discuss how the obtained static solutions would be expected to react when unforeseen events occur, and we propose some associated research avenues.

5.6.1 Comparison with existing practices

In Section 5.6.1, we discuss how classical VRP solutions can be improved with the introduction of walking only (i.e., results of the *Park-and-Loop*). Next, in Section 5.6.1, we show how also allowing carpooling competes with respect to the one-man-one-car models (i.e., results of the VRP and of the *Park-and-Loop*).

Benefits of the Park-and-Loop

Table 5.5 quantifies the aggregated gains provided by the *Park-and-Loop* formulation over the 120 considered instances. Each line corresponds to a specific time window and instance size (i.e., it covers 10 instances). The “VRP” columns characterizes the optimal VRP solutions (idle time, $|W^*|$ and driving distance). The “Park-and-Loop” columns display the average total driving distance (in km) found by VNS in column “ $d(P_{walk}^{|W^*|})$ ” and the corresponding percentage gap with respect to the optimal VRP solution values in column “% gap” (computed as: $\frac{f(P_{no\ walk}^{|W^*|}) - f(P_{walk}^{|W^*|})}{f(P_{no\ walk}^{|W^*|})} \cdot 100$).

Table 5.5 shows that a significant reduction in the driving distance is achieved when walking is allowed (average gain of 6.4% over the 120 instances). The gain remains stable

with the instance size, and no systematic effect can be observed from the time window size. The explanation of such an output will be discussed in Section 5.6.2. This driving distance gain is the consequence of transferring parts of the journeys traveled by car to walking, resulting in an efficient use of the available idle time present in the VRP optimal solutions. This aspect will be further discussed in Section 5.6.2. We also observe from Table 5.5 that a threshold effect on the achieved gains might appear depending on the number of involved jobs and the considered time window size. This is due to the fact that depending on the considered configuration (n, TW) , the worker plannings can be more or less saturated in the optimal VRP solutions. For example, configuration $(n = 40, TW = all\ day)$ yields optimal VRP solutions that contain little idle time for the involved workers. As explained later in Section 5.6.2, where the instance characteristics associated with efficient VRPTR solutions are discussed, this leads to less potential gain when walking and/or carpooling is implemented.

Table 5.5: Aggregated results for the *Park-and-Loop* configuration $(P_{walk}^{W^*})$.

Instance	VRP			Park-and-Loop	
	Idle time	$ W^* $	$d^*(P_{no\ walk}^{W^*})$	$d(P_{walk}^{W^*})$	%gap
20_A	24.8%	2.0	42.1	38.2	9.2%
20_H	22.5%	2.2	56.3	48.6	13.7%
20_Q	30.2%	2.4	64.7	62.5	3.4%
30_A	26.5%	3.0	53.4	46.8	12.4%
30_H	24.6%	3.0	65.3	59.5	8.9%
30_Q	27.2%	3.3	85.9	82.8	3.7%
40_A	8.4%	3.2	60.8	57.5	5.5%
40_H	18.4%	3.7	76.6	69.5	9.3%
40_Q	22.0%	4.0	98.6	93.2	5.5%
50_A	10.5%	4.0	69.3	63.1	9.0%
50_H	8.3%	4.0	87.8	83.9	4.5%
50_Q	16.8%	4.6	112.4	108.0	3.9%

Benefits of joint walking and carpooling

In their decision-making processes, managers have the choice of favoring one configuration over another to either reduce the driving distance (objective denoted as f_{dist} , in km) or the size of the vehicle fleet (objective denoted as f_{car}). Three different scenarios are

envisioned by EEP. ($S^{(dist)}$) focuses on the minimization of f_{dist} (vehicles are removed as long as the incurred detours are compensated); ($S^{(car)}$) targets f_{car} (vehicles are removed as long as all jobs can be served on time); and ($S_{(dist^*)}^{(car)}$) represents the balanced scenario where both f_{dist} and f_{car} are simultaneously considered (vehicles are removed as long as the driving distance is below the driving distance from the optimal VRP solution).

Table 5.6 presents the results for all of the above-mentioned scenarios and provides a comparison with the one-man-one-car models. The reported values are averaged over all instances sharing the same time window size. Instances are aggregated per time window size in order to avoid any misleading interpretation due to the threshold effect that might appear for some configurations (n, TW) (as observed in Section 5.6.1) and hence obtain a general understanding of the average gain that follows from the introduction of walking and carpooling. The “KPI” (Key Performance Indicator) column indicates the considered value $|K|$ (resp. d) for the size of the vehicle fleet (resp. the total driving distance). The “VRP” and “Park-and-Loop” columns reflect some of the values presented in Table 5.5. Column “%VRP” (resp. “%P&L”) gives the percentage gap with respect to the VRP solution (resp. with the *Park-and-Loop* solution).

Table 5.6: Results for both the vehicle fleet and the total driving distance for all scenarios.

Time Window Size	KPI	VRP	Park-and-Loop		$S^{(dist)}$			$S^{(car)}$			$S_{(dist^*)}^{(car)}$		
			Value	%VRP	Value	%VRP	%P&L	Value	%VRP	%P&L	Value	%VRP	%P&L
All day	$ K $	3.1	3.1	0.0%	2.9	4.1%	4.1%	2.4	23.0%	23.0%	2.7	13.1%	13.1%
	d	56.4	51.4	8.9%	51.2	9.3%	0.5%	55.3	1.9%	-7.7%	52.0	7.8%	-1.2%
Half day	$ K $	3.2	3.2	0.0%	3.1	1.6%	1.6%	2.6	17.3%	17.3%	2.9	7.1%	7.1%
	d	70.4	65.4	7.2%	65.3	7.3%	0.1%	71.3	-1.2%	-9.1%	66.4	5.8%	-1.5%
Quarter day	$ K $	3.6	3.6	0.0%	3.5	0.7%	0.7%	3.0	15.4%	15.4%	3.4	4.9%	4.9%
	d	90.4	86.6	4.2%	86.6	4.3%	0.0%	92.2	-2.0%	-6.5%	87.0	3.7%	-0.5%
Total	$ K $	3.3	3.3	0.0%	3.2	2.0%	2.0%	2.7	18.4%	18.4%	3.0	8.2%	8.2%
	d	72.4	67.8	6.4%	67.7	6.5%	0.1%	73.0	-0.7%	-7.6%	68.5	5.4%	-1.0%

Table 5.6 shows that, while keeping the number of workers stable with respect to the optimal VRP solution, the size of the vehicle fleet can be significantly reduced (scenario $S^{(car)}$). This reduction is up to 23% for instances with *all day* time windows and averages 18.4% for all instances. Carpooling allows for a further improvement of the total driving

distance compared with the one-man-one-car models (scenario $S^{(dist)}$). The improvement with respect to the VRP averages 6.5% for all instances, and it increases to 9.3% for instances with *all day* time windows. A small average gain still exists in terms of total driving distance compared to the *Park-and-Loop*, which shows that the vehicle fleet is more efficiently used. Whereas Table 5.6 indicates that the presence of *Park-and-Loop* sub-tours is responsible for most of the reduction of the driving distance, scenario $S_{(dist^*)}^{(car)}$ highlights that with an average increase of 1% of the driving distance with respect to *Park-and-Loop* solutions, savings in the number of used cars are as high as 8%.

From Table 5.6, we observe that a conflict exists between objectives f_{car} and f_{dist} as well as between the achieved gain and the implemented level of service (i.e., the time window size). Reducing f_{car} yields an increase of f_{dist} for 45.7% of the feasible instances with fewer cars. Indeed, reducing the number of cars might create a need for detours in the drivers' planning (to pick up and drop off non-motorized workers) that cannot be compensated by merging paths. However, we observe that the introduction of both carpooling and walking is able to generate a simultaneous gain, both in terms of fleet size and total driving distance, for 25% of the considered instances. Increasing the service level (i.e., shrinking the time window size) decreases the achieved gain for both f_{car} and f_{dist} . With a smaller time window size, the number of jobs that can be reached on foot decreases, which ultimately leads to an increased need for detours to drop off and pick up non-motorized workers. This will be further analyzed in Section 5.6.2. However, a reduction of 15.4% in terms of the size of the vehicle fleet is still achieved for instances with *quarter day* time windows.

To go beyond these average results with respect to the associated optimal VRP solutions, the following observations can be made. For *all day* time windows, the largest obtained reduction in terms of driving distance is 16.6% (for instance 40_A_8). For *half day* time windows, the largest achieved gain is 19% (for instance 40_H_9). Finally, for *quarter day* time windows, the largest observed reduction in driving distance is 8.3% (for instance

50_Q_1). Regarding the reduction of the fleet size, the largest gain is 50% (e.g., for instances 20_A_1, 20_H_5, and 30_Q_10). As several instances (e.g., 20_Q_3, 40_A_3) do not exhibit any gain (over the ten runs) either in terms of driving distance or with respect to the size of the vehicle fleet, the smallest achieved gain is therefore 0%. Detailed results for the 120 considered instances can be found in A.2.

5.6.2 Instance characteristics that favor carpooling

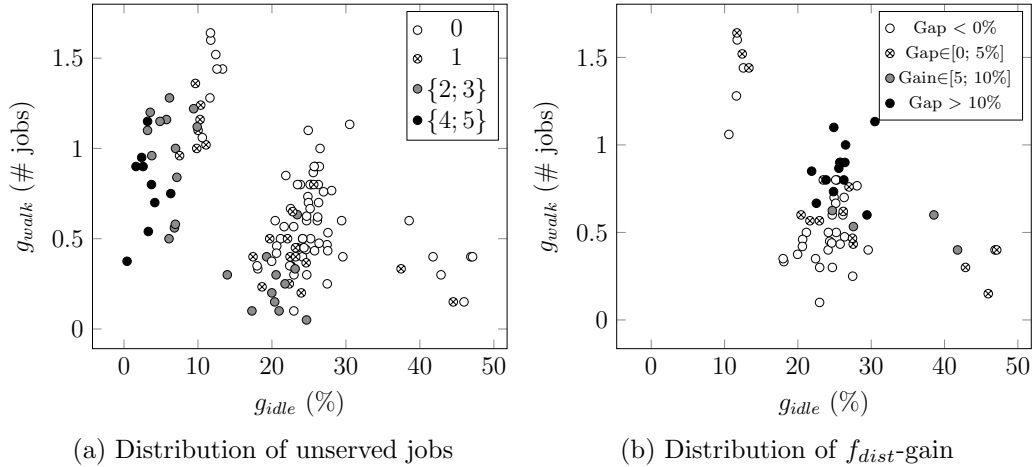
In the previous section, we analyzed the gain offered by the VRPTR formulations over the 120 considered instances. In this section, we focus on understanding which instance characteristics can be linked to the magnitude of the achieved gain.

Idle time and walking potential

Two indicators, g_{walk} and g_{idle} , are discussed. First, the walkability g_{walk} characterizes the walking potential of an instance. It represents the average number of jobs reachable on foot from a given job. A job j is said to be reachable on foot from job i if the walking time from i to j (τ_{ij}^f) is less than τ_M^f (15 minutes in our experiments) and, if leaving job i as early as possible, the worker arrives on time at job j . More precisely, let J_i^R be the set of jobs reachable from job $i \in J$. Formally, we have $J_i^R = \left\{ j \in J \mid \tau_{ij}^f \leq \tau_M^f; e_i + p_i \cdot \tau_{ij} \leq l_j \right\}$. Hence, $g_{walk} = \frac{\sum_{i \in J} |J_i^R|}{|J|}$. Depending on the instance, g_{walk} lies between 0 and 1.5 (which is in line with EEP's field observations). Second, the idle time, denoted by g_{idle} , is the percentage of time during which the workers are idle in the corresponding optimal VRP solutions. On the one hand, tightening the time window reduces g_{walk} (fewer jobs can be reached on foot without time window violations) but increases g_{idle} (waiting appears en route for the start of a job). On the other hand, g_{walk} increases with n . Indeed, the density of jobs increases, and hence, on average, more jobs can be reached on foot from

a given position. Figure 5.7 focuses on the results of $(P_{walk}^{|W^*|-1})$ according to g_{idle} and g_{walk} . Each instance is positioned on the x-axis and y-axis according to g_{idle} and to g_{walk} . More specifically, for each instance, Figure 5.7(a) gives the number of jobs that cannot be served in $(P_{walk}^{|W^*|-1})$. Figure 5.7(b) plots the gap found with the optimal VRP solution (i.e., $\frac{d^*(P_{no\ walk}^{|W^*|}) - d(P_{walk}^{|W^*|-1})}{d^*(P_{no\ walk}^{|W^*|})}$) for the 58 feasible instances for configuration $(P_{walk}^{|W^*|-1})$.

Figure 5.7(a) indicates that for instances with $g_{idle} < 10\%$, no feasible solution can be found for $(P_{walk}^{|W^*|-1})$. To put that number into perspective, $g_{idle} = 10\%$ represents an idle time of 45 minutes per worker. Figure 5.7(b) highlights that the gain increases with the walking potential g_{walk} of the instance. Moreover, for an idle time between 15% and 30%, it is necessary, from a given job, to have on average more than 0.5 jobs reachable on foot to find competitive solutions regarding the driving distance.



Each point corresponds to an instance defined by its indicators g_{idle} (x-axis) and g_{walk} (y-axis)
 (a) The number of unserved jobs is denoted by the color code given in the upper right corner.
 (b) For the feasible instances, the f_{dist} -gain range is denoted by the color code given in the upper right corner.

Figure 5.7: Distribution of the feasible instances and f_{dist} -gains for configuration $(P_{walk}^{|W^*|-1})$ (i.e., one car is removed from the VRP optimal solution).

Geographical characteristics

To keep the recommendations as general as possible, some instance characteristics that favored walking and carpooling were not explicitly taken into account in the preceding sections. Denser urban configurations are likely to appear in practice, as the considered instances have a job density between 0.2 and 0.5 jobs per km². Ultimately, only 3% of the arcs are actually eligible for traveling on foot. A greater density of jobs per km² would increase g_{walk} and, hence, the efficiency of the VRPTR formulation, as discussed in Section 5.6.2. Considering congestion or parking time explicitly would also substantially reduce the gap between walking time and driving time for some arcs. Additionally, walking would sometimes become a mandatory option in pedestrian zones.

Other experiments (not reported here) have shown that for the 40 instances with *all day* time windows, considering a non-centered depot (located at one of the corners of the 10 km by 10 km square grid) increases the efficiency of scenario $S^{(car)}$, as the f_{car} -gain goes up from 23% to 29.2% on average, whereas the f_{dist} -gain jumps from 1.9% to 8.8%. Indeed, in such a situation, the optimal VRP solutions are likely to route workers through close paths from the depot to the customer locations at the beginning of the working day and back to the depot at the end of it. A consolidation of these travels to and from the depot is hence expected to arise.

5.6.3 Expected impact of random perturbations

In this paper, we considered a static case where all the service times and travel data are assumed to be known. The aim of this study is to measure the benefits of walking and carpooling. However, in practice, the actual service and travel times are likely to differ from the forecasted ones. In the following, we discuss the expected impacts resulting from such perturbations and what mechanisms could be envisioned to overcome such issues.

Compared with VRP solutions, the routes involving carpooling are interdependent. As a consequence, an unexpected event on one route can potentially modify the schedule of all the interconnected routes. For example, considering the VRPTR solution displayed in Figure 5.6, if worker w_3 is delayed on her/his route (gray line) because of longer service or travel times, s/he will only be able to pick up the non-motorized worker w_4 behind schedule, which would ultimately lead to a late arrival at the depot for both workers. Robust approaches, such as those derived in the VRP context (e.g., [Lu and Gzara, 2019]), could be extended to the VRPTR case. In the context of online VRP (see [Pillac et al., 2013] for a recent review), [Lorini et al., 2011] and [Respen et al., 2019] propose a solution method to rebuild – in real-time – solutions that have been modified by unexpected events. Their methods rely on fast moves, including the reassignment of a limited number of jobs or vehicle diversion (i.e., modify the current destination of the vehicle). In the VRPTR situation, taxi services [Zufferey et al., 2016] could also be envisioned as an urgency backup. Removing future WRs in driver routes would also be an appropriate technique for overcoming delays in their routes.

5.7 Conclusion, perspectives, and future works

This study considers a new type of VRP called the *Vehicle Routing Problem with Transportable Resources* (VRPTR), where vehicles and walking workers coexist and must be synchronized to satisfy a given set of jobs spread over a territory. Such a formulation has not yet been introduced in the literature, and it opens up new perspectives in decision-making processes for routing problems. The proposed modeling framework is suitable for each situation in which two distinct transportation resources are available: the transporter ones (e.g., cars, trucks, or buses), which move autonomously, and the transportable ones (e.g., pedestrians, scooters, bicycles, or drones), which can either move autonomously or be transported by a transporter one for parts of their route.

In this contribution, we treat an application of the VRPTR that comes from an industrial case. It involves cars, walking, and carpooling. Coordinating all these transportation options allows managers to generate a brand new set of solutions in a routing context. Considering an urban context, we evaluate the potential of such a novel formulation, as simultaneous savings can be achieved both in the total driving distance and in the number of vehicles, even with tight time windows. Obviously, when focusing on these two objectives, the gains obtained depend on the instance characteristics. We have identified some conditions under which a significant gain can be expected. For instances involving idle times (inside the routes due to time window constraints or at the end of the routes), the new formulation is able to invest them efficiently in carpooling and walking. Further works could explore in detail the trade-off that arises between decreasing the need for resources and the total en route time, as done in a Green VRP context in [Demir et al., 2014].

5.8 Acknowledgement

The authors would like to thank Professor Guy Desaulniers (Polytechnique Montreal and GERAD, Canada) for proving the optimality of the VRP solutions with GENCOL. Thanks are due to the reviewers for their valuable comments. This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Chapter 6

General conclusion

This thesis presents four independent chapters that are inspired by different industrial partners. This chapter discusses common conclusions from the different projects. First, I review the scientific contributions. Next, I sketch out the future works envisioned.

6.1 Scientific contributions

First, I introduced new and relevant problems, directly derived from industry. These problems occur at different echelons of the supply chain and are related to the transportation of goods or persons.

In the first part of this thesis (Chapters 2 and 3), I focus on cross-docking platforms used to consolidate flows of products. More precisely, I integrate three \mathcal{NP} -hard problems (i.e., the traveling purchaser problem [Manerba et al., 2017], container loading problems [Toffolo et al., 2017], and container scheduling [Boysen et al., 2018b]), such an integration has never been addressed.

In the second part of this thesis, I introduce an extension of the *Vehicle Routing Problem*, called *Vehicle Routing Problem with Transportable Resources* (VRPTR). It allows integrating drone flights into traditional deliveries by trucks (Chapter 4). The trucks-and-drones problems have received considerable attention from researchers in the past five years, but I consider specific characteristics that have not been addressed in the literature. The VRPTR also allows integrating walking and carpooling into vehicle routing (Chapter 5). Walking and carpooling have never been integrated into a routing context, and doing so has great potential in the context of growing urbanization with limited access to city center. In the context of major concern regarding human activities and pollution, these two extensions have a great potential for greenhouse gas reduction.

I show the practical relevance of these new problems by conducting extensive computational experiments and sensitivity analyses on real or realistic instances. I identify that significant savings can be achieved when compared to current industrial practices. I show that it is possible to improve the performance of several indicators of the supply chain. There is room for reducing not only costs but also the supply chain's environmental impact. It is also possible to find solutions that also improve working conditions. For instance, in Chapters 2 and 3, I describe how multiple problems can be integrated and solved simultaneously to reduce the storage required and the gap between the most and least loaded day in cross-docking platforms. Compared with the case where all problems are solved individually in a sequential manner, which is the current industrial practice, I have been able to identify gains of more than 50% on the real instances provided by the industrial partners. In Chapter 4, by using drones in tandem with trucks, I identify a potential cost reduction of up to 35% in cases where all the jobs can be served by drones. Finally, in Chapter 5, compared with a case where all workers operate their own vehicle, carpooling and walking allow decreasing the size of the vehicle fleet by 8% and the total driving distance by 5%.

I also propose several methodological contributions. First, I point out the efficiency of

using heuristics based on the destroy-and-repair principle to solve problems of various types. These methods require an insertion heuristic able to reconstruct a solution from a partial solution and a removal heuristic able to build a partial solution from a feasible solution. A partial solution denotes a solution in which only a subset of variables can be changed (the other variables are fixed). Using the insertion and removal heuristics iteratively allows finding effective solutions in very different contexts (as shown in the results displayed above). This thesis presents several ways to implement a destroy-and-repair solution method. I present a *Fix-and-Optimize Metaheuristic* (FOM) (Chapters 2 and 3). The FOM acknowledges the interest of using mathematical programming during the reconstruction phase. I also present an *Adaptive Large Neighborhood Search* (ALNS) (Chapter 4) and a *Variable Neighborhood Search* (VNS) (Chapter 5). The strengths of the latter methods rely on the diversity of the insertion heuristics used to repair the solution. From a general viewpoint, considering as many possible specific features of the problem as possible and adapting the solution method to these features improves the output. In Chapters 4 and 5 (i.e., in the routing context), I detail algorithms (called the fast feasibility check and fast cost evaluation) used to speed up the insertion heuristics. Without this speed-up phase, I would not have been able to solve realistic instances within the allowed execution time.

Moreover, heuristics based on the destroy-and-repair principle act as integrated solution methods; they can make decisions on the several parts of the solution simultaneously. I compare such integrated solution methods with decomposition solution methods that decompose the main problem into smaller sub-problems. The sub-problems are solved successively, and the output of a sub-problem becomes the input of the next one. I show that integrated solution methods that can revoke decisions on several parts of the solution simultaneously significantly improves the results of the decomposition approach. However, it comes at the price of an increased execution time. For instance, Chapter 3 indicates that the decomposition method that independently solves the truck-loading problems and then the containers-loading problem produces results (e.g., require storage

space) that can be improved by 10% by the FOM regarding the objective function. FOM can make decisions on the content of trucks and containers simultaneously. In Chapter 4, similarly, for the truck-and-drone problem, a decomposition method that first builds truck routes and then assigns the jobs to drones provides solutions that the ALNS improves by 8% on average.

6.2 Future work

I expect to use the proposed solution methods on other complex logistics problems. The modeling framework used to integrate the complex loading constraints with the container scheduling problem (i.e., how and when to load containers, see Chapters 2 and 3) is generic. It is based on product permutation and allows significantly decreasing the complexity of the problem while keeping a rather important number of solutions. It can be applied in any case where several products can be transported by the same box-type. The methods developed allow testing a huge number of feasible loading assignments by moving products from one box to another. Other industrial contexts also propose integrating loading constraints with another optimization problem (e.g., the routing with 3D loading constraint [Bortfeldt et al., 2015]). These contributions do not use the abovementioned simplification to address the loading problem. Based on the good results found with the proposed modeling framework, I expect to find competitive algorithms in those cases with the proposed simplification. Moreover, the FOM is not frequently used (a few papers directly refer to fix-and-optimize procedures). Because of its advantages (e.g., efficient solutions are provided, only one mathematical model needs to be maintained, and it allows new features to be easily introduced into the problem), I expect to use this methodology in several other contexts to assess its performance more precisely.

Also, in the situations considered in this thesis, I focus on building efficient solutions in a static case (i.e., the information is known in advance). In future work, I expect to

examine how a dynamic context impacts the proposed solution. As these solutions are of considerable interest to industry, a natural next step is to evaluate their robustness. Indeed, managing highly synchronized solutions raises the question of robustness (e.g., the routes proposed in Chapters 4 and 5 are interconnected, and the container scheduling strongly depends on products received in Chapters 2 and 3). What would be the impact of an unexpected event on the proposed solutions? For instance, if a truck is delayed, a drone or non-motorized worker might not be picked up on time, and hence, the remaining part of the solution might turn out to be infeasible. I see two potential axes for future works. First, I could propose robust solutions. Robust solutions are built so that the deviation in the objective function is minimized given a total level of hazard in the data. Another research axis is to propose real-time algorithms that recompute a feasible solution each time an unexpected event precludes continuing to use the current solution.

Both problems described in Chapters 4 and 5 can be generalized as the VRPTR. In this thesis, because of the specific constraints found in Chapter 4 and in Chapter 5, I proposed two different solution methods for each problem. Indeed, by splitting the modeling, I could make some assumptions on each problem that speed up the insertion phase of the destroy-and-repair procedure. As future work, it would be interesting to propose a common and generic model for the VRPTR that can solve the two problems. Indeed, unifying the modeling framework could have several advantages. First of all, improving an algorithm would be beneficial for both versions of the problem. Also, integrating walking and drones in the same solution could lead to very effective solutions.

Finally, the problems proposed in this thesis can be extended, and the proposed solution methods could be tested on these extended problems. Among the potential extensions, I envision to consider the customer priority in the cross-docking platforms described in Chapters 2 and 3, (i.e., sent as soon as possible the products concerning customers with high priority). In the same context, I could also integrate the design of truck routes to the cross-docking operations. Regarding the second part of the thesis, myriad of extensions

can be proposed: increase the drone's loading capacity in Chapter 4, consider multiple transportation means for walking workers in Chapter 5 (e.g., public transport or bike). All these extensions are relevant for the industry.

Abbreviations

- ALNS: Adaptive Large Neighborhood Search
- APVRP: Active Passive Vehicle Routing Problem
- BP: Branch and Price
- DM: Decomposition Matheuristic
- DP: Dynamic Programming
- ECM: European Car Manufacturer
- EEP: European Energy Provider
- ELP: European Logistics Provider
- FOM: Fix-and-Optimize Matheuristic
- GRASP: Greedy Randomized Adaptive Search Procedure
- ILP: Intermodal Logistics Platform
- LB: Lower Bound
- LNS: Large Neighborhood Search
- MC-VRPTW-D: Minimum Cost Vehicle Routing Problem with Time Windows and Drones
- MH: Matheuristic
- MILP: Mixed Integer Linear Programming
- MKP: Multidimensional Knapsack Problem
- RFCS: Route First Cluster Second
- SA: Simulated Annealing

- TSCD: Truck Scheduling Cross-Dock
- TDM: Temporal Decomposition Matheuristic
- VNS: Variable Neighborhood Search
- VND: Variable Neighborhood Descent
- VRP: Vehicle Routing Problem
- VRPTR: Vehicle Routing Problem with Transportable Resources
- VRPTT: Vehicle Routing Problem with Trailer and Transshipment
- VRPTW: Vehicle Routing Problem with Time Windows
- TW: Time Window
- WR: Walking Route

Bibliography

- [Agatz et al., 2018] Agatz, N., Bouman, P., and Schmidt, M. (2018). Optimization approaches for the traveling salesman problem with drone. *Transportation Science*, 52(4):965–981.
- [Amazon, 2016] Amazon (2016). Unmanned aerial vehicles in logistics. <https://www.amazon.com/Amazon-Prime-Air/b?ie=UTF8&node=8037720011>.
- [Archetti and Speranza, 2014] Archetti, C. and Speranza, M. G. (2014). A survey on metaheuristics for routing problems. *EURO Journal on Computational Optimization*, 2(4):223–246.
- [Baldacci et al., 2012] Baldacci, R., Mingozzi, A., and Roberti, R. (2012). Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. *European Journal of Operational Research*, 218(1):1–6.
- [Bellanger et al., 2013] Bellanger, A., Hanafi, S., and Wilbaut, C. (2013). Three-stage hybrid-flowshop model for cross-docking. *Computers & Operations Research*, 40:1109–1121.
- [Boctor et al., 2003] Boctor, F. F., Laporte, G., and Renaud, J. (2003). Heuristics for the traveling purchaser problem. *Computers & Operations Research*, 30(4):491–504.
- [Bodin et al., 2000] Bodin, L., Mingozzi, A., Baldacci, R., and Ball, M. (2000). The rollon–rolloff vehicle routing problem. *Transportation Science*, 34(3):271–288.

- [Bortfeldt et al., 2015] Bortfeldt, A., Hahn, T., Männel, D., and Mönch, L. (2015). Hybrid algorithms for the vehicle routing problem with clustered backhauls and 3d loading constraints. *European Journal of Operational Research*, 243(1):82–96.
- [Bouman et al., 2018] Bouman, P., Agatz, N., and Schmidt, M. (2018). Dynamic programming approaches for the traveling salesman problem with drone. *Networks*, 72(4):528–542.
- [Boysen, 2010] Boysen, N. (2010). Truck scheduling at zero-inventory cross docking terminals. *Computers & Operations Research*, 37(1):32–41.
- [Boysen et al., 2018a] Boysen, N., Briskorn, D., Fedtke, S., and Schwerdfeger, S. (2018a). Drone delivery from trucks: Drone scheduling for given truck routes. *Networks*, 72(4):506–527.
- [Boysen and Fliedner, 2010] Boysen, N. and Fliedner, M. (2010). Cross dock scheduling: Classification, literature review and research agenda. *Omega*, 38(6):413–422.
- [Boysen et al., 2018b] Boysen, N., Schwerdfeger, S., and Weidinger, F. (2018b). Scheduling last-mile deliveries with truck-based autonomous robots. *European Journal of Operational Research*, 271(3):1085–1099.
- [Buijs et al., 2014] Buijs, P., Vis, I. F., and Carlo, H. J. (2014). Synchronization in cross-docking networks: A research classification and framework. *European Journal of Operational Research*, 239(3):593–608.
- [Carlsson and Song, 2017] Carlsson, J. G. and Song, S. (2017). Coordinated logistics with a truck and a drone. *Manag Science (to appear)*, 0(0):null.
- [Cattaruzza et al., 2017] Cattaruzza, D., Absi, N., Feillet, D., and González-Feliu, J. (2017). Vehicle routing problems for city logistics. *EURO Journal on Transportation and Logistics*, 6(1):51–79.

- [Cattaruzza et al., 2018] Cattaruzza, D., Brotcorne, L., Semet, F., and Tounsi, B. (2018). A three-phase matheuristic for the packaging and shipping problem. *Applied Mathematical Modelling*, 64:713–732.
- [Chao, 2002] Chao, I.-M. (2002). A tabu search method for the truck and trailer routing problem. *Computers & Operations Research*, 29(1):33–51.
- [Chen and Lee, 2009] Chen, F. and Lee, C.-Y. (2009). Minimizing the makespan in a two-machine cross-docking flow shop problem. *European Journal of Operational Research*, 193(1):59–72.
- [Chen, 2015] Chen, H. (2015). Fix-and-optimize and variable neighborhood search approaches for multi-level capacitated lot sizing problems. *Omega*, 56:25–36.
- [Cherkassky et al., 2009] Cherkassky, B. V., Georgiadis, L., Goldberg, A. V., Tarjan, R. E., and Werneck, R. F. (2009). Shortest-path feasibility algorithms: An experimental evaluation. *Journal of Experimental Algorithmics*, 14:7.
- [Coindreau et al., 2019a] Coindreau, M.-A., Gallay, O., and Zufferey, N. (2019a). Vehicle routing with transportable resources: Using carpooling and walking for on-site services. *European Journal of Operational Research*, 279(3):996–1010.
- [Coindreau et al., 2019b] Coindreau, M.-A., Gallay, O., Zufferey, N., and Laporte, G. (2019b). Integrating workload smoothing and inventory reduction in three intermodal cross-docking platforms of a European car manufacturer. *Computers & Operations Research*, 112:104762.
- [Cordeau et al., 2007] Cordeau, J.-F., Laporte, G., Savelsbergh, M. W., and Vigo, D. (2007). Vehicle routing. *Handbooks in Operations Research and Management Science*, 14:367–428.
- [Daimler, 2017] Daimler (2017). Vans & robots paketbote 2.0. <https://www.daimler.com/innovation/specials/future-transportation-vans/paketbote-2-0.html>.

- [Dechter et al., 1991] Dechter, R., Meiri, I., and Pearl, J. (1991). Temporal constraint networks. *Artificial Intelligence*, 49(1-3):61–95.
- [Della Croce and Salassa, 2014] Della Croce, F. and Salassa, F. (2014). A variable neighborhood search based matheuristic for nurse rostering problems. *Annals of Operations Research*, 218(1):185–199.
- [Demir et al., 2014] Demir, E., Bektaş, T., and Laporte, G. (2014). The bi-objective pollution-routing problem. *European Journal of Operational Research*, 232(3):464–478.
- [Derigs et al., 2013] Derigs, U., Pullmann, M., and Vogel, U. (2013). Truck and trailer routing problems, heuristics and computational experience. *Computers & Operations Research*, 40(2):536–546.
- [Desaulniers et al., 2008] Desaulniers, G., Lessard, F., and Hadjar, A. (2008). Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. *Transportation Science*, 42(3):387–404.
- [DHL, 2014] DHL (2014). First prime air delivery. www.logistics.dhl/content/dam/dhl/global/core/documents/pdf/glo-logistics-insights-uav-trend-report.pdf.
- [Domínguez-Martín et al., 2018] Domínguez-Martín, B., Rodríguez-Martín, I., and Salazar-González, J.-J. (2018). The driver and vehicle routing problem. *Computers & Operations Research*, 92:56–64.
- [Dorneles et al., 2014] Dorneles, Á. P., de Araújo, O. C., and Buriol, L. S. (2014). A fix-and-optimize heuristic for the high school timetabling problem. *Computers & Operations Research*, 52:29–38.
- [Drexler, 2012] Drexler, M. (2012). Synchronization in vehicle routing: a survey of VRPs with multiple synchronization constraints. *Transportation Science*, 46(3):297–316.
- [Drexler, 2014] Drexler, M. (2014). Branch-and-cut algorithms for the vehicle routing problem with trailers and transshipments. *Networks*, 63(1):119–133.

- [Emde et al., 2010] Emde, S., Boysen, N., and Armin, S. (2010). Balancing mixed-model assembly lines: a computational evaluation of objectives to smoothen workload. *International Journal of Production Research*, 48(11):3173–3191.
- [Enderer et al., 2017] Enderer, F., Contardo, C., and Contreras, I. (2017). Integrating dock-door assignment and vehicle routing with cross-docking. *Computers & Operations Research*, 88:30–43.
- [Ferrandez et al., 2016] Ferrandez, S. M., Harbison, T., Weber, T., Sturges, R., and Rich, R. (2016). Optimization of a truck-drone in tandem delivery network using k-means and genetic algorithm. *Journal of Industrial Engineering and Management*, 9(2):374.
- [Fikar and Hirsch, 2015] Fikar, C. and Hirsch, P. (2015). A matheuristic for routing real-world home service transport systems facilitating walking. *Journal of Cleaner Production*, 105:300–310.
- [Fikar and Hirsch, 2017] Fikar, C. and Hirsch, P. (2017). Home health care routing and scheduling: A review. *Computers & Operations Research*, 77:86–95.
- [Gendreau et al., 2006] Gendreau, M., Iori, M., Laporte, G., and Martello, S. (2006). A tabu search algorithm for a routing and container loading problem. *Transportation Science*, 40(3):342–350.
- [Ghiani and Laporte, 2001] Ghiani, G. and Laporte, G. (2001). Location-arc routing problems. *Opsearch*, 38(2):151–159.
- [Gintner et al., 2005] Gintner, V., Kliwer, N., and Suhl, L. (2005). Solving large multiple-depot multiple-vehicle-type bus scheduling problems in practice. *OR Spectrum*, 27(4):507–523.
- [Grangier et al., 2016] Grangier, P., Gendreau, M., Lehuédé, F., and Rousseau, L.-M. (2016). An adaptive large neighborhood search for the two-echelon multiple-trip vehicle routing problem with satellite synchronization. *European Journal of Operational Research*, 254(1):80 – 91.

- [Gussmagg-Pfieggl et al., 2011] Gussmagg-Pfieggl, E., Tricoire, F., Doerner, K., and Hartl, R. (Cadiz, 2011). Mail-delivery problems with park-and-loop tours: a heuristic approach. In *Proceedings of the ORP3 Meeting*.
- [Ha et al., 2018] Ha, Q. M., Deville, Y., Pham, Q. D., and Hà, M. H. (2018). On the min-cost traveling salesman problem with drone. *Transportation Research Part C: Emerging Technologies*, 86:597–621.
- [Ham, 2018] Ham, A. M. (2018). Integrated scheduling of m-truck, m-drone, and m-depot constrained by time-window, drop-pickup, and m-visit using constraint programming. *Transportation Research Part C: Emerging Technologies*, 91:1–14.
- [Heilporn et al., 2011] Heilporn, G., Cordeau, J.-F., and Laporte, G. (2011). An integer L-shaped algorithm for the Dial-a-Ride Problem with stochastic customer delays. *Discrete Applied Mathematics*, 159 (9):883 – 895.
- [Helber and Sahling, 2010] Helber, S. and Sahling, F. (2010). A fix-and-optimize approach for the multi-level capacitated lot sizing problem. *International Journal of Production Economics*, 123(2):247–256.
- [Hollis et al., 2006] Hollis, B., Forbes, M., and Douglas, B. (2006). Vehicle routing and crew scheduling for metropolitan mail distribution at australia post. *European Journal of Operational Research*, 173(1):133–150.
- [Jabali et al., 2012] Jabali, O., Woensel, T., and de Kok, A. (2012). Analysis of travel times and CO2 emissions in time-dependent vehicle routing. *Production and Operations Management*, 21(6):1060–1074.
- [Jourdan et al., 2009] Jourdan, L., Basseur, M., and Talbi, E.-G. (2009). Hybridizing exact methods and metaheuristics: A taxonomy. *European Journal of Operational Research*, 199(3):620–629.
- [Kim et al., 2006] Kim, B.-I., Kim, S., and Sahoo, S. (2006). Waste collection vehicle routing problem with time windows. *Computers & Operations Research*, 33(12):3624–3642.

- [Kirkpatrick et al., 1983] Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680.
- [Knörr, 2008] Knörr, W. (2008). Ecotransit: ecological transport information tool, environmental methodology and data. Technical report, Institut für Energie (ifeu) und Umweltforschung Heidelberg GmbH, http://www.ecotransit.org/download/ecotransit_background_report.pdf.
- [Ladier and Alpan, 2016] Ladier, A.-L. and Alpan, G. (2016). Cross-docking operations: Current research versus industry practice. *Omega*, 62:145–162.
- [Ladier et al., 2014] Ladier, A.-L., Alpan, G., and Penz, B. (2014). Joint employee weekly timetabling and daily rostering: A decision-support tool for a logistics platform. *European Journal of Operational Research*, 234(1):278–291.
- [Lam et al., 2015] Lam, E., Van Hentenryck, P., and Kilby, P. (2015). Joint vehicle and crew routing and scheduling. In *International Conference on Principles and Practice of Constraint Programming*, pages 654–670. Springer.
- [Laporte, 2009] Laporte, G. (2009). Fifty years of vehicle routing. *Transportation Science*, 43(4):408–416.
- [Larbi et al., 2011] Larbi, R., Alpan, G., Baptiste, P., and Penz, B. (2011). Scheduling cross docking operations under full, partial and no information on inbound arrivals. *Computers & Operations Research*, 38(6):889–900.
- [Levy and Bodin, 1989] Levy, L. and Bodin, L. (1989). The arc oriented location routing problem. *INFOR: Information Systems and Operational Research*, 27(1):74–94.
- [Lin, 2008] Lin, C. (2008). A cooperative strategy for a vehicle routing problem with pickup and delivery time windows. *Computers & Industrial Engineering*, 55(4):766–782.

- [Lin et al., 2009] Lin, S.-W., Vincent, F. Y., and Chou, S.-Y. (2009). Solving the truck and trailer routing problem based on a simulated annealing heuristic. *Computers & Operations Research*, 36(5):1683–1692.
- [Lorini et al., 2011] Lorini, S., Potvin, J.-Y., and Zufferey, N. (2011). Online vehicle routing and scheduling with dynamic travel times. *Computers & Operations Research*, 38(7):1086–1090.
- [Lu and Gzara, 2019] Lu, D. and Gzara, F. (2019). The robust vehicle routing problem with time windows: Solution by branch and price and cut. *European Journal of Operational Research*, 275(3):925–938.
- [Maknoon and Laporte, 2017] Maknoon, Y. and Laporte, G. (2017). Vehicle routing with cross-dock selection. *Computers & Operations Research*, 77:254–266.
- [Maknoon et al., 2017] Maknoon, Y., Soumis, F., and Baptiste, P. (2017). An integer programming approach to scheduling the transshipment of products at cross-docks in less-than-truckload industries. *Computers & Operations Research*, 82:167–179.
- [Manerba et al., 2017] Manerba, D., Mansini, R., and Riera-Ledesma, J. (2017). The traveling purchaser problem and its variants. *European Journal of Operational Research*, 259(1):1–18.
- [Masson et al., 2013] Masson, R., Lehuédé, F., and Péton, O. (2013). Efficient feasibility testing for request insertion in the pickup and delivery problem with transfers. *Operations Research Letters*, 41(3):211–215.
- [Masson et al., 2014] Masson, R., Lehuédé, F., and Péton, O. (2014). The dial-a-ride problem with transfers. *Computers & Operations Research*, 41:12–23.
- [Meisel and Kopfer, 2014] Meisel, F. and Kopfer, H. (2014). Synchronized routing of active and passive means of transport. *OR spectrum*, 36(2):297–322.

- [Merengo et al., 1999] Merengo, C., Nava, F., and Pozetti, A. (1999). Balancing and sequencing manual mixed-model assembly lines. *International Journal of Production Research*, 37:2835–2860.
- [Mladenović and Hansen, 1997] Mladenović, N. and Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100.
- [Murray and Chu, 2015] Murray, C. C. and Chu, A. G. (2015). The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. *Transportation Research Part C: Emerging Technologies*, 54:86–109.
- [Otto et al., 2018] Otto, A., Agatz, N., Campbell, J., Golden, B., and Pesch, E. (2018). Optimization approaches for civil applications of unmanned aerial vehicles (UAVs) or aerial drones: A survey. *Networks*, 72(4):411–458.
- [Papageorgiou et al., 2018] Papageorgiou, D. J., Cheon, M.-S., Harwood, S., Trespalacios, F., and Nemhauser, G. L. (2018). Recent progress using matheuristics for strategic maritime inventory routing. In *Konstantopoulos, C., Pantziou, G. (eds) Modeling, Computing and Data Handling Methodologies for Maritime Transportation*, volume 131, pages 59–94. Intelligent Systems Reference Library, Springer, Cham.
- [Parragh and Cordeau, 2017] Parragh, S. and Cordeau, J.-F. (2017). Branch-and-price and adaptive large neighborhood search for the truck and trailer routing problem with time windows. *Computer & Operations Research*, 83:28–44.
- [Pillac et al., 2013] Pillac, V., Gendreau, M., Guéret, C., and Medaglia, A. L. (2013). A review of dynamic vehicle routing problems. *European Journal of Operational Research*, 225(1):1–11.
- [Pisinger, 1997] Pisinger, D. (1997). A minimal algorithm for the 0-1 knapsack problem. *Operations Research*, 45(5):758–767.
- [Pisinger and Ropke, 2007] Pisinger, D. and Ropke, S. (2007). A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34(8):2403–2435.

- [Poikonen et al., 2019] Poikonen, S., Golden, B., and Wasil, E. A. (2019). A branch-and-bound approach to the traveling salesman problem with a drone. *INFORMS Journal on Computing*, 31(2):335–346.
- [Poikonen et al., 2017] Poikonen, S., Wang, X., and Golden, B. (2017). The vehicle routing problem with drones: Extended models and connections. *Networks*.
- [Ponza, 2016] Ponza, A. (2016). Optimization of drone-assisted parcel delivery. Master’s thesis, University of Padova, Italy.
- [Puchinger et al., 2010] Puchinger, J., Raidl, G. R., and Pferschy, U. (2010). The multidimensional knapsack problem: Structure and algorithms. *INFORMS Journal on Computing*, 22(2):250–265.
- [Pugliese and Guerriero, 2017] Pugliese, L. D. P. and Guerriero, F. (2017). Last-mile deliveries by using drones and classical vehicles. In *International Conference on Optimization and Decision Science*, pages 557–565. Springer.
- [Respen et al., 2019] Respen, J., Zufferey, N., and Potvin, J.-Y. (2019). Impact of On-line Tracking on a Vehicle Routing Problem with Dynamic Travel Times. *RAIRO-Operations Research*, 53:401 – 414.
- [Rieck et al., 2014] Rieck, J., Ehrenberg, C., and Zimmermann, J. (2014). Many-to-many location-routing with inter-hub transport and multi-commodity pickup-and-delivery. *European Journal of Operational Research*, 236(3):863–878.
- [Rochat and Semet, 1994] Rochat, Y. and Semet, F. (1994). A tabu search approach for delivering pet food and flour in switzerland. *Journal of the Operational Research Society*, 45(11):1233–1246.
- [Ropke and Pisinger, 2006] Ropke, S. and Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4):455–472.

- [Sacramento et al., 2019] Sacramento, D., Pisinger, D., and Ropke, S. (2019). An adaptive large neighborhood search metaheuristic for the vehicle routing problem with drones. *Transportation Research Part C: Emerging Technologies*, 102:289–315.
- [Sahling et al., 2009] Sahling, F., Buschkühl, L., Tempelmeier, H., and Helber, S. (2009). Solving a multi-level capacitated lot sizing problem with multi-period setup carry-over via a fix-and-optimize heuristic. *Computers & Operations Research*, 36(9):2546–2553.
- [Savelsbergh, 1992] Savelsbergh, M. W. (1992). The vehicle routing problem with time windows: minimizing route duration. *ORSA Journal On Computing*, 4(2):146–154.
- [Schermer et al., 2019a] Schermer, D., Moeini, M., and Wendt, O. (2019a). A hybrid vns/tabu search algorithm for solving the vehicle routing problem with drones and en route operations. *Computers & Operations Research*, 109:134–158.
- [Schermer et al., 2019b] Schermer, D., Moeini, M., and Wendt, O. (2019b). A matheuristic for the vehicle routing problem with drones and its variants. *Transportation Research Part C: Emerging Technologies*, 106:166–204.
- [Schrimpf et al., 2000] Schrimpf, G., Schneider, J., Stamm-Wilbrandt, H., and Dueck, G. (2000). Record breaking optimization results using the ruin and recreate principle. *Journal of Computational Physics*, 159(2):139–171.
- [Semet, 1995] Semet, F. (1995). A two-phase algorithm for the partial accessibility constrained vehicle routing problem. *Annals of Operations Research*, 61(1):45–65.
- [Semet and Taillard, 1993] Semet, F. and Taillard, E. (1993). Solving real-life vehicle routing problems efficiently using tabu search. *Annals of Operations research*, 41(4):469–488.
- [Serrano et al., 2017] Serrano, C., Delorme, X., and Dolgui, A. (2017). Scheduling of truck arrivals, truck departures and shop-floor operation in a cross-dock platform, based on trucks loading plans. *International Journal of Production Economics*, 194:102–112.

- [Shaw, 1997] Shaw, P. (1997). A new local search algorithm providing high quality solutions to vehicle routing problems. Technical report, Department of Computer Science, University of Strathclyde, Scotland.
- [Shaw, 1998] Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. *Lecture Notes in Computer Science*, 1520:417–431.
- [Smilowitz, 2006] Smilowitz, K. (2006). Multi-resource routing with flexible tasks: an application in drayage operations. *Iie Transactions*, 38(7):577–590.
- [Speranza, 2018] Speranza, M. G. (2018). Trends in transportation and logistics. *European Journal of Operational Research*, 264:830–836.
- [Tilk et al., 2017] Tilk, C., Bianchessi, N., Drexl, M., Irnich, S., and Meisel, F. (2017). Branch-and-price-and-cut for the active-passive vehicle-routing problem. *Transportation Science*, 52(2):300–319.
- [Toffolo et al., 2017] Toffolo, T. A., Esprit, E., Wauters, T., and Vanden Berghe, G. (2017). A two-dimensional heuristic decomposition approach to a three-dimensional multiple container loading problem. *European Journal of Operational Research*, 257(2):526–538.
- [Van Belle et al., 2012] Van Belle, J., Valckenaers, P., and Cattrysse, D. (2012). Cross-docking: State of the art. *Omega*, 40(6):827–846.
- [Wang et al., 2017] Wang, X., Poikonen, S., and Golden, B. (2017). The vehicle routing problem with drones: Several worst-case results. *Optimization Letters*, 11(4):679–697.
- [Wang and Sheu, 2019] Wang, Z. and Sheu, J.-B. (2019). Vehicle routing problem with drones. *Transportation research part B: methodological*, 122:350–364.
- [Wohlsen, 2014] Wohlsen, M. (2014). The next big thing you missed: Amazon’s delivery drones could work—they just need trucks. *Wired. com*. [<http://www.wired.com/2014/06/the-nextbig-thing-you-missed-delivery-drones-launched-from-trucks-are-the-future-ofshipping/>]. Accessed 10/15/2014.]

- [Ye et al., 2018] Ye, Y., Li, J., Li, K., and Fu, H. (2018). Cross-docking truck scheduling with product unloading/loading constraints based on an improved particle swarm optimisation algorithm. *International Journal of Production Research*, 56(16):5365–5385.
- [Yu and Egbelu, 2008] Yu, W. and Egbelu, P. J. (2008). Scheduling of inbound and outbound trucks in cross docking systems with temporary storage. *European Journal of Operational Research*, 184(1):377–396.
- [Yurek and Ozmutlu, 2018] Yurek, E. E. and Ozmutlu, H. C. (2018). A decomposition-based iterative optimization algorithm for traveling salesman problem with drone. *Transportation Research Part C: Emerging Technologies*, 91:249–262.
- [Zufferey et al., 2016] Zufferey, N., Cho, B. Y., and Glardon, R. (2016). Dynamic multi-trip vehicle routing with unusual time-windows for the pick-up of blood samples and delivery of medical material. In *Proceedings of the 5th International Conference on Operations Research and Enterprise Systems, ICORES 2016, Rome, Italy, February 23–25*.

Appendix A

Appendix for chapter 5

A.1 Fastening the insertion heuristic

[Masson et al., 2013] proposed a *fast feasibility check* (FFC) procedure for interdependent routes. It extends the *forward slack time* procedure introduced by [Savelsbergh, 1992]. In the VRPTR case, an additional modeling effort is required to take into account that all WRs depend on each other (as the same worker can be transported through multiple WRs).

A.1.1 Modeling: aggregated nodes

For the vehicle routes, only the entry and exit points of a WR are relevant (as all intermediate nodes are not visited by the vehicles). Accordingly, we introduce two aggregated nodes for a WR: one for the drop-off (at the beginning of the WR) and one for the pick-up (at the end of the WR). These nodes are visited by the vehicles and gather consolidated

information about the WR (total duration and resulting time window).

Let $\Omega = \bigcup_{w \in W} \Omega_w$ be the set of all WRs, where Ω_w represents the ordered set of WRs performed by worker $w \in W$ in her/his schedule ($\kappa_{i+1} \in \Omega_w$ is directly performed after WR κ_i in the schedule of worker w , $\forall i < |\Omega_w|$). $w(\kappa)$ is the worker associated with κ , and $i(\kappa)$ is the position of WR κ in the worker's planning. Furthermore, $\rho(\kappa)$ (resp. $\sigma(\kappa)$) represents the predecessor (resp. successor) of κ in the corresponding worker planning. $\rho(\kappa)$ (resp. $\sigma(\kappa)$) is \emptyset if κ is the first (resp. the last) WR done. $[e_\kappa, l_\kappa]$ and p_κ denotes the time window of κ (to serve all jobs of κ on time) and the total processing time (i.e., all processing times, walking times, and waiting times along κ).

\mathcal{D} (resp. \mathcal{P}) is the set of all drop-off (resp. pick-up) points. $D_\kappa \in \mathcal{D}$ (resp. $P_\kappa \in \mathcal{P}$) is the drop-off (resp. pick-up) point of κ . $O_W \subset \mathcal{P}$ (resp. $O'_W \subset \mathcal{D}$) represents the set of worker pick-up (resp. drop-off) points at the depot. Moreover, O_K (resp. O'_K) denotes the set of the first (resp. the last) nodes visited by the vehicles. Finally, $\mathcal{M} = \mathcal{P} \cup \mathcal{D} \cup O_K \cup O'_K$ denotes the set of all aggregated nodes for a given solution.

A transportation request arises between the end of a WR κ and the beginning of the next WR $\sigma(\kappa)$, denoted by $(P_\kappa, D_{\sigma(\kappa)})$. Furthermore, transportation requests are required between the pick-up at the depot and the drop-off at the beginning of the first WR as well as between the pick-up in the last WR (of any worker's planning) and the final drop-off at the depot: (P_{O_w}, D_{κ_1}) (resp. $(P_{\kappa_{|\omega_w|}}, D_{O'_w})$) for the transportation between the depot (resp. last WR $\kappa_{|\omega_w|}$) and the first WR κ_1 (resp. the depot).

For each $v \in \mathcal{M}$, $k(v)$ is the route that visits v , and $i(v)$ is the position of v in the route. $\rho(v)$ (resp. $\sigma(v)$) denotes the predecessor (resp. successor) of v in the route. Finally, for each $v \in \mathcal{D} \cup \mathcal{P}$, $\kappa(v)$ denotes the WR that contains v . Workers' pick-up and drop-off times are set to be null. Each $v \in \mathcal{M}$ can be characterized by an associated time window $[e_v, l_v]$, which corresponds to the time a car must drop off a worker to have an on-time

arrival for the jobs composing $\kappa(v)$. For each $v \in \mathcal{D}$, we have $e_v = e_{\kappa(v)}$ and $l_v = l_{\kappa(v)}$, whereas for each $v \in \mathcal{P}$, we have $l_v = \infty$, and e_v depends on the drop-off time at $D(v)$.

A.1.2 Vehicle constraints

A vehicle route is an ordered set of aggregated nodes that must satisfy the following constraints, where $D(v)_{v \in \mathcal{P} \cup O_W}$ (resp. $P(v)_{v \in \mathcal{D} \cup O'_W}$) designates the drop-off (resp. pick-up) in the pick-up and drop-off couple. More precisely, $D(v) = P_{\rho(\kappa(v))}$, $\forall v \in \mathcal{P}$, and $D(v) = P_{0,w(\kappa(v))}$, $\forall v \in O_W$. $\mathbb{1}_{i=P} = 1$ (resp. $\mathbb{1}_{i=D} = 1$) if i is a pick-up (resp. drop-off) aggregated node, and 0 otherwise. Constraints (A.1) ensure that the nodes of a pick-up and drop-off couple are managed by the same vehicle, and the pick-up must occur before the drop-off. Constraints (A.2) ensure that a vehicle cannot move without its associated driver by scheduling the driver's pick-up directly after her/his drop-off in the vehicle route. Constraints (A.3) ensure that the vehicle capacity q is never exceeded. A set of routes is feasible if the above constraints are satisfied and if it fulfills the temporal constraints that are detailed in the next subsection.

$$k(v) = k(D(v)) \text{ and } i(v) < i(D(v)), \quad \forall v \in O_W \cup \mathcal{P} \quad (\text{A.1})$$

$$k(D_{\kappa(v)}) = k(P_{\kappa(v)}) \text{ and } i(P_{\kappa(v)}) = i(D_{\kappa(v)}) + 1, \quad \forall v \in \mathcal{D} / w(\kappa(v)) \text{ is a driver} \quad (\text{A.2})$$

$$\sum_{v \in k} (\mathbb{1}_{v=P} - \mathbb{1}_{v=D}) \leq q, \quad \forall k \in K \quad (\text{A.3})$$

A.1.3 Temporal constraints

A solution to the VRPTW is feasible if and only if each of its routes satisfies temporal feasibility. The temporal constraints are modeled using a *Simple Temporal Problem* (as described by [Dechter et al., 1991]), for which efficient algorithms and representations exist in the literature. Temporal constraints are expressed as follows in Equations (A.4)–

(A.6), where h_v represents the service time at the aggregated node $v \in \mathcal{M}$:

$$h_{\sigma(v)} \geq h_v + \tau_{v,\sigma(v)}^d, \quad \forall v \in \mathcal{M} \setminus O'_K \quad (\text{A.4})$$

$$h_{P_\kappa} \geq \max\{h_{D_\kappa}, e_{D_\kappa}\} + p_\kappa, \quad \forall \kappa \in \Omega \quad (\text{A.5})$$

$$h_v \leq l_v, \quad \forall v \in \mathcal{M} \quad (\text{A.6})$$

Equations (A.4) set the temporal constraints in a route, for which the arrival time at a node depends on the departure time at the previous node. Equations (A.5) specify the time at which a worker is available to be picked up after completing a WR. The time at which a worker starts working on a WR depends on both the drop-off time h_{D_κ} and on the time window e_{D_κ} of the WR. Finally, Equations (A.6) state that the service time cannot start after the end of the corresponding time window.

This set of equations can be modeled with a precedence graph, called G^p , where constraints of type $h_u - h_v \geq a_{uv}$ (a_{uv} is a real number) represent an arc from u to v with a cost of a_{uv} . Node o is introduced to represent the beginning of the planning horizon, and, for every drop-off point $D \in \mathcal{D}$, a virtual node $D^{(dup)}$ is introduced to get rid of the max function in Equations (A.5). $\mathcal{D}^{(dup)}$ is the set of duplicated nodes. Equations (A.4) to (A.6) can therefore be rewritten as follows:

$$h_{\sigma(v)} - h_v \geq \tau_{v,\sigma(v)}^d, \quad \forall v \in \mathcal{M} \setminus O'_K \quad (\text{A.7})$$

$$h_{P_\kappa} - h_{D_\kappa^{(dup)}} \geq p_\kappa, \quad \forall \kappa \in \Omega \quad (\text{A.8})$$

$$h_{D_\kappa^{(dup)}} - h_{D_\kappa} \geq 0, \quad \forall \kappa \in \Omega \quad (\text{A.9})$$

$$h_{D_\kappa^{(dup)}} - h_o \geq e_{D_\kappa}, \quad \forall \kappa \in \Omega \quad (\text{A.10})$$

$$h_o - h_v \geq -l_v, \quad \forall v \in \mathcal{M} \quad (\text{A.11})$$

$$h_v \geq 0, \quad \forall \kappa \in \Omega \quad (\text{A.12})$$

$$h_o = 0 \quad (\text{A.13})$$

Checking the feasibility of the VRPTR set of temporal constraints is equivalent to show-

ing that there is no cycle of negative length in the precedence graph. This can be done using the so-called BFCT algorithm, which has a complexity of $\mathcal{O}(|\mathcal{M}| \times |A'|)$ [Cherkassky et al., 2009]. For any solution satisfying the temporal constraints, the precedence graph is a direct acyclic graph.

Figure A.1 presents the precedence graph associated with Figure 5.1 using the above-introduced notation. In Figure 5.1, the solution with carpooling and walking contains three WRs, which can be denoted as $\Omega = \{\kappa_1 = \{j_1\}, \kappa_2 = \{j_2, j_3\}, \kappa_3 = \{j_4\}\}$. It involves six pick-up and drop-off couples denoted as $(P_{O_{w_1}}, D_{\kappa_1})$, $(P_{\kappa_1}, D_{0'_{w_1}})$, $(P_{O_{w_2}}, D_{\kappa_2})$, $(P_{\kappa_2}, D_{0'_{w_2}})$, $(P_{O_{w_3}}, D_{\kappa_3})$, and $(P_{\kappa_3}, D_{0'_{w_3}})$.

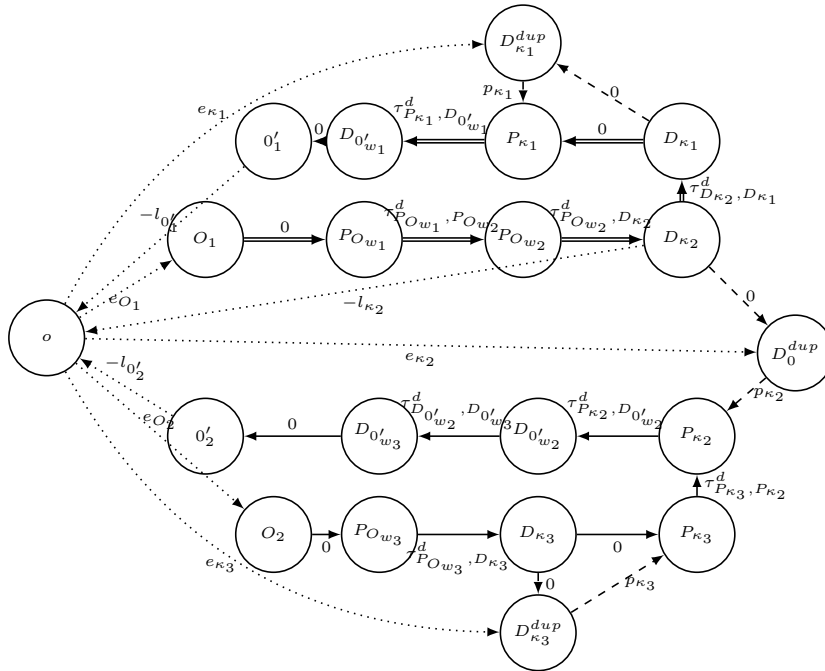


Figure A.1: Precedence graph representing the VRPTR solution of Figure 5.1. Dotted arcs represent time window constraints (for the sake of clarity, not all time window constraints are drawn), dashed arcs represent precedence constraints due to WRs, and both double and normal arcs represent precedence constraints due to the routes. The order of the nodes in the route must satisfy the constraints in Equations (A.1)–(A.3).

The FFC procedure pre-computes, for each aggregated node $v \in \mathcal{M}$, the earliest service time (h_v), the latest departure time (λ_v), and the matrix of waiting times between all aggregated nodes ($(\Phi_{uv})_{u,v \in \mathcal{M}}$). As the precedence graph G^p represents a feasible solution, it does not contain any cycle of positive weight; therefore, the longest path is the shortest

path in $-G^p$, where the arcs of $-G^p$ have the opposite weight of the arcs in G^p . The precedence graph is a direct acyclic graph, where the shortest paths can be computed in linear time. $(\Phi_{uv})_{u,v \in \mathcal{M}}$ is computed as the shortest paths in the precedence graph, where the arcs are weighted with the waiting time in the solution at the terminal node of the arc (i.e., the waiting time at node v is equal to $\max\{0, e_v - h_v\}$), and it can be computed in $\mathcal{O}(n^2)$. All these shortest paths are computed once, and then the $\mathcal{O}(n^4)$ insertion positions are tested in constant time.

A.2 Detailed results for all instances

Tables A.1 to A.4 detail the results found by VNS for all VRPTR configurations over the 120 generated instances. The table contents correspond to those of Table 5.3, a description of which can be found in Section 5.5.5.

Table A.1: Detailed results for instances involving 20 jobs.

Instance	VRP ($P^{ \mathcal{W}^* }_{no\ walk}$)				Park-and-Loop ($P^{ \mathcal{W}^* }_{walk}$) d	Carpooling					
	\mathcal{W}^*	d^*	Idle Time			$(P^{ \mathcal{W}^* -1}_{no\ walk})$		$(P^{ \mathcal{W}^* -1}_{walk})$		$(P^{ \mathcal{W}^* -2}_{walk})$	
			Route	Depot		d	J_{out}	d	J_{out}	d	J_{out}
20_A.1	2	37.8	0.0%	26.1%	32.4	47.2	0	40.3	0	0	6
20_A.2	2	46.2	0.0%	24.1%	41.6	59.0	0	50.0	0	0	6
20_A.3	2	43.2	0.0%	29.6%	39.4	48.8	0	45.6	0	0	6
20_A.4	2	42.0	0.0%	26.3%	39.1	48.7	0	46.6	0	0	6
20_A.5	2	41.5	0.0%	25.2%	35.3	48.4	0	49.6	0	0	7
20_A.6	2	46.0	0.0%	23.0%	43.8	64.9	0	69.0	0	0	8
20_A.7	2	44.3	0.0%	24.0%	40.1	66.1	0	66.1	0	0	7
20_A.8	2	42.3	0.0%	19.7%	39.1	55.4	0	55.4	0	0	7
20_A.9	2	33.6	0.0%	25.3%	29.9	40.0	0	38.0	0	0	6
20_A.10	2	43.8	0.0%	24.7%	41.2	49.0	0	48.2	0	0	6
20_H.1	2	49.0	16.0%	7.5%	41.4	-	1	-	1	0	7
20_H.2	2	53.6	5.0%	17.4%	52.2	-	1	-	1	0	7
20_H.3	2	52.1	14.8%	12.7%	51.4	77.8	0	66.7	0	0	8
20_H.4	2	51.1	13.5%	10.7%	49.4	-	1	-	1	0	9
20_H.5	2	52.6	8.4%	14.2%	47.6	80.1	0	80.1	0	0	7
20_H.6	2	51.1	16.4%	5.4%	50.3	-	1	-	1	0	9
20_H.7	2	59.4	11.4%	9.0%	55.3	-	2	-	2	0	8
20_H.8	2	51.7	3.5%	13.9%	46.6	-	2	-	1	0	8
20_H.9	2	44.2	13.0%	9.8%	39.8	-	1	-	1	0	7
20_H.10	2	54.6	14.6%	7.5%	52.0	-	1	-	1	0	7
20_Q.1	2	61.2	15.0%	5.5%	60.1	-	3	-	2	0	9
20_Q.2	2	59.4	6.1%	14.9%	58.0	-	3	-	3	0	7
20_Q.3	2	63.9	23.3%	1.3%	63.9	-	2	-	2	0	9
20_Q.4	2	68.6	15.9%	4.1%	67.0	-	3	-	3	0	11
20_Q.5	2	74.9	14.4%	2.9%	71.0	-	2	-	2	0	10
20_Q.6	3	72.1	40.6%	3.9%	70.5	-	1	-	1	-	1
20_Q.7	3	65.4	38.0%	8.0%	60.8	70.6	0	64.9	0	-	1
20_Q.8	2	66.4	3.5%	10.5%	63.1	-	3	-	2	-	9
20_Q.9	3	54.6	40.5%	6.4%	50.7	66.0	0	54.8	0	-	1
20_Q.10	3	60.7	37.5%	9.6%	59.8	61.6	0	60.3	0	-	0

Table A.2: Detailed results for instances involving 30 jobs.

Instance	VRP ($P_{no\ walk}^{ W^* }$)				Park-and-Loop ($P_{walk}^{ W^* }$) d	Carpooling					
	$ W^* $	d^*	Idle Time			$(P_{no\ walk}^{ W^* -1})$		$(P_{walk}^{ W^* -1})$		$(P_{walk}^{ W^* -2})$	
			Route	Depot		d	J^{out}	d	J^{out}	d	J^{out}
30_A.1	3	48.8	0.0%	25.6%	41.3	49.5	0	43.0	0	-	2
30_A.2	3	53.0	0.0%	26.5%	46.6	53.4	0	46.8	0	-	1
30_A.3	3	57.1	0.0%	27.4%	51.0	54.3	0	54.3	0	-	2
30_A.4	3	52.8	0.0%	30.5%	46.3	49.8	0	45.8	0	-	1
30_A.5	3	55.9	0.0%	24.9%	48.4	54.3	0	44.8	0	-	2
30_A.6	3	57.3	0.0%	23.8%	51.7	54.0	0	49.5	0	-	2
30_A.7	3	54.7	0.0%	29.4%	45.5	53.3	0	46.7	0	-	1
30_A.8	3	50.6	0.0%	22.5%	45.4	52.9	0	44.6	0	-	3
30_A.9	3	49.5	0.0%	26.3%	41.9	48.1	0	43.0	0	-	1
30_A.10	3	53.8	0.0%	27.6%	49.5	54.5	0	49.9	0	-	1
30_H.1	3	62.5	20.2%	3.3%	57.6	-	2	-	2	-	-
30_H.2	3	61.5	4.4%	20.8%	57.2	78.3	0	65.3	0	-	1
30_H.3	3	74.3	14.0%	10.8%	67.6	84.5	0	75.5	0	-	4
30_H.4	3	68.2	19.8%	8.3%	63.7	88.0	0	71.2	0	-	4
30_H.5	3	68.0	5.2%	17.8%	61.1	79.4	0	67.9	0	-	3
30_H.6	3	71.1	9.3%	12.3%	65.0	80.6	0	70.1	0	-	3
30_H.7	3	66.5	12.1%	15.4%	58.1	72.6	0	64.7	0	-	3
30_H.8	3	59.2	8.1%	13.1%	52.7	76.3	0	65.0	0	-	4
30_H.9	3	56.5	10.0%	15.2%	50.7	64.3	0	59.5	0	-	3
30_H.10	3	65.2	14.9%	10.8%	61.3	75.1	0	67.8	0	-	3
30_Q.1	3	88.6	18.5%	0.7%	87.9	-	3	-	3	-	-
30_Q.2	3	73.8	8.3%	14.9%	68.7	-	1	-	1	-	-
30_Q.3	3	85.4	18.1%	4.9%	81.6	115.4	0	115.4	0	-	7
30_Q.4	3	89.9	18.9%	5.8%	86.3	-	1	-	1	-	-
30_Q.5	3	94.9	14.1%	4.6%	90.6	-	2	-	1	-	-
30_Q.6	4	102.9	35.5%	2.0%	97.7	-	1	-	1	-	-
30_Q.7	3	94.3	19.8%	3.3%	94.3	-	3	-	3	-	-
30_Q.8	3	78.6	13.1%	5.0%	77.7	110.4	0	104.7	0	-	4
30_Q.9	4	74.1	32.2%	9.6%	68.6	77.1	0	67.9	0	78.5	0
30_Q.10	4	77.5	37.1%	5.8%	74.3	78.6	0	74.8	0	89.1	0

Table A.3: Detailed results for instances involving 40 jobs.

Instance	VRP ($P_{no\ walk}^{ W^* }$)				Park-and-Loop ($P_{walk}^{ W^* }$) d	Carpooling					
	$ W^* $	d^*	Idle Time			$(P_{no\ walk}^{ W^* -1})$		$(P_{walk}^{ W^* -1})$		$(P_{walk}^{ W^* -2})$	
			Route	Depot		d	J^{out}	d	J^{out}	d	J^{out}
40_A.1	3	60.0	0.0%	2.4%	59.2	-	4	-	4	-	-
40_A.2	3	59.9	0.0%	3.2%	58.5	-	4	-	4	-	-
40_A.3	3	63.8	0.0%	2.6%	63.2	-	3	-	3	-	-
40_A.4	3	56.9	0.0%	10.0%	50.4	-	2	-	1	-	-
40_A.5	4	63.3	0.0%	24.9%	52.8	61.0	0	53.3	0	60.3	0
40_A.6	3	62.7	0.0%	3.5%	62.7	-	4	-	3	-	-
40_A.7	3	58.6	0.0%	7.0%	57.8	-	2	-	2	-	-
40_A.8	4	66.2	0.0%	21.9%	56.4	65.0	0	53.8	0	68	0
40_A.9	3	57.5	0.0%	3.2%	56.5	-	3	-	3	-	-
40_A.10	3	59.3	0.0%	4.9%	57.7	-	3	-	3	-	-
40_H.1	4	75.5	19.6%	5.3%	68.4	90.3	0	81.1	0	-	3
40_H.2	4	73.1	0.0%	25.8%	63.9	75.1	0	62.1	0	94.6	0
40_H.3	4	82.6	14.1%	10.6%	71.5	85.3	0	76.0	0	-	1
40_H.4	3	80.2	3.1%	3.2%	78.1	-	4	-	4	-	-
40_H.5	4	75.4	8.9%	14.5%	65.9	81.0	0	72.3	0	98.7	0
40_H.6	4	73.1	13.7%	12.7%	63.6	75.4	0	63.5	0	87.9	0
40_H.7	3	76.3	2.8%	1.3%	76.3	-	4	-	4	-	-
40_H.8	4	78.3	5.2%	15.3%	69.1	85.6	0	75.2	0	-	3
40_H.9	4	71.5	11.3%	14.4%	57.9	72.0	0	64.1	0	84.1	0
40_H.10	3	79.8	0.0%	1.6%	79.8	-	5	-	5	-	-
40_Q.1	4	96.5	21.1%	1.3%	90.8	109.0	1	108.8	1	-	-
40_Q.2	4	85.6	4.0%	20.3%	80.2	98.2	1	88.6	0	101.7	3
40_Q.3	4	101.7	15.1%	7.3%	97.9	116.4	0	106.3	0	114.1	3
40_Q.4	3	117.4	0.0%	0.4%	117.4	103.6	6	98.5	5	-	-
40_Q.5	4	104.8	15.7%	4.3%	96.5	123.0	0	106.3	0	114.9	3
40_Q.6	5	100.3	33.7%	4.9%	92.0	103.9	0	94.1	0	103	0
40_Q.7	4	91.0	23.2%	3.2%	84.4	108.8	0	97.9	0	93.8	3
40_Q.8	4	98.9	14.6%	3.3%	96.9	114.8	0	104.4	0	71.9	3
40_Q.9	4	92.8	16.8%	6.4%	87.1	90.3	2	87.7	1	-	-
40_Q.10	4	97.1	19.5%	4.7%	88.6	106.5	0	98.6	0	101.3	3

Table A.4: Detailed results for instances involving 50 jobs.

Instance	VRP ($P^{W^* }_{no\ walk}$)				Park-and-Loop ($P^{W^* }_{walk}$) d	Carpooling					
	$ W^* $	d^*	Idle Time			$(P^{W^* -1})_{no\ walk}$		$(P^{W^* -1})_{walk}$		$(P^{W^* -2})_{walk}$	
			Route	Depot		d	$ J^{out} $	d	$ J^{out} $	d	$ J^{out} $
50_A_1	4	65.4	0.0%	10.4%	60.5	74.4	0	74.4	0	-	-
50_A_2	4	69.0	0.0%	11.7%	62.5	87.9	0	72.1	0	-	4
50_A_3	4	71.4	0.0%	9.7%	65.9	-	1	-	1	-	-
50_A_4	4	68.1	0.0%	13.3%	58.2	85.8	0	68.0	0	-	4
50_A_5	4	68.3	0.0%	6.2%	64.2	-	1	-	1	-	-
50_A_6	4	71.8	0.0%	11.7%	66.1	78.7	0	70.9	0	-	4
50_A_7	4	71.9	0.0%	12.6%	63.7	78.5	0	74.7	0	-	3
50_A_8	4	71.7	0.0%	5.7%	70.9	-	2	-	2	-	-
50_A_9	4	67.4	0.0%	11.6%	58.2	76.4	0	70.4	0	-	4
50_A_10	4	68.3	0.0%	12.4%	60.8	76.3	0	66.1	0	-	5
50_H_1	4	92.5	6.4%	0.8%	92.5	-	3	-	3	-	-
50_H_2	4	81.1	0.0%	10.3%	75.4	-	2	-	1	-	-
50_H_3	4	89.4	2.0%	5.5%	84.8	-	2	-	1	-	-
50_H_4	4	86.9	2.4%	8.7%	82.3	106.8	0	106.8	0	-	5
50_H_5	4	88.4	0.0%	3.7%	87.9	-	3	-	3	-	-
50_H_6	4	90.9	8.2%	1.2%	88.4	-	2	-	2	-	-
50_H_7	4	88.5	4.5%	6.1%	82.6	-	1	97.5	0	-	6
50_H_8	4	88.8	1.7%	2.0%	85.9	-	4	-	4	-	-
50_H_9	4	82.4	3.2%	6.7%	74.2	-	1	-	1	-	-
50_H_10	4	89.3	7.8%	2.1%	84.8	-	2	-	2	-	-
50_Q_1	5	104.1	19.6%	5.0%	95.5	117.5	0	107.4	0	109.7	1
50_Q_2	4	110.2	1.3%	5.5%	110.2	-	4	-	3	-	-
50_Q_3	4	124.9	1.0%	2.3%	124.9	-	5	-	4	-	-
50_Q_4	4	121.7	3.2%	3.7%	121.7	-	4	-	2	-	-
50_Q_5	5	113.2	17.9%	2.7%	104.4	130.7	0	117.0	0	116.4	2
50_Q_6	5	110.7	23.1%	2.5%	106.4	-	1	-	1	-	-
50_Q_7	5	104.3	22.1%	4.9%	96.1	111.0	0	101.4	0	106.5	2
50_Q_8	5	113.5	18.7%	1.9%	111.5	132.0	0	117.3	0	147.1	1
50_Q_9	4	113.9	3.3%	2.8%	111.6	-	3	-	3	-	-
50_Q_10	5	107.6	21.3%	4.9%	97.5	112.3	0	103.3	0	126.2	1