Problèmes de tournées de véhicules avec plusieurs produits et applications à la livraison de produits frais en circuits courts et locaux

Wenjuan Gu

N° d'ordre: | 3 | 8 | 2 |

CENTRALE LILLE

THESE

Présentée en vue
d'obtenir le grade de

DOCTEUR

En

Spécialité : Informatique

Par

Wenjuan GU

DOCTORAT DELIVRE PAR CENTRALE LILLE

Titre de la thèse :

Problèmes de tournées de véhicules avec plusieurs produits et applications à la livraison de produits frais en circuits courts et locaux

Multiple commodities routing problems with applications in the local fresh food supply chain

---

Soutenue le 4 novembre 2019 devant le jury d'examen :

Président	M. Fabien LEHUÉDÉ	Professeur, IMT Atlantique, France
Rapporteur	Mme Yasemin ARDA	Professeur, HEC Liège, Belgique
Rapporteur	M. Kenneth SÖRENSEN	Professeur, University of Antwerpen, Belgique
Examinateur	Mme Luce BROTCORNE	Directrice de Recherche, INRIA Lille-Nord Europe, France
Examinateur	M. Fabien LEHUÉDÉ	Professeur, IMT Atlantique, France
Examinateur	M. Daniele VIGO	Professeur, Università di Bologna, Italie
Directeur de thèse	M. Frédéric SEMET	Professeur, Centrale Lille, France
Co-encadrant	M. Diego CATTARUZZA	Maitre de Conférences, Centrale Lille, France
Co-encadrant	M. Maxime OGIER	Maitre de Conférences, Centrale Lille, France

Thèse préparée au Centre de Recherche en Informatique Signal et Automatique de Lille
CRIStAL, UMR CNRS 9189 - Centrale Lille
Ecole Doctorale SPI 072

Acknowledgements

This thesis has become reality with the kind support and help of many people. At this moment, I would like to express my sincere gratitude to those who have given me support, help and encouragement during my work.

First of all, I would like to express my deepest thanks to my supervisors: Diego and Maxime and Frédéric. Thank you for your knowledge, your patience and your serious way to do the research. I have learnt a lot from you and I really appreciate working with you. Thank you for teaching me in accordance with my aptitude, and providing me with very useful and important guidance for my work. Without your wise guidance, the conducted research would not have been possible. I have also obtained a lot of experiences in research, these will be invaluable on both my future study as well as my career, and encourage me to grow as an independent thinker.

Besides, I would like to thank Prof. Yasemin ARDA and Prof. Kenneth SÖRENSEN for having accepted the invitation of my committee and reviewed the dissertation. I would also like to thank Prof. Luce BROTCORNE, Prof. Fabien LEHUÉDÉ and Prof. Daniele VIGO for accepting to be members of the jury. Thank all the jury members for their insightful comments and nice questions.

In particular, I would like to gratefully thank Prof. Claudia ARCHETTI and Prof. Maria Grazia SPERANZA who have welcomed me for two weeks in Brescia. Thank them for the precious and inspiring collaborations.

I also wish to thank all the members in the INOCS team. Thank you for many beautiful moments and interesting discussions. I felt really lucky and happy being with you here in Lille. Thanks to all my friends, so many that I could not cite their names here, for your warm care and friendship, for your help, and for the time we spent together, which bring me a lot of happiness in the three years life in France. I also want to express my great thanks to the staff in Inria and Centrale Lille for their kind help.

ACKNOWLEDGEMENTS

Finally, biggest thanks to my beloved family for their unwavering support. My parents, brother and sister, who give me all their unselfish love, and always stand behind me. I greatly appreciate my husband Wei HU who constantly believes in me and accompanies me. He always gives me unconditional love, meticulous caring, and continuous support. Special thank goes to my grandmother for her precious encouragements to help me face the life in an unfamiliar country, overcome the difficulties in my life.

My Ph.D. study was supported and funded by CSC (China Scholarship Council). Therefore, I would like to thank all the people who have contributed to select candidates for this interesting project and given me this precious opportunity.

Villeneuve d'Ascq, France *Wenjuan GU*
November, 2019

Contents

List of Figures

List of Tables

LIST OF TABLES

Chapter 1

Introduction

The production and delivery of fresh food products have undergone important changes in Europe since the 1950s, especially through the modernization of the tools and processes in order to meet the customer demand with low production costs. Multinational companies have played a major role as intermediaries between farmers and consumers (Rucabado-Palomar & Cuéllar-Padilla, 2018). Nowadays, one of the major problems faced by farmers is the shortfall of their incomes: over the last decades, they have been encouraged to produce more, while their unit selling price was decreasing. However, in many regions there coexist (1) supplies with medium-sized farms where various products of high quality (freshness, few pesticides) are cultivated and (2) customers with a strong desire for product quality and traceability (King *et al.*, 2015). Hence, the idea has emerged to locally connect suppliers and customers (Berti & Mulligan, 2016), through a short (and/or) local food supply chain. The main purpose of this kind of supply chain is to capture more end-use value for the farmers.

Short food supply chains are officially defined by the French Ministry of Agriculture as a marketing mode for agricultural products either through direct sales from producers to consumers, either through indirect sales with only one intermediary between producers and consumers. Local food supply chains may involve several intermediaries, but all the actors have to be located on a limited area (e.g. considering geographical or political restrictions). The maximum distance between actors usually corresponds to around 80 km (Blanquart *et al.*, 2010). Conventional supply chains involve numerous manual operations on the products and significant storage and transportation times. It is then crucial, in long supply chains, to avoid food

waste and economic loss due to fresh products perishability. Indeed, according to a report from the Food and Agriculture Organization of the United Nations (Gustavsson *et al.*, 2012), around one-third of the food for human consumption is lost or wasted around the world. In short and local food supply chains, actors are located in a restricted area and there are few intermediaries. This naturally provides better guarantee on the traceability and freshness of products. Indeed, the time between collection and delivery is typically at most 24 hours, and handling activities are limited since the number of intermediaries is restricted.

Short and local food supply chains involve few intermediaries. Hence farmers have to take charge of a large part of their product marketing and distribution, which is not their core business. It is feasible when farmers directly sell their products to customers since the volumes are usually low. For indirect sales (canteens, restaurants or supermarkets), volumes are more important, so the supply chain has to be designed appropriately to organize product flows and minimize transportation costs to be competitive with conventional food supply chains.

Short and local food supply chains usually rely on a set of distribution centers. As pointed out by Berti & Mulligan (2016), distribution centers (also named food hubs) are the most commonly used infrastructures to meet the growing demand for local products. Farmers usually supply these distribution centers by performing direct trips since the volumes are large. The distribution centers are then in charge of consolidation and delivery of the products to customers. Besides, it is usually assumed that a single decision-maker manages all the distribution centers, and coordinates the transportation planning for both collection and delivery operations. This decision-maker can be an association of farmers or a local political authority. The distribution centers are considered as the only intermediary in the supply chain.

The objective of this thesis is to design and implement efficient solution methods for routing problems that arise in fresh and local food supply chain. In particular, a key feature of these routing problems is to explicitly consider multiple commodities since: (1) all farmers do not produce the same commodities, and (2) the commodities may be delivered to customers by different vehicles.

In this thesis, we aim to study a complex distribution problem in a two-echelon supply chain where three sets of stakeholders are involved: suppliers, distribution centers and customers. Multiple commodities are collected from the suppliers and delivered to the customers through distribution centers for consolidation purposes.

Each supplier has a given available quantity for each commodity (possibly 0), and each customer has a demand for each commodity (possibly 0). The commodities are collected from suppliers and delivered to distribution centers through direct trips, and distributed from the distribution centers to customers with a fleet of vehicles performing routes. Direct deliveries from suppliers to customers are not considered. We assume that commodities are compatible, that is any vehicle can transport any subset of commodities as long as its capacity is not exceeded. Multiple visits to a customer are allowed to reduce transportation costs. However, for the sake of customers convenience, a single commodity has to be delivered at once.

In the following, we define the structure of the thesis and the topics covered in it.

Thesis structure

The thesis contains six chapters including this one. In the following, we present a brief description of the content of each chapter.

Chapter 2: Vehicle routing with multiple commodities: a survey

In this chapter, we survey vehicle routing problems that explicitly consider multiple commodities. Classically, routing problems implicitly consider multiple commodities, for example by aggregating the requested commodities based on volume or weight to form a single demand, or by decomposing the problem for each commodity (e.g. if they are delivered from different depots or if the vehicles are dedicated to a single commodity). In this survey, for each problem, we determine what motivates the explicit consideration of multiple commodities, how multiple commodities are modeled in the formulations and solving methods, and what are the main applications.

Chapter 3: Adaptive Large Neighborhood Search for the Commodity constrained Split Delivery VRP

In this chapter, we develop a heuristic based on the Adaptive Large Neighborhood Search (ALNS) to address the delivery of customers from a single distribution center,

considering multiple commodities. More precisely, the problem studied is named the Commodity constrained Split Delivery Vehicle Routing Problem (C-SDVRP). This problem arises when customers require multiple commodities and accept that they are delivered separately. All the commodities can be mixed in a vehicle as long as the vehicle capacity is satisfied. Multiple visits to a customer are allowed, but a given commodity must be delivered in a single delivery.

We propose a heuristic based on the Adaptive Large Neighborhood Search (ALNS) to solve the C-SDVRP, with the objective of efficiently tackling medium and large sized instances. We take into account the distinctive features of the C-SDVRP and adapt several local search moves to improve a solution. Moreover, a Mathematical Programming based Operator (MPO) that reassigns commodities to routes is used to improve a new global best solution. Computational experiments have been performed on benchmark instances from the literature. The results assess the efficiency of the algorithm, which can provide a large number of new best-known solutions in short computational times.

This chapter has been published as:

Gu, W., Cattaruzza, D., Ogier, M., and Semet, F. (2019). Adaptive large neighborhood search for the commodity constrained split delivery VRP. Computers & Operations Research, 112.

Chapter 4: A decomposition approach to a multi-commodity two-echelon distribution problem

In this chapter, we propose sequential solving approaches for the whole problem, namely the Multi-Commodity two-echelon Distribution Problem (MC2DP). We propose a decomposition of the MC2DP into two subproblems, namely a collection problem where farmers transport their own commodities to distribution centers by performing round trips, and a delivery problem where the distribution centers deliver customers with a fleet of homogeneous vehicles performing routes. The collection problem is modeled as a mixed integer linear program. The delivery problem is the multi-depot case of the C-SDVRP. Hence, we extend the ALNS developed for the C-SDVRP to the multi-depot case for the delivery operations. Then, we address sequential solving approaches for the whole MC2DP. In the MC2DP, collection decisions (which quantity of each commodity is delivered to which distribution center) impact delivery operations. Thus, collection and delivery must be determined

jointly. We present two sequential solution approaches based on the solving, in different order, of the collection and the delivery subproblems. In both cases, the solution of the first subproblem determines the quantity of each commodity at each distribution center. The second subproblem takes this information as an input. We also propose different strategies to guide the solution of the first subproblem in order to take into account the impact of its solution on the second subproblem. The proposed sequential approaches are evaluated and compared both on randomly generated instances and on a case study related to a short and local fresh food supply chain. The results show the impact of problem characteristics on solution strategies.

This chapter has been done in cooperation with Professor Claudia Archetti from ESSEC Business School in Paris and Professor M.Grazia Speranza from University of Brescia. This chapter has been submitted to Omega-The International Journal of Management Science.

Chapter 5: An integrated approach to a multi-commodity two-echelon distribution problem

In this chapter, we develop an integrated solution approach for the MC2DP. Based on the sequential approaches presented in Chapter 4, we improved the ALNS. We add new operators able to modify both collection and delivery operations, by moving some quantities from one distribution center to another. Some of these operators are based on mathematical programming in order to modify the current solution of the collection subproblem. This integrated approach is evaluated and compared to the best solutions obtained with the sequential approaches.

This chapter has been done in cooperation with Professor Claudia Archetti from ESSEC Business School in Paris and Professor M.Grazia Speranza from University of Brescia.

Chapter 6: Conclusions and perspectives

This chapter concludes the thesis and provides some perspectives. We first conclude the main contributions of the thesis. Then, we provide some directions for further research in terms of methodology and problems to study.

Chapter 2

Vehicle routing with multiple commodities: a survey

Contents

Abstract: In this chapter we propose a survey on vehicle routing problems with multiple commodities and we focus on the routing problems where multiple commodities are explicitly considered. It is a common assumption when solving routing problems to implicitly consider multiple commodities, for example by aggregating the requested commodities based on volume or weight to form a single demand, or by decomposing the problem by commodity (e.g. if they are delivered from different depots, or the vehicles are dedicated to a single commodity).

This practice may lead to infeasible solutions since regulations impose that particular commodities (e.g. food and detergent) cannot be transported together or dedicated vehicles are needed for special commodities (e.g. vehicles with capacitated compartments for petroleum products). Moreover, the explicit consideration of several commodities may lead to transportation planning with lower delivery cost. This is the case, for example, when customer requests can be split with respect to the commodities and several visits to customers are allowed. These savings come at the price of more complicated (thus challenging and interesting) optimization problems.

Moreover, vehicle routing problems that consider multiple commodities are very relevant in the real life with applications in transportation of food, distribution of petroleum products, collection of waste, transportation of hazardous materials.

In this chapter, for each routing problem with multiple commodities, we motivate the interest to explicitly consider the multiple commodities, and how they are considered in the models and solving methods. We also provide a review on the main applications of the multiple commodities routing problems.

Keywords: survey, multiple commodities, vehicle routing problems.

2.1 Introduction

The Vehicle Routing Problem (VRP) and its extensions have been widely studied in the last decades (Laporte (2009), Toth & Vigo (2014), Braekers *et al.* (2016)). In the classical Capacitated VRP (CVRP), a set of customers with a known demand have to be served by a homogeneous fleet of capacitated vehicles located at a single depot. The CVRP consists in determining a set of feasible routes that start and end

at the depot and such that each customer is visited exactly once. A route is feasible if the sum of the customers' demands served during the route does not exceed the vehicle capacity. A solution of the CVRP is feasible if all its routes are feasible. The objective is to minimize the total transportation cost.

Note that in this definition of the CVRP, the demand of a customer i is just represented by a scalar $D_i \geq 0$, and the term *commodity* is not even mentioned. The underlying assumption behind this definition is that there is a single commodity delivered to all the customers or that different commodities to be delivered to same customer are aggregated to form a request that occupies D_i units of the capacity of the vehicle.

However, in many cases, classical VRPs implicitly consider multiple commodities. Let us first consider the case where commodities can be aggregated. This case considers that each customer requires a set of different commodities that are compatible, i.e. any set of commodities can be mixed inside the same vehicle. Moreover, it is imposed that all the commodities required by a customer have to be delivered at once by a single vehicle, i.e. the demand of a customer is the sum of the demands of the required commodities and the different commodities do not have to be considered explicitly. The problem that arises by aggregating the commodities is a classical CVRP.

Figure 2.1 shows an example, in which each customer requires two commodities and the vehicle capacity is 10. The number on each edge corresponds to the associated travel cost, and the numbers in the dotted ellipses are the demands for each commodity. By aggregating the commodities, the problem that arises is a classical CVRP where the demands for customers 1, 2 and 3, are respectively 6, 8 and 6. Then, an optimal solution uses 3 vehicles, and the total cost is 25, as depicted in Figure 2.2.

Aggregation of the commodities can also concern extensions of the VRPs. Some examples are provided in the following. In the Split Delivery VRP (SDVRP), a customer can be served by different vehicles, as long as the sum of the quantities delivered by the vehicles equals its demand. This problem has been introduced by Dror & Trudeau (1989) and Dror & Trudeau (1990). For an extensive review of SDVRP, the interested reader is referred to the survey by Archetti & Speranza (2012). Split deliveries are usually interesting when customer demands are large compared to vehicle capacity. Splitting the delivery permits to decrease the transportation

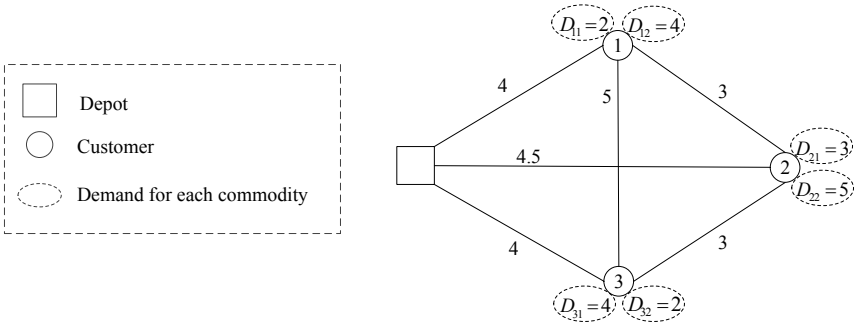


Fig. 2.1. An instance of VRP with two commodities.

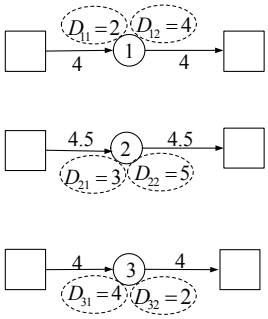


Fig. 2.2. An optimal solution when commodities are aggregated.

costs since it allows for better packing of the demands in the vehicles and results in better use of the vehicle capacity. Moreover it increases the chances to find feasible solutions when the fleet of the vehicles is limited and it is the only viable solutions when the demands of customers are larger than vehicle capacity.

If these large demands result from an aggregation of several commodities, then splitting the deliveries may also imply that the delivery of a single commodity is performed by several vehicles, which may not be convenient especially from the customer point of view. Figure 2.3 shows the optimal solution of the SDVRP for the instance presented in Figure 2.1. Commodity 2 of customer 2 is delivered by two vehicles, the transportation total cost is 23. This strategy minimizes transportation costs but may cause inconvenience to customers.

In the Multi-Depot VRP (Golden *et al.* (1977), Montoya-Torres *et al.* (2015)), the vehicles operate from several depots and a customer can be delivered from any of them. If only one aggregated demand quantity is associated with each customer, it is implicitly supposed that each of the commodity is available at each depot resulting in a much larger inventory than needed. On the other side, explicit consideration of each commodity would require knowledge of its availability in each depot.

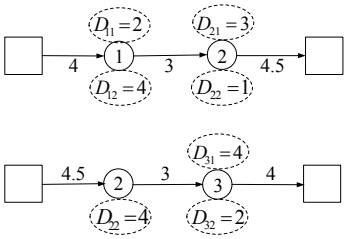


Fig. 2.3. An optimal solution of the SDVRP.

The pickup and delivery problem (Parragh *et al.*, 2008) consists in satisfying a set of transportation requests instead of delivering a set of customers. A transportation request consists in a pair of pickup and delivery operations with an associated quantity that has to be brought from the pickup location to the delivery location. A pickup operation has to take place before the associated delivery operation, and be performed in the same route. One can imagine that different commodities may be part of a single or different requests. However these commodities are usually not explicitly considered. As a consequence, it is tacitly supposed that all commodities are compatible and can be transported by any vehicle of the fleet.

However, all the above assumptions about compatibility among commodities, compatibility between commodities and vehicles, availability of commodities at the depots, split delivery of a single commodity are not always satisfied in real life.

For example, regulations impose that fresh food (meat, fish, dairy products) has to be transported in a refrigerated vehicle, or that food and detergent can not be transported at the same time in the same vehicle, even if the vehicle is compatible with both products separately. It is also the case for bulk organic and conventional food products that cannot be mixed in the same vehicle. More generally, bulk materials cannot be transported in the same vehicle without specific equipment to avoid mixing different types of material.

A different way to implicitly consider multiple commodities in routing problems is to have a natural decomposition by commodity of the original problem, and then to solve separate routing problems. If a specific set of vehicles is dedicated to each commodity, and any commodity has to be delivered to any customer by a single vehicle visit, then the problem can be decomposed by commodities. A consequence of such a policy is that each customer is visited as many times as the number of commodities that he/she requires. In other words, if a customer needs multiple commodities, this customer needs to be served several times, even if all

the commodities come from the same depot and could be aggregated into a single request. The solution to this separate routing problem for the instance presented in Figure 2.1 is shown in Figure 2.4. One vehicle is dedicated to commodity 1, and two vehicles are dedicated to commodity 2. The total transportation cost of such solution is 33.5. The same situation appears in the multi-depot case if each depot is dedicated to a single commodity. Then, the problem can also be decomposed by commodity.

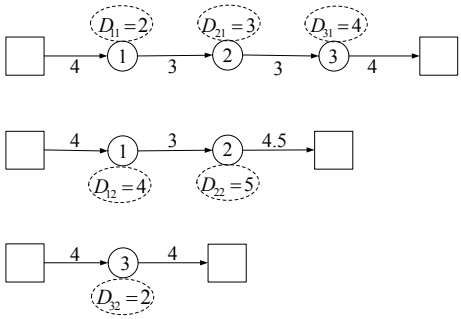


Fig. 2.4. Solution of separate routing problems for each commodity.

Note that the decomposition of the routing problem by commodity may also appear for convenience reasons. As an example, it is easier to load and unload a vehicle carrying a single commodity. A depot dedicated to a single commodity may permit to reduce the transportation costs related to supply since each depot requires a single commodity. It also makes it easier to manage the depot. However, it should be noticed that decomposing the problem by commodity results in higher distribution costs since commodities cannot be mixed in a same vehicle. The example of Figure 2.4 provides an example of this cost increase when we compare with a solution with aggregation of commodities.

In this chapter, we aim at reviewing the literature on vehicle routing where multiple commodities are explicitly considered. The main question we want to answer is: why do we need and when it is beneficial to consider multiple commodities?

We propose to classify the problems in two main categories. In the first, we consider grouping problems with vehicles with multiple compartments. Here different compartments accommodate different commodities in order to allow the transport of incompatible commodities in the same vehicle. Thus, the explicit consideration of multiple commodities is natural for this class of problems. The second category contains extension of well-known routing problems (that usually do not treat multiple

commodities) where multiple commodities are explicitly considered. We also focus on some applications where it is usual to consider multiple commodities explicitly.

In order to ease the description of the problems, from here on we will refer to a *customer-commodity* to indicate a single commodity required by a customer, with its associated demand and the location of the customer. For example, in Figure 2.1, there are two customer-commodities associated with the customer 1: the first with a demand of 2 for commodity 1, the second with a demand of 4 for commodity 2.

The remainder of this chapter is organized as follows. Section 2.2 surveys the multi-compartment vehicle routing problems and related applications. In Section 2.3, six extensions of multi-commodity VRP are surveyed. Section 2.4 discusses two additional applications where multiple commodities are explicitly considered. Section 2.5 concludes the chapter.

2.2 Multi-Compartment VRP and its applications

In this section, we provide an overview of the works on Multi-Compartment VRP. We first describe the problem and its characteristics related to multiple commodities (see Section 2.2.1). Then, the main related applications are reviewed: petroleum products transportation (Section 2.2.2), waste collection (Section 2.2.3), livestock collection (Section 2.2.4), and food transportation (Section 2.2.5).

2.2.1 Multi-Compartment VRP

In the Multi-Compartment VRP (MCVRP), according to the general definition proposed by Derigs *et al.* (2011), each vehicle is divided in the same set of compartments each with a limited capacity and each customer requires a set of commodities. Two dimensions of incompatibilities are considered. First, some pairs of commodities are incompatible and must not be transported in the same compartment. Second, some commodities must not be loaded into some compartments. Each customer may be visited by several vehicles, but each required commodity has to be delivered at once by a single vehicle. In term of assignment decisions, the MCVRP requires to determine, for each customer, in which compartment goes each required commodity. The result is a more complex problem than the classical CVRP where decisions simply involve the assignment of customers to vehicles. The MCVRP imposes additional

constraints specifically related to the compartments: (i) the capacity of a compartment has to be respected, (ii) incompatible commodities cannot be assigned to the same compartment, (iii) each commodity is assigned to a compatible compartment.

Hence the main change from the classical CVRP is in the structure of the vehicles that are adapted to be able to transport several incompatible commodities. When all commodities are incompatible with each other, the routing problem is usually decomposed, which results in a higher routing cost. Using specific vehicles with multiple compartments is a way to decrease the routing costs when commodities are incompatible.

We still refer to the example shown in Figure 2.1, and we assume that each vehicle has 2 compartments with 5 units of capacity for each commodity. Figure 2.5 shows the solution to this instance, the total cost is 31. Obviously, this cost is lower than the case of separate routing (Figure 2.4), but greater than the SDVRP variant (Figure 2.3) where all commodities are compatible, and the deliveries can be split.

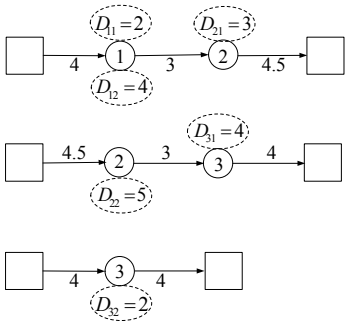


Fig. 2.5. Solution of the MCVRP.

Note that in the definition of the MCVRP provided by Derigs *et al.* (2011), the number of compartments of each vehicle, as well as the size of each compartment, are fixed. Henke *et al.* (2015) introduce a variant of the MCVRP where the number and the size of the compartments on each vehicle need to be determined. Differently than Derigs *et al.* (2011), in Henke *et al.* (2015) compartments may transport all commodities but only one at a time. As a consequence, commodities are supposed to be incompatible with each other.

In the following, we address some works on MCVRP. El Fallahi *et al.* (2008) study a case where compartments are not flexible, and each compartment is dedicated to a single commodity. Two algorithms are proposed to solve the problem: a memetic algorithm improved by a path relinking method, and a tabu search. For

each algorithm, the MCVRP is first separated into several VRPs, one per commodity, in order to obtain an initial solution. A solution is represented by considering all the customer-commodities in the routes. Derigs *et al.* (2011) propose a set of heuristic components to solve the general case of the MCVRP. The representation of a solution is as El Fallahi *et al.* (2008). They propose several construction heuristics, local search operators, large neighborhood search, and metaheuristics.

Mirzaei & Wøhlk (2019) study two versions of the MCVRP where compartments are not flexible, and each compartment is dedicated to a single commodity. In the first version, the commodities of a single customer can be delivered by different vehicles, while in the second version all the commodities of a single customer have to be delivered at once by a single vehicle. They developed a branch-and-price algorithm to solve these two variants of the MCVRP. Henke *et al.* (2015) and Henke *et al.* (2019) address the MCVRP where the compartments are flexible, and their capacity can take some discrete values. Each vehicle has a maximum number of compartments, potentially lower than the number of commodities, and the capacity of the compartments should be multiple of a defined unit capacity. Henke *et al.* (2015) propose a variable neighborhood search algorithm while in Henke *et al.* (2019) the authors propose a Mixed Integer linear Programming (MIP) formulation, and develop a branch-and-cut algorithm to solve the problem. Coelho & Laporte (2015) propose a classification of the MCVRP, based on the possibility to use a compartment for one or several deliveries, and the possibility to split the delivery of a customer or not. For each case, they propose two MIP formulations, with explicit and implicit assignment of products to compartments.

The MCVRP arises in many real-life applications. In the following sections, we first overview the most frequent applications of MCVRP: petroleum distribution, waste collection, livestock collection, and food transportation.

2.2.2 Petroleum distribution

The petroleum products/fuel distribution problem is known as the petrol station replenishment problem (Cornillier *et al.*, 2008a). It is the most widely studied application of the MCVRP. The objective is to minimize the transportation costs for the distribution of several fuel products to a set of fuel stations (customers), using vehicles with multiple compartments since the fuel products cannot be mixed in the same compartment. Hence, each fuel product represents a commodity. Since

accidental mixing of fuel products can be hazardous, compartments are not flexible and well separated from each other (Chajakis & Guignard, 2003). According to Cornillier *et al.* (2008a), compartments are usually not equipped with debit meters, which implies that when a delivery is made from a compartment to a tank in the fuel station, all the quantity loaded in the compartment has to be delivered. Hence, each compartment is dedicated to a single customer and a single commodity. Several compartments can be used to deliver a single commodity. In this application, all the commodities required by a customer have to be delivered at once by a single vehicle. Note that in practice a vehicle contains from 3 to 6 compartments and stations require from 2 to 3 fuel products, hence the number of stations visited on a given route rarely exceed 2 (Cornillier *et al.*, 2008a). Moreover, the quantity of a fuel product delivered to a station is a decision variable. It has to be sufficient to cover the demand for this product, but cannot exceed the capacity of the tank in the station.

The introduction of the petroleum products distribution can be credited to Brown & Graves (1981). They studied a special case with time windows for the delivery considering only direct trips to the stations. Later, more and more researchers began to pay attention to the petroleum products distribution problems. Avella *et al.* (2004) study a particular case where the vehicle compartments cannot be partially filled: they have be either completely filled or empty. The authors propose a fast heuristic and a branch-and-price algorithm to solve the problem. Cornillier *et al.* (2008a) propose an exact algorithm to solve the problem by decomposing the routing problem and the vehicle loading problem. The routing problem takes advantage of the fact that routes are usually short and can be enumerated. The vehicle loading problem is formulated as a MIP, and the authors propose an optimal algorithm to solve the problem. Cornillier *et al.* (2009) study an extension of the problem where stations have to be delivered within specified time windows. In that case, a vehicle can perform several trips during the one-day planning horizon. The authors propose two heuristics to solve the problem.

The petroleum products are usually delivered over a planning horizon of several days. Hence they integrate inventory management for the tanks in the stations. Cornillier *et al.* (2008b) propose a MIP model and a heuristic to solve the problem in the multi-period case. In this case, the objective is to maximize the total profit.

Cornillier *et al.* (2012) address the multi-depot version, and they propose a heuristic based on the generation of feasible pairs of routes and vehicles.

Surjandari *et al.* (2011) study a particular case faced by the national petroleum company in Indonesia. A set of 208 petrol stations require two fuel products within a given time window. A fleet of 76 heterogeneous vehicles are located in two depots and have to deliver the stations over a horizon of one day. A vehicle can perform several trips. In this work, it is allowed to split the delivery for a station, but not for a given commodity. The authors proposed a tabu search algorithm to solve the problem. Christiansen *et al.* (2015) study a real-world application in Greece for an oil company where commodities are different types of fuel. The customers are ships that are supplied with fuel by a fleet of specialized fuel supply vessels. The vessels have fixed compartments where the fuels are loaded. Each vessel can perform several trips during the planning horizon. Each trip starts by refilling the compartments with fuel. In a given trip, a compartment contains only one type of fuel, but in the next trip the type of fuel loaded in this compartment may change. Different compartments can contain the same type of fuel. Moreover, if different ships require the same fuel type, a single compartment can be used to deliver them. The authors propose a MIP formulation for the problem.

Vidović *et al.* (2014) and Popović *et al.* (2012) study the inventory routing version of the petrol station replenishment problem. The planning horizon is divided into several periods, and the consumption at each station is known for each period. Each station can be served only once, i.e. splitting is not allowed for a customer. The objective is to minimize the inventory and routing costs, while ensuring that the level of fuel at the stations always belong to a given interval. Vidović *et al.* (2014) propose a MIP formulation and a heuristic approach based on Variable Neighborhood Descent (VND) and local search to solve this problem. Popović *et al.* (2012) develop a Variable Neighborhood Search (VNS) heuristic to tackle this problem. The two main procedures of the VNS are the local search and the shaking procedure, which are based on three adapted neighborhoods: relocation of individual compartments, relocation of all compartments for the current station's fuel type, and relocation of all compartments for the current station. The proposed VNS heuristic is compared to the results obtained by Vidović *et al.* (2014). The results showed that the VNS provides better results on larger size instances.

2.2.3 Waste collection

The collection of waste is an essential and challenging operational problem for any manager who is in charge of it (Golden *et al.*, 2002). In this context, different types of waste have to be collected, for example, general waste, glass, paper, plastic. Each type of waste has a specific disposal or recycling process and naturally represents a commodity. If all commodities are aggregated, then they have to be sorted when they arrive at the depot, which is a complex and time consuming task. Nowadays, waste is usually sorted by the users, and then the collection can be performed by decomposing the problem by commodity (dedicated vehicles are used for each type of waste), or by using more sophisticated vehicles with multiple compartments. In the latter case, compartments are not flexible, and each compartment is dedicated to a single commodity. Muyldermans & Pang (2010) conduct experiments that show that using vehicles with multiple compartments is beneficial, especially when the number of commodities increases, when the vehicle capacity increases, when a large number of clients request the collection of all commodities, when the client density is lower and when the depot is centrally located in the collection area. A customer may be collected by a single vehicle (Reed *et al.*, 2014), or by different vehicles, but each commodity to be collected is collected by a single vehicle (Muyldermans & Pang, 2010).

Muyldermans & Pang (2010) propose a local search procedure with classical operators (2-opt, cross, exchange and relocate) to solve the problem. The representation of a solution is obtained by duplicating each customer as many times as the number of commodities required. Reed *et al.* (2014) study a version of the problem where all the commodities of a customer are collected at once by a single vehicle. They develop an ant colony algorithm for this problem. Henke *et al.* (2019) study the case where compartments are flexible. This comes from a real-world application where different types of glass (colorless, green, brown) have to be collected.

Note that there are several works on vehicle routing with application on waste collection that do not consider multiple compartments. The interested reader can refer to the review by Ghiani *et al.* (2013) for example.

2.2.4 Livestock collection

Transportation of live animals is a problem where a fleet of heterogeneous vehicles with limited capacity collects animals from a set of farms in order to deliver them to a central slaughterhouse over a planning horizon of several days (Gribkovskaia *et al.*, 2006). The slaughterhouse has a demand for several types of animals (bovine, sheep, pigs). Each day, vehicles with different compartments collect the animals in order to satisfy the demand at the slaughterhouse. Due to animals welfare regulations, different types of animals cannot be mixed inside the same compartment. Thus, it is natural to consider each type of animal as a commodity and to impose incompatibility among commodities. A farm can be visited by several vehicles, but all the animals of the same type have to be collected at once by a single vehicle. The planning extends over several days since it is possible to keep the animals at the slaughterhouse, which makes it possible to anticipate some demands during the collection.

If we focus on the vehicle routing part of the problem, the vehicles are divided into compartments, and different types of animals cannot be mixed inside the same compartment. The vehicles are very specific and their configuration is dynamic (Oppen *et al.* (2010)). In (Oppen *et al.*, 2010) vehicles are divided horizontally into three sections with permanent partitions. Usually, it is possible to split these sections in an upper part and a lower part using a movable floor. Some vehicles have enough height to have pigs or sheep in the upper compartment and bovine in the lower. Since loading operations take place only from the rear of the vehicles and different animals cannot share the same compartment, the order in which the animals are loaded into the vehicle is critical to optimize capacity.

Gribkovskaia *et al.* (2006) present a MIP model for the livestock collection problem. This MIP formulation is only able to solve very small size instances. Oppen & Løkketangen (2008) propose a tabu search heuristic algorithm to tackle larger instances. They represent a solution by considering all the customer-commodities in the routes. Oppen *et al.* (2010) develop a column generation procedure to provide optimal solutions to the livestock collection problem for instances with less than 30 orders to collect. Miranda-De La Lama *et al.* (2014) provide a general review on livestock transportation, which does not focused only on vehicle routing.

2.2.5 Food transportation

The MCVRP arises in food transportation because of two mains reasons: the quality of products and the required temperature.

Caramia & Guerriero (2010) study a milk collection problem where different types of raw milk produced by the farmers have to be transported in vehicles with multiple compartments. The types of milk represent commodities. The compartments are not flexible, and each compartment can be filled with a single commodity. Each farmer can be visited by several vehicles, even for the same commodity. El Fallahi *et al.* (2008) consider an application of distribution of cattle food to farm. Lahyani *et al.* (2015) study the collection of olive oil in Tunisia. Each commodity represents a quality of olive oil: extra, virgin, and lampante. The vehicles contain several compartments, and different commodities cannot be mixed in the same compartment since the problem is studied on a multi-period planning horizon, the vehicle can perform different trips. Additional constraints impose to perform cleaning operations if a compartment loaded with lampante oil has to transport another type of commodity in the subsequent trip. Cleaning operations involve additional cost and time.

Chen *et al.* (2019) solve a MCVRP for a cold-chain distribution company which manages the distribution of fresh perishable foods. Each commodity represents a group of products that has to be transported within a certain temperature zone. Each compartment is dedicated to a single commodity. Chajakis & Guignard (2003) and Ostermeier & Hübner (2018) also consider similar problems where the compartments correspond to different temperature zones.

2.3 Multi-Commodity VRP extensions

In this section, we review vehicle routing problems that explicitly consider multiple commodities. First, we consider extensions for the one vehicle case, i.e. the Traveling Salesman Problem (TSP): the Multi-Commodity Pickup and Delivery Traveling Salesman Problem (Section 2.3.1), and the Traveling Purchaser Problem (Section 2.3.2). Then, we review variants for the multi-vehicle case: the Commodity constrained Split Delivery VRP (Section 2.3.3), the Multi-Commodity Inventory Routing Problem (Section 2.3.4), the Multi-Commodity Multi-Trip VRP (Section 2.3.5), and the Multi-Commodity Location Routing Problem (Section 2.3.6).

2.3.1 Multi-Commodity Pickup and Delivery TSP

The Pickup and Delivery Traveling Salesman Problem (PDTSP), also known as the one-commodity PDTSP, introduced by Hernández-Pérez & Salazar-González (2004), is a generalization of the classical TSP where the customers are divided into two groups: delivery customers that require a given amount of a commodity, and pickup customers that provide a given amount of this same commodity. A single vehicle with a fixed capacity has to perform a single tour to visit all the customers, starting and ending at the depot. The initial load of the vehicle can take any value between 0 and the capacity. The visit of a delivery customer decreases the load of the vehicle while the visit of a pickup customer increases the load. The objective of the PDTSP is to design a minimum cost Hamiltonian route such that the load of the vehicle is always feasible (between 0 and the capacity). It is assumed that a single commodity that has to be transported from pickup points to delivery points without pairing these points. Note that, according to the classification of pickup and delivery problems (Battarra *et al.*, 2014), this definition corresponds to the *many-to-many* problems where a single commodity has multiple origins and destinations. For an extensive review of the pickup and delivery problems, the interested reader is referred to the survey by Berbeglia *et al.* (2007).

In the Multi-Commodity Pickup and Delivery Traveling Salesman Problem (m-PDTSP), a set of different commodities are transported by a single capacitated vehicle. Each customer requires and/or provides a given quantity of one or several commodities and must be visited once by the vehicle. All the commodities are compatible and can be mixed in the vehicle. The initial quantity of each commodity in the vehicle is a decision variable. The m-PDTSP aims to determine a Hamiltonian circuit such that all pickup and delivery requirements are satisfied and the vehicle capacity is not exceeded. The objective is to minimize the total transportation cost. In this problem, it is important to explicitly consider the different commodities since we have to ensure that there is sufficient amount of each required commodity in the vehicle when a customer is delivered.

The m-PDTSP was introduced by Hernández-Pérez & Salazar-González (2014). A straightforward application of the m-PDTSP is the inventory repositioning between retailers (Anily & Bramel, 1999). Some retailers have an excess of inventory for some commodities which others have a lack of stock. The firm, that manages all

retailers, wants to use a vehicle to balance the inventory levels between the retailers. An example is the self-service bike-sharing system (Raviv *et al.*, 2013), where every night, a capacitated vehicle picks up or delivers the bikes from/to the bike stations to restore the initial configuration of the system. Multiple commodities are considered when there are different types of bikes (for instance, with and without baby chairs). Another example involves bank services (Hernández-Pérez & Salazar-González, 2007) where some branches of a bank provide or require money. Here the depot is the main branch of the bank. When different bills and coins are considered, the problem to solve is a m-PDTSP.

Hernández-Pérez & Salazar-González (2014) propose a MIP formulation for the m-PDTSP, and some valid inequalities to strengthen the model. They develop a branch-and-cut procedure to solve the problem. Hernández-Pérez *et al.* (2016) propose a three stage algorithm where they apply constructive heuristics and local search operators to modify the route. Note that, for each solution, it is necessary to apply a procedure to check the feasibility of the solution. With n customers and m commodities, the authors propose an improved checking procedure in $\mathcal{O}(m \log n)$.

In the following, we list some problems related to the m-PDTSP. The non-preemptive capacitated swapping problem (NCSP) was proposed by Erdoğan *et al.* (2010). Multiple commodities are considered, but each customer requires or provides a single commodity, or two commodities (one for pickup and the other one for delivery). Moreover, some transshipment locations may be used for the temporary storage of the load of the vehicle. Hence, the route of the vehicle is not a Hamiltonian circuit anymore. The one-to-one m-PDTSP is a special case of the m-PDTSP in which each commodity must be transported from a given pickup customer (origin) to a given delivery customer (destination). In that case, it is not necessary anymore to explicitly consider multiple commodities since the problem corresponds to the capacitated version of the TSP with precedence constraints (Hernández-Pérez & Salazar-González, 2009). Li *et al.* (2019) study a real life problem faced by a fast fashion retailer. The application considers a warehouse and a set of stores. Each week, based on the forecasted demands, a fleet of vehicles picks up and delivers several products from/to the stores in order to balance the storage levels of these products at the stores. Moreover commodities can be picked-up and delivered from/to the warehouse with an additional handling cost. The underlying problem

is a variant of the m-PDTSP with multiple vehicles and handling costs at the warehouse. A branch-price-and-cut algorithm is developed to solve the problem. Zhang *et al.* (2019) propose a heuristic method to solve this problem. They design some local search operators, and they propose a procedure to check the feasibility of a segment of consecutive nodes in $\mathcal{O}(m)$ where m is the number of commodities. Each segment is associated with information for each commodity.

2.3.2 Traveling Purchaser Problem

The Traveling Purchaser Problem (TPP) is an extension of the Traveling Salesman Problem (TSP) in which a purchaser is based at the depot, and other nodes represent suppliers. The purchaser has a list of commodities to buy with an associated required quantity. A set of suppliers offer a limited amount of these commodities (possibly 0) with an associated unit selling price. The TPP consists in finding an optimal tour, starting and ending at the depot, that visits some of the suppliers in order to buy all the required commodities. The objective is to minimize the sum of traveling and purchasing costs. The TPP involves compatible commodities that must be explicitly considered due to the required quantities, and the availabilities and the selling prices at the suppliers.

There is are numerous papers on the TPP. The interested reader is referred to the recent literature review by Manerba *et al.* (2017). In vehicle routing, a common application for corresponds to a company in charge of the collection of several raw materials from a set of reliable suppliers. Another application is the school bus routing, where suppliers correspond to bus stop, and the commodities to students to pick-up (Riera-Ledesma & Salazar-González, 2012). A student may be assigned to different bus stops, and the objective is to find a route for the bus such that all students are picked-up at a bus stop, while minimizing the total transportation cost and assignment cost of the students to stops of the bus.

2.3.3 Commodity constrained Split Delivery VRP

The name C-SDVRP was introduced by Archetti *et al.* (2014). The Commodity constrained Split Delivery VRP (C-SDVRP) is an extension of the Split Delivery VRP in which additional constraints are imposed to split a delivery. Each customer requires multiple commodities, that are compatible with each other. The demand

of a customer can be split, but each customer-commodity must be delivered by a single vehicle. As a consequence, a customer is visited at most as many times as the number of commodities he/she requires.

As mentioned in the introduction (Section 2.1), the aggregation of commodities leads to large customer demands, when multiple commodities are compatible. Hence, splitting the delivery permits to optimize the load of the vehicles and, then, to decrease the transportation costs. On the other side, this is not convenient for the customers: if a single commodity is split, a customer may need to handle the deliveries of the same commodity several times during the planning horizon.

The C-SDVRP is an intermediate delivery strategy between delivering all the commodities at once by a single vehicle, and arbitrarily splitting the deliveries. Note that the C-SDVRP can be seen as a special case of the MCVRP where only one compartment is available and all commodities are compatible. Thus the assignment of commodities to compartments is trivial. The C-SDVRP can also be viewed as a special case of the MCVRP with flexible compartments. We have as many compartments as the commodities and the step of division is the greatest common divisor of the customer demands. Thus, each packing of commodities is a feasible solution of the problem that determines the size of the compartments.

Figure 2.6 shows the solution to the C-SDVRP for the instance provided in Figure 2.1. Customer 3 is served by two vehicles, and the total cost is 24.5. This cost is higher than the cost of the VRP when commodities are aggregated, and lower than the cost of the SDVRP.

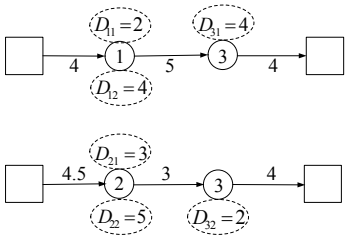


Fig. 2.6. Solution of the C-SDVRP.

The C-SDVRP is of practical relevance when the demand consists in a set of items with different sizes (Nakao & Nagamochi, 2007). Indeed, in this case, the solution of the SDVRP may not be feasible since the splitting of a customer request may not correspond to a splitting of the set of items required by the customers. Another application of the C-SDVRP (Gu *et al.*, 2019) occurs when we consider categories of

products: dairy products, fresh fruits, or vegetables. A group of products represents commodities which are compatible. From the customer point of view, it is acceptable to have more than one delivery, but splitting the delivery of a specific commodity (a category of products) is not practical. When a few commodities are considered, the number of deliveries for a customer remains acceptable.

Nakao & Nagamochi (2007) introduce this problem under the name Discrete Split Delivery VRP. They propose a heuristic based on dynamic programming, where the customer-commodities are iteratively inserted into the routes. Ceselli *et al.* (2009) study a rich VRP where the demand of a customer consists in a set of items where each item is a pallet of products with specific dimensions. Each item represents a commodity. The delivery of a customer can be split, but it is not possible to split the delivery of an item. They also consider incompatibilities between commodities and vehicles. The authors propose a three-phase column generation approach: in the first phase, splitting is not considered; in the second phase, splitting of large demands is possible by considering customer-commodities instead of customers; in the third phase, all the customer-commodities are considered in the pricing algorithm. Archetti *et al.* (2014) introduce the name C-SDVRP, and propose to solve the problem as a CVRP considering all the customer-commodities. A branch-and-cut is used to solve small instances, and a heuristic algorithm for the CVRP is used to solve medium and large instances. Archetti *et al.* (2015) propose an extended formulation for the C-SDVRP and develop a branch-price-and-cut algorithm. Gschwind *et al.* (2019) develop a new branch-price-and-cut algorithm that includes stabilization techniques and uses the dual optimal inequalities for the stabilization of the column-generation process. Gu *et al.* (2019) propose an Adaptive Large Neighborhood Search (ALNS) heuristic to address medium and large size instances. The destroy and repair operators, as well as local search operators are designed for a representation of the solution with either customers or customer-commodities. Depending on the operator to apply, only one representation of the solution is considered.

Salani & Vacca (2011) study an extension of the C-SDVRP where any subset of commodities required by a customer represents an order. With each order is associated a service time, which may not be linear in the number of commodities or the size of the order. This problem is solved by a branch-and-price algorithm.

2.3.4 Multi-Commodity Inventory Routing Problem

Multi-Commodity Inventory Routing Problem (MCIRP) is an extension of the Inventory Routing Problem (IRP). The IRP is different from the classical VRP since it is based on customers' consumption rather than customers' orders. It is then a combination of vehicle routing and inventory management problems. The IRP considers the distribution of a single commodity, from a supplier (the depot), to a set of customers over a given planning horizon. At each period, a quantity of the commodity is available at the supplier, with the possibility to make storage between two periods. Each customer consumes the commodity at a given rate (number of units per period) and maintains a local inventory of the commodity up to a maximum volume. A fleet of homogeneous capacitated vehicles is available for the delivery of the commodity. The objective is to minimize the average distribution costs plus the inventory costs over the planning horizon without causing any stockout at the customers. The interested reader is referred to the literature review on the IRP by Coelho *et al.* (2013).

Multi-Commodity Inventory Routing Problem (MCIRP) is the extension the the IRP when multiple commodities have to be delivered to the customers. It is often referred in the literature as multi-product IRP. Usually, multiple commodities have to be considered explicitly since they share some common resources (storage capacity, vehicle capacity). They also have different inventory costs and customers' consumption rates.

Several applications motivate the study of the MCIRP. The most common one is related to maritime logistics, i.e. different types of fuel and gases have to be delivered using compartmentalized ships. Other applications are in the distribution of perishable commodities, the transportation of gases by tanker trucks, the automobile components industry, and fuel delivery (Coelho & Laporte, 2013). In the IRP and the MCIRP, the number of suppliers and customers may vary, and the problems are usually classified based the structure of the distribution network: *one-to-one*, *one-to-many*, *many-to-one* or *many-to-many*. In the following, we review the works on MCIRP according to this classification.

In the *one-to-one* case, a single supplier serves a single customer. Speranza & Ukovich (1994, 1996) consider a special case of the problem where several commodities are shipped from the supplier to the customer on a regular basis over an infinite horizon. More precisely, they consider a set of finite frequencies to ship

the commodities. A frequency is expressed as a number of trips per period. Each commodity is produced at the supplier and consumed by the customer at a given constant rate. A product may be partially shipped with different frequencies, and the commodities can be mixed inside the same vehicle. The problem is to determine, for each frequency, the portion of each commodity shipped, and the number of vehicles to use. The objective is to minimize the total transportation and holding costs while ensuring that all the products are shipped to the customer. Multiple commodities have to be explicitly considered since they do not have the same unit holding cost, unit volume and rate of production, but they share the same vehicles. Note that Speranza & Ukovich (1994) also consider a variant where all simultaneous shipments (maybe with different frequencies) may share the same vehicles. Speranza & Ukovich (1994) propose MIP formulations for the two variants of the problem. Speranza & Ukovich (1996) propose a branch-and-bound to solve the first variant.

One-to-many is the most common case in which one supplier delivers several customers. It directly refers to the definition provided at the beginning of this section. Coelho & Laporte (2013) propose a MIP formulation and some additional valid inequalities. They develop a branch-and-cut to solve the problem to optimality. Cordeau *et al.* (2015) consider a different version of the problem where the commodities are not mixed in the inventory of the customers and have a dedicated storage capacity. This arises, for example, when customers have dedicated storage locations for each commodity and cannot mix them. The authors propose a three-phase heuristic to tackle this problem. The first phase consists in planning the delivery quantities for each period, each customer and each commodity while ensuring the respect of the capacity constraints. In the second phase, the delivery quantities computed in the first phase are aggregated for all the commodities. Then, for each period, a set of routes is determined in order to deliver the customers with their corresponding aggregated quantities. The third phase is a reoptimization procedure. Popović *et al.* (2012) consider an application in fuel delivery with multi-compartment homogeneous vehicles. Only full compartments are delivered to petrol stations. The authors propose a MIP formulation and develop a Variable Neighborhood Search (VNS) heuristic to solve the problem. In the VNS, the shaking procedure does not consider the routing decisions, but only changes the inventory decisions by modifying the delivering periods for customers or customer-commodities. In the local search procedure, they develop intra-period operators

for the routing part (only considering customers) and inter-period operators on the delivery planning by moving customers or customer-commodities. Shaabani & Kamalabadi (2016) study an extension of the MCIRP when the commodities are perishable. This results in additional constraints for the storage capacity. In the proposed solution method the whole problem is decomposed into a set of routing subproblems (one for each period) and an inventory subproblem based on the solution of the routing subproblems. The routing subproblems only consider customers and are solved using a population-based simulated annealing heuristic. The inventory subproblem is modeled as a MIP and solved with a commercial solver. The commodities only appear in the inventory subproblem.

The case *many-to-one* considers a single depot, multiple suppliers and only one customer. Moin *et al.* (2011) study this distribution network where many suppliers deliver a single assembly plant. It is assumed that the customer requires multiple commodities and each supplier provides a single commodity. A route starts at the depot, visits some suppliers to pick up the commodities and then visits the customer to deliver the commodities just before going back to the depot. A supplier may be visited by more than one vehicle during a given period in order to reduce transportation costs. In the proposed genetic algorithm, there are two representations of a solution. The first one is based on the visits of the suppliers over the planning horizon, and the quantities to collect are computed based on a simple inventory rule. It consists in collecting at period t, the sum of the demands from t to t' where t' is the next period during which the commodity is collected. The second representation only focuses on the collected quantity and is more flexible than the inventory rule mentioned before. Mjirda *et al.* (2014) propose a two-phase heuristic based on VNS. The representation of a solution corresponds to the set of routes performed during each period. The first phase consists in solving a VRP for each period without considering the inventory. In the second phase, the moves modify the routes, and the solution is evaluated taking into account the different commodities for the inventory. Based on a routing solution, the inventory subproblem is solved using a MIP solver or a dedicated heuristic.

The case *many-to-many* considers several suppliers and several customers. Ramkumar *et al.* (2012) study the extension of the *many-to-one* case where there are several customers. Each supplier may produce several commodities, but a single supplier

produces each commodity. Each route begins at the depot, then consists of a sequence of pick-ups from suppliers, followed by a sequence of deliveries to customers before returning to the depot. At each period, the collection and the delivery of an individual commodity may be split, i.e. performed by different vehicles. Ramkumar *et al.* (2012) propose a MIP formulation for this problem. Ghorbani & Jokar (2016) study a variant of in a two-echelon supply chain: each supplier delivers one or several depots using direct trips, and each depot has an inventory and delivers several customers performing routes. Decisions involve the depots to open and the assignment of customers to depots. They propose a MIP formulation and a hybrid algorithm based on Simulated Annealing (SA) to solve the problem. Decisions are taken in sequence: first, the design of the network, then the inventory management, and finally the routing decisions.

2.3.5 Multi-Commodity Multi-Trip VRP

The Multi-Commodity Multi-Trip VRP with Time Windows (MC-MTVRP-TW) was introduced by Battarra *et al.* (2009). It is an extension of the Multi-Trip VRP with Time Windows (MTVRP-TW) that in turn extends the Multi-Trip VRP (MTVRP). Cattaruzza *et al.* (2016) provide a review for the MTVRP. In the MTVRP each vehicle is allowed to perform several trips during the planning horizon. To allow vehicles to perform multiple trips is important, especially when the vehicle capacity is small compared with the demand over the time horizon. In the MTVRP-TW the delivery at the customer location must take place during a pre-determined time interval called Time Window (TW).

The MC-MTVRP-TW explicitly considers different commodities. It is assumed that commodities are incompatible and cannot be transported during the same trip. However, a vehicle can deliver incompatible commodities in different trips. As a consequence, each commodity for a customer is delivered in different trips. Thus, the TWs are not associated with the customers (for all the deliveries) but with the customer-commodities. The objective is to minimize the size of the fleet. Ties are broken by minimizing the routing cost. Allowing a vehicle to carry different commodities for each trip makes it possible to make a full use of the vehicles during the planning horizon, and therefore to reduce the size of the fleet. Using vehicles dedicated to a single commodity would lead to decompose the problem by commodities (see Section 2.1).

In Battarra *et al.* (2009), the application that motivates the MC-MTVRP-TW is the distribution of three commodities to supermarkets: vegetables, fresh products, and non-perishable items. The regulation imposes that these commodities are incompatible with each other and cannot be transported together in a vehicle. The authors propose an adaptive mechanism that is used to guide simple heuristics. Cattaruzza *et al.* (2014) develop an iterated local search for the MC-MTVRP-TW. Note that the methods of both authors consider instances with customer-commodities since two customer-commodities associated with the same customer will never be served together on the same trip.

2.3.6 Multi-Commodity Location Routing Problem

In the classical Location Routing Problem (LRP), there are a set of potential depots associated with an opening cost. A set of customers with known demands for a single commodity must be served from the opened depots using a fleet of vehicles. Each customer has to be visited by a single vehicle. No inventory considerations apply, neither at depots nor at customers. The LRP consists in determining the depots to open, assigning each customer to one open depot, and determining the vehicle routes to deliver customers from the open depots. The objective is to minimize the total cost: opening costs of the depots, fixed costs for each vehicle used, and the total routing cost (Prodhon & Prins, 2014). For the review of the LRP, interested reader is referred to the surveys by Prodhon & Prins (2014) and by Drexl & Schneider (2015).

The Multi-Commodity Location Routing Problem (MCLRP) is studied in the context of many-to-many LRP introduced by Nagy & Salhi (1998), where the network structure consists of three layers with suppliers, potentials depots to open and customers. Suppliers and customers have a known demand to be, respectively, picked-up or delivered for a set of commodities. At each open depot, a fleet of vehicles performs routes with pickup operations at the suppliers and delivery operations at the customers. Some routes may contain only pickup or only delivery operations. Each supplier and customer is visited once by a single vehicle. Another fleet of vehicles performs direct trips between the open depots in order to consolidate the commodities at the depots. All the commodities are compatible, thus can be mixed inside the vehicles. The objective is to minimize the total opening and transportation costs. In the original many-to-many LRP studied by Nagy & Salhi (1998), it

was assumed that a couple supplier-customer is associated with each commodity. Hence the multi-commodity aspect of the problem was not explicitly considered. At the opposite, it is necessary to explicitly consider commodities when they are produced by several suppliers and/or required by several customers.

Burks (2006) is known as the first work on MCLRP (Boccia *et al.*, 2018a). In addition to the above definition of the MCLRP, they also consider timing constraints (time windows, maximal route duration) and the decisions about the opening of the supply points. They consider an application for the distribution of personnel, equipment and material to a theater of war. Customers are cities, suppliers are airports, sea-port or train stations, and depots are the locations where the vehicles are based. Multiple commodities are considered since the same resources are used to transport materials and personnel to the theater of war. Burks (2006) propose a MIP formulation and a Tabu Search (TS) algorithm to solve the problem.

Rieck *et al.* (2014) study the MCLRP, motivated by a real-life application in the timber-trade industry. Suppliers are sawmills and wood manufacturers, and customers are distributors of wood products. Sawmills produce some commodities like beams and planks from different kind of woods, whereas wood manufacturers produce wood for construction purpose or indoor equipment (e.g. door and window frames). Distributors of wood require a mix of all these commodities for their production or retail activities. In order to avoid extensive stock holding at the customers, and to better exploit the capacity of the trucks, it is beneficial to consider a supply network that explicitly takes into account these commodities. Rieck *et al.* (2014) propose a MIP formulation and some valid inequalities. They develop a heuristic procedure and a genetic algorithm to solve medium and large size instances. In both algorithms, partial solutions represent the depots to open, and the pickup and delivery routes performed from these depots. Based on these partial solutions, a multi-commodity fixed charge network flow problem is solved to determine the number of direct trips between depots.

Gianessi *et al.* (2015) study a particular variant of the MCLRP: the multi-commodity-ring LRP, where the depots to be located have to be connected via a ring, i.e. the selected direct trips between the open depots must form a Hamiltonian cycle. The application is in city logistics where depots represent urban distribution centers (UDCs), and self-service parking lots (SPLs) permit to park electric vehicles. The ring connecting depots (UDCs) is used to transport massive flow of goods from

one depot to another. Final customers are served by electric vehicles performing routes. A route starting at a depot (UDC) may end at the same depot, another depot, or a SPL, while a route starting at a SPL ends at a depot. Gianessi *et al.* (2015) propose a MIP formulation based on set partitioning where an exponential number of variables represent the feasible routes. They propose a matheuristic algorithm to solve this problem: only a subset of the feasible routes is generated, and the solving is decomposed in three phases. Based on a set of routes, the first phase decides the depots to open and the routes assigned the depots; the second phase builds the ring; the third phase sends the flow of commodities along the ring. This third phase is modeled as a multi-commodity flow problem.

Rahmani *et al.* (2016) consider a MCLRP with two-echelon distribution network. At the first echelon, a single supplier produces several commodities and performs routes to deliver the open depots. At the second echelon, each route starts and ends at the same depot. A route is composed of delivery operations at the customers and potential pickup operations at the depots. Hence, a route may visit several depots to pick up some commodities. Note that the delivery of a customer may be split, but the delivery of an individual commodity cannot be split. To solve the problem, the authors propose extensions of nearest neighbor, insertion, and clustering heuristics.

Boccia *et al.* (2018b) also address a MCLRP in a two-echelon distribution network. Each commodity has a single origin (supplier) and a single destination (customer). The authors propose a MIP formulation based on a flow model where each commodity explicitly appears.

2.4 Other applications with multiple commodities

In this section, we overview two applications of routing problems where multiple commodities are considered: transportation of many hazardous materials (Section 2.4.1) and multi-commodity disaster relief (Section 2.4.2).

2.4.1 Transportation of multiple hazardous materials

The transportation of hazardous materials (hazmat or dangerous goods, e.g., gasoline, toxic gases) is treated differently from the classical transportation problems since the risk associated with accident or incident has to be considered. Due to its nature, the transportation of hazardous materials is regulated by law in most

countries (Holeczek, 2019). There are several modes of transportation for hazmat: road, rail, water, air, and pipeline (Erkut *et al.* (2007), Bianco *et al.* (2013)). Hazmat routing problems often aim at finding one or several paths to route a single commodity from its origin to its destination (Holeczek, 2019). However, selecting optimal routes for each pair of origin-destination may result in the concentration of the transportation of hazmat on some links of the network. This is dangerous since a huge traffic on a link of the network leads to increase the probability of an accident (Iakovou *et al.*, 1999). Hence, some works explicitly consider multiple hazmat commodities since they have different origin and/or destinations, but are routed in the same network with shared capacity constraints.

Several articles on hazmat transportation with multiple commodities consider origins and destinations for each commodity. The aim is to find an optimal path to route each origin-destination pair in a shared network. Iakovou *et al.* (1999) study the marine transportation of oil products. Each commodity represents an oil product, associated with a transportation cost and a transportation risk. Given a capacitated network and a set of commodities with their associated origins and destinations, the problem is to find a path to route each commodity for each origin-destination pair, such that no arc of the network has a flow exceeding the capacity. The objective is to minimize a trade-off between transportation costs and risks. Iakovou *et al.* (1999) propose a MIP formulation with an exponential number of variables, based on the selection of paths to route each commodity. To solve the problem, they propose a Lagrangean relaxation and adapt a subgradient algorithm to provide both lower and upper bounds. Verter & Kara (2008) study the transportation of several hazmats from the point of view of a single regulator who may forbid the use of some links in the network, and whose objective is to minimize the total exposure of the population. From the carriers' point of view, based on the decisions of the regulator, they choose the path with the lowest transportation cost. They provide a MIP formulation with an exponential number of variables, requiring to list all the paths to ship a commodity from an origin to a destination. They propose some applications of the transportation of hazmats through the highway network of some Canadian provinces. Erkut & Gzara (2008) propose a bi-level multi-commodity network flow model for the problem, as well as a dedicated heuristic. Bianco *et al.* (2009) consider a hazmat transportation problem with two levels of regulators: (1) a regional regulator who imposes capacities for the transportation

of hazmats on the network links and aims at minimizing the maximum risk; and (2) local regulators who choose the path to route the commodities from their origin to their destination while respecting the capacity constraints, and with the aim to minimize the total risk. The authors propose a bi-level formulation and solve it using a reformulation into a single level MIP.

To the best of our knowledge, Paredes-Belmar *et al.* (2017) is the first work to deal with a VRP with multiple commodities in the context of hazmat transportation. A set of hazmats must be collected by a fleet of homogeneous capacitated vehicles starting and ending their route at a single depot. Each hazmat represents a commodity and is associated with a level of risk. All the commodities are compatible and can be mixed inside the same vehicle. One or several commodities must be collected at each customer. The collection at a customer may be split, but a single commodity has to be collected at once. When the vehicle passes through an arc of the network (a street) it imposes a risk equals to the highest risk of the commodities currently loaded in the vehicle. Hence, each time a vehicle collects hazmats from a customer, the risk associated with the vehicle remains the same or increases. The objective is to minimize a weighted sum of costs and risks. Note that the sequence of the commodities collected in the vehicle impacts the risks. Paredes-Belmar *et al.* (2017) propose a MIP formulation based on an auxiliary graph of the network and use a commercial solver.

2.4.2 Transportation of multiple commodities in disaster relief

Golden *et al.* (2014) define disaster as an extraordinary event that occurs with or without limited forewarning and has devastating effects on the population (e.g., earthquakes, hurricanes, terrorist attacks, and industry disasters). The response phase considers the logistics operations to satisfy the urgent needs of a population just after a disaster. Unlike most of the VRPs, the goal of the VRPs in disaster relief is to provide the best possible supply regardless of the expenses rather than maximize the profits or to meet customer demands at minimum cost (Golden *et al.*, 2014).

Routing problems in disaster response consider a set of supply points (suppliers) and a set of demand points (customers). Each supplier provides one or several commodities with limited quantities. The required commodities may be water, food,

medicine, clothes, machinery. Usually, demands are large, and it is necessary to split the delivery to a single customer. Since the commodities have different characteristics, several origins and destinations, they have to be explicitly considered.

Distribution of commodities in disaster relief is often considered in the literature as a multi-commodity network flow problem. Barbarosoğlu & Arda (2004) study a transportation problem for disaster response in case of earthquake. They consider several suppliers and customers and a set of commodities with multiple origins and destinations for each commodity. Some additional points are used for transshipment operations. Each arc of the network represents a route, and may be traversed using different transportation modes e.g. trucks and helicopters. Each commodity is compatible with one or several modes, and a change of mode of transportation for a commodity entails in a cost. Since it is difficult to predict a disaster, the authors consider uncertainty on the supply quantities and the capacities of the arcs of the network. The problem is consists in determining paths in the network in order to route each commodity from its origin to destination while satisfying the capacity constraints. The objective is to minimize the sum of transportation costs plus shortage costs for the unsatisfied demands. They propose a two-stage stochastic programming model to solve this problem. Özdamar *et al.* (2004) study a similar problem in a deterministic context, but over a planning horizon. They explicitly consider the flow of vehicles at each period. This is not a classical routing problem since the vehicles do not go back to a depot. The authors propose a MIP formulation for this problem. Yi & Özdamar (2007) extend the work of Özdamar *et al.* (2004) by considering the problem of locating emergency medical centers to evacuate wounded people that have to be picked up at some gathering points. Yi & Kumar (2007) propose a two-phase heuristic to solve the problem defined by Özdamar *et al.* (2004). At the first stage, they use an ant colony algorithm to find vehicle paths for each period while the second phase consists in the distribution of the commodities using the set of vehicle paths. This second stage is modeled as a multi-commodity flow problem. Tzeng *et al.* (2007) dynamically choose the number of relief items to be transported from depots to demand points such that three objectives are achieved: minimum total cost, minimum travel time, and maximum demand satisfaction.

Abounacer *et al.* (2014) consider a location-transportation problem for disaster response, which involves three conflicting objectives: minimize the total trans-

portation duration of required products from the distribution centers to the demand points, minimize the number of agents needed to open and operate the selected distribution centers, and minimize the non-covered demand for all demand points within the affected area. They propose an exact method (epsilon-constraint) to solve the problem. Ghasemi *et al.* (2019) propose a multi-objective multi-commodity multi-period multi-vehicle location-allocation MIP model for the response phase (i.e. the logistics operations to satisfy the urgent needs of a population just after a disaster) of an earthquake. The objective is to minimize the total cost of the location-allocation of facilities and minimize the amount of the shortage of relief commodities.

2.5 Conclusions and directions for further research

In this chapter, we have reviewed the literature on vehicle routing problems with multiple commodities. We have focused on the routing problems where multiple commodities have to be explicitly considered. The problems have been classified into three categories: the VRP where the vehicle has multiple compartments to transport incompatible commodities, extensions of VRPs where multiple commodities are explicitly considered, and applications where the transportation problems usually consider multiple commodities. For each problem, we determined what is the motivation to explicitly consider multiple commodities. We also reviewed the main applications related to the multiple commodities VRP.

From this literature review, it can be concluded that multiple commodities have to be considered for the following reasons.

- Commodities are incompatible, i.e. they cannot be mixed. However, it is possible to transport them in the same vehicle. A solution is to use specific vehicles with multiple compartments such that a compartment contains a single commodity. Another solution is when a vehicle performs multiple trips. In this case, two incompatible commodities cannot be loaded in the same vehicle in the same trip, but they can be loaded in the same vehicle on different trips.

  Solution methods that face these problems need to determine the size of each compartment or the scheduling of the trips assigned to the same vehicle.

- Commodities are compatible with each other. Hence they share a common resource. However, they have different characteristics. For example in the

traveling purchaser problem, commodities have different prices, volumes and supply quantities; or in the inventory routing problem, they have different inventory costs and customer consumption rates.

Solution methods that face these problems not only need to determine minimum cost tours to visit a set of locations but also need to determine the quantity of each commodity to collect or deliver at each location.

- Each commodity has to be transported from some origin locations (suppliers) to destination locations (customers). When delivering a demand location, we have to ensure that the amount of each required commodity has been loaded inside the vehicle. The related problems are pickup and delivery TSP, location routing problems, or routing problems related to transportation of hazardous materials and transportation for disaster relief.

  Solution approaches that deal with these problems need to take into account precedence constraints imposed by the pairing of pickup and delivery locations.

- The delivery of a customer can be split, but the delivery to an individual commodity cannot be split. The related problem is the Commodity constrained Split Delivery VRP.

  Here, solution approaches need to determine whether or not to split a delivery resulting in methods that need to deal with both customers and customer-commodities.

There are many applications for the transportation of multiple commodities. The main ones are the following.

- Transportation of petroleum, fuel, gas. The different commodities represent different products, e.g. different types of fuel or gas. The transportation of these products is considered in the Multi-Compartment VRP where the products cannot be mixed in the same compartment. However, it is also considered in the Multi-Commodity IRP and in the transportation of hazardous materials.

- Transportation of perishable food. Commodities represent different kinds of food. Some problems consider Multi-Compartment VRP since the foods cannot be mixed because of different quality, or because they have to be transported at different temperatures, or it is imposed by regulations. Transporting

perishable food in the same vehicles is interesting since it permits to have more frequent deliveries and to reduce the storage levels at the customers. That is why this aspect is studied for example in Multi-Commodity IRP.

- Transportation of waste and livestock are usually studied in the Multi-Compartment VRP since the different types of waster or livestock cannot be mixed in the same compartment.

- Transportation of products in case of disaster or war. In these applications, the commodities may be food, water, medicine, and people. It is interesting to transport all these commodities together in order to fully use the capacity of the vehicles and to ensure short delivery times. It can be noticed that in this case the products to transport are heterogenous.

- Transportation of the same type of product with different customization. This is typically the case when the products are compatible in the same vehicle, but they have different origins and destinations, or different characteristics. Some examples we have reviewed deal with bikes (with different configurations), money (bills and coins), wood (beam, plank, door and window frames).

Moreover, we also found that an interesting feature for several problems is related to the fact to split or not the deliveries. In multi-commodity VRP, there are three cases: (1) no split: all the commodities have to be delivered at once; (2) constrained split: the delivery can be split, but each commodity has to be delivered at once by a single vehicle; (3) full split: the deliveries can be split in any way. In the case with constrained split, the solving methods usually directly deal with the individual commodities, by replicating the same customer as many times as the number of required commodities. Some solving methods deal with both representations of the solution: customers and commodities.

Considering multiple commodities in VRPs implies to increase the number of variables and constraints in the MIP formulations, and the problems are more challenging to solve than the ones with a single commodity. Some solving methods decompose the solving into two phases: the routing problem is solved in a first phase, and then the commodities to transport on each route are decided on the second phase.

The main perspective for VRPs with multiple commodities is to develop efficient solution methods. One interesting perspective consists in avoiding to replicate the

customers when the delivery of an individual commodity cannot be split. Another one relies in more integrated approaches to solve large size instances involving many commodities.

Chapter 3

Adaptive Large Neighborhood Search for the Commodity constrained Split Delivery VRP

Contents

Abstract: This paper addresses the Commodity constrained Split Delivery Vehicle Routing Problem (C-SDVRP) where customers require multiple commodities. This problem arises when customers accept to be delivered separately. All commodities can be mixed in a vehicle as long as the vehicle capacity is satisfied. Multiple visits to a customer are allowed, but a given commodity must be delivered in one delivery.

In this paper, we propose a heuristic based on the Adaptive Large Neighborhood Search (ALNS) to solve the C-SDVRP, with the objective of efficiently tackling medium and large sized instances. We take into account the distinctive features of the C-SDVRP and adapt several local search moves to improve a solution. Moreover, a Mathematical Programming based Operator (MPO) that reassigns commodities to routes is used to improve a new global best solution.

Computational experiments have been performed on benchmark instances from the literature. The results assess the efficiency of the algorithm, which can provide a large number of new best-known solutions in short computational times.

Keywords: vehicle routing problem; multiple commodities; adaptive large neighborhood search; local search.

3.1 Introduction

The Vehicle Routing Problem (VRP) and its variants have been widely studied in the literature (Toth & Vigo, 2014). In the VRP with multiple commodities, capacitated vehicles are used to deliver a set of commodities and meet the demands of customers. Four different strategies to deliver a set of commodities to customers were presented

by Archetti *et al.* (2014): *separate routing*, *mixed routing*, *split delivery mixed routing* and *commodity constrained split delivery mixed routing*.

In the *separate routing* strategy, a specific set of vehicles is dedicated to each commodity, and any commodity has to be delivered to any customer by a single vehicle visit. In this case, considering each commodity separately is a classical capacitated VRP (CVRP). A customer is visited as many times as the number of commodities that he/she requires. In other words, if a customer needs multiple commodities, this customer needs to be served several times, even if all the commodities could be delivered by only one vehicle.

In the *mixed routing* strategy any set of commodities can be mixed in the same vehicle, and all customers must be delivered at once. If a customer requires one or more commodities, all of them are delivered by a single vehicle in a single visit. Here again, the problem that arises corresponds to a single CVRP. In this delivery strategy, when the remaining capacity of a vehicle is not sufficient to deliver all the demand of a given customer, it is wasted with a possible increase in transportation costs.

In the *split delivery mixed routing* strategy, any set of commodities can be mixed in the same vehicle. The commodities can be split in any possible way. Moreover, a commodity can be delivered to a customer by several vehicles. The problem that arises corresponds to the split delivery VRP (SDVRP) that was introduced by Dror & Trudeau (1989) and Dror & Trudeau (1990). For an extensive review of SDVRP, the interested reader is referred to the survey by Archetti & Speranza (2012). In the split delivery mixed routing strategy, a customer can be visited several times if this is beneficial, even if this customer requires only one commodity. This strategy minimizes transportation costs but may cause inconvenience to customers (Archetti *et al.*, 2008). For example, if one commodity is delivered by several vehicles, the handling time for the customer may increase significantly.

The *commodity constrained split delivery mixed routing* (C-SDVRP) is a strategy recently proposed by Archetti *et al.* (2014) to deliver multiple commodities. In the C-SDVRP, any set of commodities can be mixed in the same vehicle. The demand of a customer can be split, but only as many times as the required commodities and when a commodity is delivered to a customer, the entire required amount is handed over. As a consequence, one commodity can be delivered by only one vehicle.

3. ADAPTIVE LARGE NEIGHBORHOOD SEARCH FOR THE COMMODITY CONSTRAINED SPLIT DELIVERY VRP

The C-SDVRP shares similarities with the SDVRP. However, a significant difference is that the demand of a customer cannot be arbitrarily split since a commodity has to be delivered by the same vehicle. Thus, only splitting a request of a customer according to different commodities is allowed. Considering the convenience of customers and transportation costs, the C-SDVRP is more interesting than the other three strategies to deliver multiple commodities. This problem arises in several real-life situations, for instance in the delivery of groups of products: dairy products, fresh fruits, or vegetables to supermarkets, catering services or restaurants. Each group of products represents a commodity. These commodities can be mixed in the same vehicle. For the customer, it is acceptable to have more than one delivery, but splitting the delivery of a specific commodity (a group of products) is not practical at all. Few commodities are considered. The number of deliveries for a customer is therefore acceptable. Moreover, considering to split deliveries is beneficial for the entire logistic system since it reduces transportation costs.

Despite its practical relevance, the C-SDVRP has received very little attention. The C-SDVRP was first introduced by Archetti *et al.* (2014). A branch-and-cut algorithm was proposed by the authors and is able to solve to optimality 25 out of 64 small instances (15 customers) within 30 minutes. Archetti *et al.* (2014) also proposed a heuristic method to tackle this problem. The heuristic consists in making copies of each customer (one for each required commodity) and uses an open-access injection-ejection algorithm for the CVRP. This solving method is simple but does not seem to be very efficient. Indeed, customer replicas share the same location, so the resulting capacitated VRP has many equivalent solutions. We use an example to highlight this point.

Figure 3.1 shows an example where customers require multiple commodities. A square indicates the location of the depot and circles represent the locations of the three clients. The number of commodities is three. The number on each edge corresponds to the associated travel cost, and the numbers in the dotted ellipses are the demands for each commodity. To increase readability, a different colored ellipse represents each commodity. For instance, for each customer, the number in the red ellipse is the demand of commodity one required by this customer. Each vehicle has a capacity of $Q = 10$ units.

According to the heuristic used by Archetti *et al.* (2014), the C-SDVRP is solved as a CVRP where the set of customers contains all the customer replicas. Figure

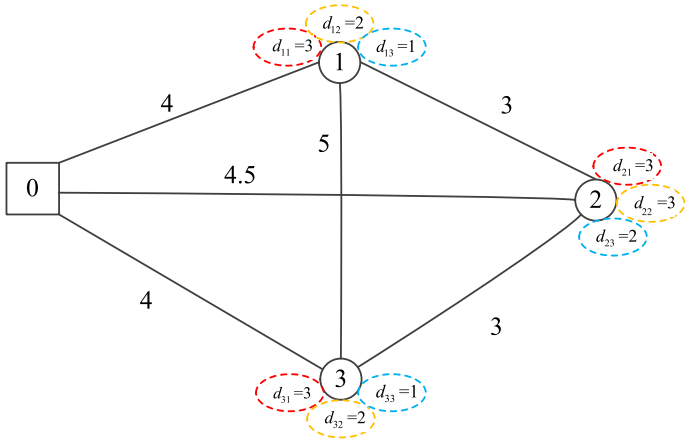
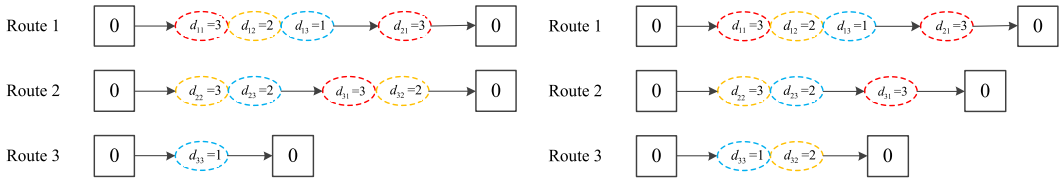


Fig. 3.1. An example from Archetti *et al.* (2014).

3.2 shows two solutions of this example.



(a) Solution 1. (b) Solution 2.

Fig. 3.2. Two solutions of the instance in Figure 3.1.

Solution 1 is composed of three routes to serve all of the demand. Route 1 serves the whole demand for customer 1 (6 units) and commodity 1 of customer 2 (3 units) for a total of 9 units. Route 2 delivers the remaining of the demand of customer 2 (5 units) and commodities 1 and 2 (5 units) of customer 3. The remaining demand for customer 3 (1 unit) is delivered in route 3. The three routes cost 11.5, 11.5 and 8 respectively. The total cost is then 31.

Solution 2 is also composed of three routes. Route 1 is the same as in solution 1. Route 2 delivers the remaining of the demand of customer 2 (5 units) and commodity 1 (3 units) of customer 3. The remaining demand for customer 3 (3 units) is delivered in route 3. The three routes cost 11.5, 11.5 and 8 respectively, for a total cost of 31.

These two solutions are different. However, in essence, the two solutions are equivalent in that customers' replicas share the same location. Taking this specific feature of the problem into account to avoid exploring many equivalent solutions is a real challenge.

3. ADAPTIVE LARGE NEIGHBORHOOD SEARCH FOR THE COMMODITY CONSTRAINED SPLIT DELIVERY VRP

In a more recent work Archetti *et al.* (2015) proposed an extended formulation for the C-SDVRP and developed a branch-price-and-cut algorithm. This algorithm has limitations for solving large-scale instances: optimal solutions were obtained with up to 40 customers and 3 commodities per customer within 2 hours. We are not aware of any other study in the literature solving the C-SDVRP.

This paper aims at proposing an efficient heuristic to tackle medium and large sized C-SDVRP instances. To this end, we propose an Adaptive Large Neighborhood Search (ALNS) heuristic taking into account the specialty of the C-SDVRP. ALNS is an efficient metaheuristic proposed by Ropke & Pisinger (2006) and extends the Large Neighborhood Search (LNS) heuristic (Shaw, 1997, 1998) by allowing multiple destroy and repair methods to be used within the same search. Recently, ALNS has been successfully applied to the capacitated VRP (Sze *et al.*, 2016) and to many variants of the VRP (Azi *et al.*, 2014; François *et al.*, 2016; Masson *et al.*, 2013; Sze *et al.*, 2017).

The contributions of this paper are as follows. First, a new heuristic method for the C-SDVRP is proposed. As mentioned earlier, if a customer is replicated as many time as the required commodities, many equivalent solutions exist. Hence, to avoid this pitfall, the proposed method explicitly takes this feature into account. To improve a solution, we adapt existing local search (LS) operators to deal with a customer as a whole (i.e., with the whole demand he/she requires) or only as a part (i.e., with a single commodity he/she requires). Second, in order to further improve the quality of a new and better encountered solution, a mixed integer programming (MIP) based operator is developed. Finally, we provide a large number of new best-known solutions for medium and large sized C-SDVRP instances up to 100 customers and 3 commodities per customer within about 10 minutes of computing time.

The rest of this paper is organized as follows. Section 3.2 defines the C-SDVRP. The proposed algorithm for the C-SDVRP is described in Section 3.3. Section 4.5 reports the computational results. Section 3.5 concludes the paper and suggests new directions for future research.

3.2 Problem definition

The C-SDVRP is defined on a directed graph $G = (\mathcal{V}, \mathcal{A})$ in which $\mathcal{V} = \{0\} \bigcup \mathcal{V}_C$ is the set of vertices, and $\mathcal{A}$ is the set of arcs. More precisely, $\mathcal{V}_C = \{1, ..., n\}$ represents the set of customer vertices, and 0 is the depot. A cost c_{ij} is associated with each arc $(i, j) \in \mathcal{A}$ and represents the non-negative cost of travel from i to j. Let $\mathcal{M} = \{1, ..., M\}$ be the set of available commodities. Each customer $i \in \mathcal{V}_C$ requires a quantity $d_{im} \geq 0$ of commodity $m \in \mathcal{M}$. Note that a customer $i \in \mathcal{V}_C$ may request any subset of commodities, namely d_{im} may be equal to zero for some $m \in \mathcal{M}$.

A fleet of identical vehicles with capacity Q is based at the depot and is able to deliver any subset of commodities.

The problem is to find a solution that minimizes the total transportation cost, and that involves two related decisions such as finding a set of vehicle routes that serve all customers and selecting the commodities that are delivered by a vehicle route to each customer. Moreover, each solution must be such that:

1. each route starts and ends at the depot;
2. the total quantity of commodities delivered by each vehicle does not exceed the vehicle capacity;
3. each commodity requested by each customer must be delivered by a single vehicle;
4. the demands of all customers need to be satisfied.

We use the example presented in Figure 3.1 to illustrate an optimal solution of a C-SDVRP instance. The solution is provided in Figure 3.3. In the C-SDVRP case, two vehicle routes are required to deliver all the commodities required by the customers. One route (black line) delivers all the commodities of customer 1 (6 units) and delivers commodities 1 and 3 of customer 3. The cost of this route is 13. The other route (purple line) delivers all the commodities of customer 2 (8 units) and commodity 2 of customer 3 (2 units). The cost of this route is 11.5. The total cost of the solution is 24.5. For the solutions obtained using the other three delivery strategies (separate routing, mixed routing and split delivery mixed routing), the interested reader is referred to Archetti *et al.* (2014).

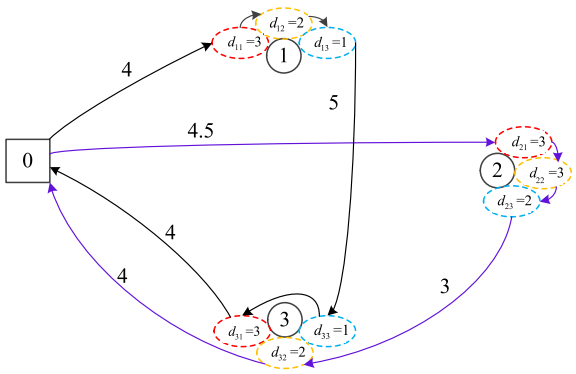


Fig. 3.3. An optimal solution for the C-SDVRP instance proposed in Figure 3.1.

3.3 Adaptive Large Neighborhood Search

In order to tackle the C-SDVRP for medium and large instances, we propose a heuristic method based on the ALNS framework of Ropke & Pisinger (2006). We make use of local search, and we develop a mathematical programming based operator to improve the quality of solutions.

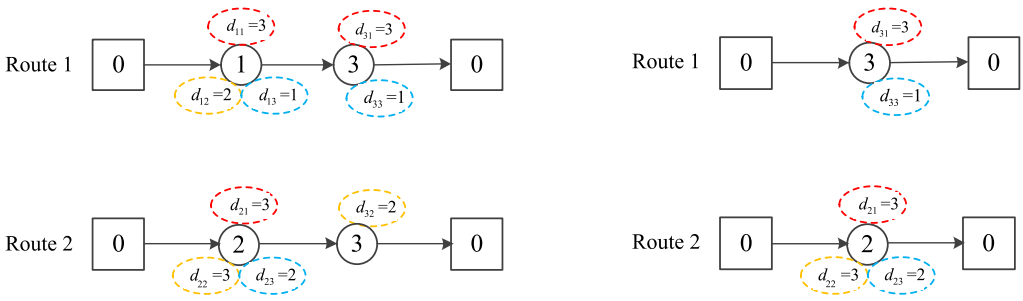
Due to the characteristics of the problem under study, one node can be duplicated as many times as the number of commodities required by the associated customer (Archetti *et al.*, 2014). With each duplicated node, we then associate the demand of the customer for the corresponding commodity. However, the simple duplication of customers without further consideration of the customer location can produce several equivalent solutions as illustrated previously in the paper. To enhance the performance of the algorithm, we explicitly consider customer replications for each commodity *and* the customer as a single entity associated with its total demand. To this intent and for the sake of clarity, in the following, we will call the duplicated nodes *customer-commodity*, and we will use the term *customer* to refer to the customer associated with the total demand.

We represent a solution of C-SDVRP as the set of routes needed to serve all customers. In order to take into account the specific features of the C-SDVRP, a route can be represented by (1) a sequence of *customers*, or by (2) a sequence of *customer-commodities*. Note that, because a customer can be delivered by several vehicles, in the representation with customers, it is possible that a customer appears in several routes (with different commodities).

The second representation gives more flexibility but increases in complexity. To clarify this, we consider the case of removing a customer from a solution. In the

first case, when removing a customer from a route, the customer associated with all the commodities delivered by the route is removed. In the second case, it is possible to remove only one commodity. We illustrate the two representations in Figures 3.4 and 3.5. We use the example presented in Figure 3.1. Figure 3.4 shows a set of routes represented as two sequences of customers. If customer 1 is removed from route 1, then customer 1 with all the three commodities is removed. If customer 3 in route 2 is removed, then customer 3 with commodity 2 is removed. Once a customer is removed, the remaining capacity of this route increases and the cost decreases. Note that even if customer 3 has been removed from route 2, he is still present in route 1.

Figure 3.5 shows a set of routes represented as two sequences of *customer-commodities*. In order to better understand the feature of C-SDVRP, we hide the circle which represents the customer. In fact, the two routes imply the same solution as shown in Figure 3.4. If commodity 2 of customer 3 in route 2 is removed, then the remaining capacity of this route increases and the cost decreases. However, if one commodity (like commodity 1) of customer 1 is removed, then the remaining capacity of this route increases but the cost of this route does not change.

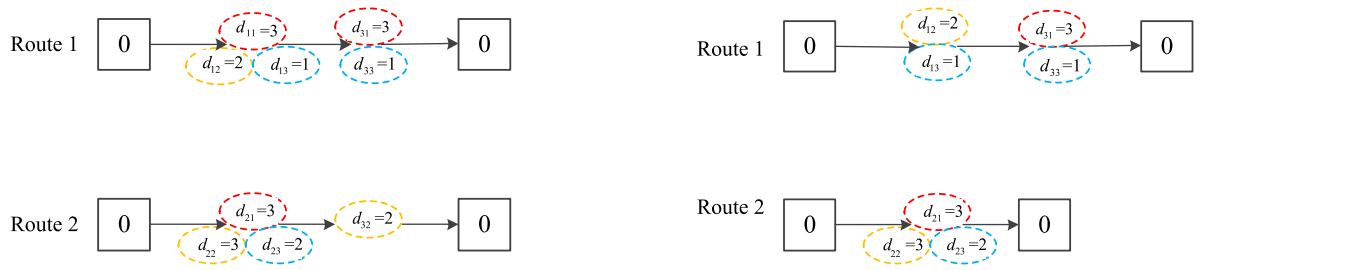


(a) Initial solution.

(b) Solution after removing customer 1 from route 1 and customer 3 from route 2.

Fig. 3.4. Two sequences of customers.

As mentioned above, in this paper we present operators for *customers* and *customer-commodities* that work based on the specific need. It is important to note that to deal with both *customers* and *customer-commodities*, each route in the solution has two concurrent representations. In the first representation, each route contains a sequence of *customers*, and a set of commodities is associated with each customer (see Figure 3.4 for an example). In the second representation, each route contains a sequence of *customer-commodities* (see Figure 3.5 for an example).



(a) Initial solution.

(b) Solution after removing commodity 1 of customer 1 from route 1 and commodity 2 of customer 3 from route 2.

Fig. 3.5. Two sequences of *customer-commodities*.

When using an operator, the corresponding representation (customers or *customer-commodities*) is used. The following considerations are then taken into account when translating one representation to the other:

- In the second representation with *customer-commodities*, when a route contains several commodities of the same customer, it is always optimal to group them (since there is zero cost to travel between these *customer-commodities*). Thus, in the first representation, a customer will not appear twice on the same route.

- When dealing with the second representation with *customer-commodities*, it is possible that a customer appears in different routes (with different commodities). Hence, it is possible in the first representation with customers that a customer appears in several routes (e.g., customer 3 in Figure 3.4(a)).

- When dealing with the first representation with customers, moving a customer means to move that customer with the associated commodities in the current route. These commodities may be a subset of the commodities required by this customer.

3.3.1 General framework

The basic idea of ALNS is to improve the current solution by destroying it and rebuilding it. It relies on a set of removal and insertion heuristics which iteratively destroy and repair solutions. The removal and insertion heuristics are selected using a roulette wheel mechanism. The probability of selecting a heuristic is dynamically

influenced by its performance in past iterations (Pisinger & Ropke, 2010). A sketch of the method is outlined in Algorithm 1.

In Algorithm 1, s_{best} represents the best solution found during the search, while s is the current solution at the beginning of an iteration. The cost of a solution s is denoted by $f(s)$.

A removal heuristic h_{rem} and an insertion heuristic h_{ins} are applied to the current solution s. We indicate by s_{rem} and s_{ins} the intermediate solutions obtained after applying h_{rem} and h_{ins} respectively.

h_{rem} removes and h_{ins} inserts ρ customers or *customer-commodities*, where ρ is a parameter that varies between ρ^{min} and ρ^{max}. We adapt the strategy proposed by François *et al.* (2016) to set the value of ρ: we slightly *destroy* the current solution when a new solution has just been accepted (small values of ρ), and we increase the value of ρ proportionally to the number of iterations without improvements. The probabilities of selecting a removal or an insertion heuristic are dynamically adjusted during the search (Section 3.3.8).

Once the heuristics h_{rem} and h_{ins} have been applied, the solution obtained s_{ins} is possibly improved by applying a local search. The resulting solution is denoted by s' (Section 3.3.3).

We allow the insertion heuristics to propose solutions that violate the capacity constraint. This is done with the aim of reducing the number of routes in the solution. Infeasibility is then penalized in the objective function by adding a factor proportional to the violation. Details on the penalization of the load violation are given in Section 3.3.9. We then try to recover feasibility by applying the local search.

Whenever a new best solution is obtained, a Mathematical Programming based Operator (MPO) is applied to further improve the new best solution (Section 3.3.7). This can be seen as an intensification phase of the algorithm.

The new solution s' is then subject to an acceptance rule. If accepted, the new solution becomes the current solution. Otherwise, the current solution does not change. This is repeated until a stopping criterion is met and the best solution found s_{best} is returned.

In the following, each component of this algorithm is explained in detail.

Algorithm 1 Adaptive Large Neighborhood Search framework.

$s_{init} \leftarrow$generate an initial feasible solution using split procedure and LS
$\rho \leftarrow \rho^{min}$, $s \leftarrow s_{init}$, $s_{best} \leftarrow s_{init}$
repeat
    Roulette wheel: select a removal heuristic h_{rem} and an insertion heuristic h_{ins}
    Destroy: $s_{rem} \leftarrow$ remove ρ customers (or *customer-commodities*) from s applying h_{rem}
    Repair: $s_{ins} \leftarrow$ insert removed *customer-commodities* into s_{rem} applying h_{ins}
    Improve: $s' \leftarrow$ improve solution s_{ins} with local search (Algorithm 2)
    if $f(s') < f(s_{best})$ **then**
        $s_{best} \leftarrow$ improve solution s' with MPO (Section 3.3.7)
        $s' \leftarrow s_{best}$
    end if
    if accept(s', s) **then**
        $s \leftarrow s'$, $\rho \leftarrow \rho^{min}$
    else
        $\rho \leftarrow \rho + 1$, or ρ^{min} if $\rho = \rho^{max}$
    end if
    Update roulette wheel
until stopping criterion is met
return s_{best}

3.3.2 Initial solution

Let n_{cc} be the number of *customer-commodities*. A feasible initial solution is constructed as follows. First, we randomly determine a sequence of *customer-commodities* and we construct a giant tour $S = (S_0, S_1, \ldots, S_{n_{cc}})$, where S_0 represents the depot and S_i is the i^{th} *customer-commodity* in the sequence. Then, we apply a split procedure to get a feasible solution. This procedure is inspired by the works of **?** and Prins (2004). It works on an auxiliary graph $\mathcal{H} = (\mathcal{X}, \mathcal{A}_{cc}, \mathcal{Z})$, where $\mathcal{X}$ contains $n_{cc} + 1$ nodes indexed from 0 to n_{cc}, where 0 is a dummy node and node $i, i > 0$, represents *customer-commodity* S_i. $\mathcal{A}_{cc}$ contains one arc (i, j), $i < j$, if a route serving *customer-commodities* S_{i+1} to S_j is feasible with respect to the capacity Q of the vehicle. The weight z_{ij} of arc (i, j) is equal to the cost of the trip that serves $(S_{i+1}, S_{i+2}, \ldots, S_j)$ in this same order. The optimal splitting of the giant tour S corresponds to a minimum cost path from 0 to n_{cc} in $\mathcal{H}$. Finally, this feasible solution is improved by applying local search (Algorithm 2).

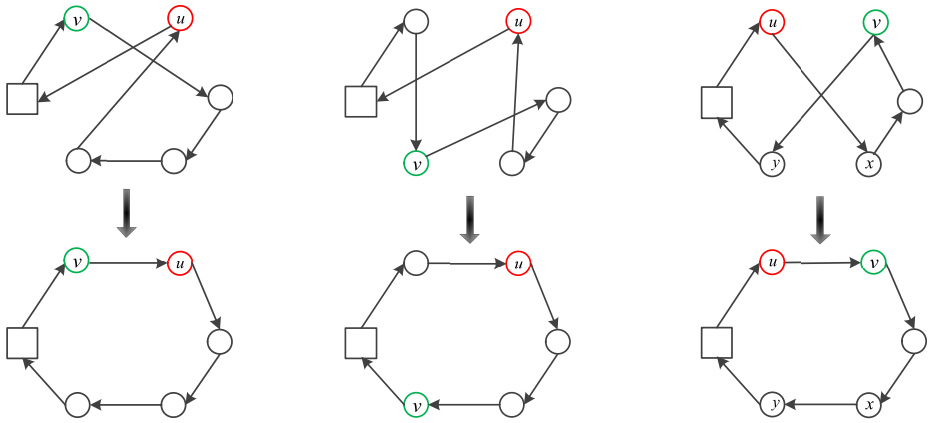
3.3.3 Local search

In order to improve a solution, a Local Search procedure (LS) is applied. LS is based on a set of classical operators that work on customers and on *customer-commodities*.

Let us first introduce some notation that will be useful in the remaining part of the section. Let u and v be two different nodes. They are associated with a customer or a *customer-commodity*, or one of them may be the depot depending on the operator. These nodes may belong to the same route or different routes. Let x and y be the successors of u and v in their respective routes. $R(u)$ denotes the route that visits node u.

Operators on customers

Here we present the operators that are defined for customers. These operators consider a customer together with all the commodities delivered to this customer in a given route. The different operators are illustrated in Figure 3.6 (where we only represent the intra-route cases).



(a) Insert customer. (b) Swap customers. (c) 2-opt on customers.

Fig. 3.6. Local search operators with customers.

Insert customer: this operator removes a customer u and inserts it after customer v.

Swap customers: this operator swaps the positions of customer u and customer v.

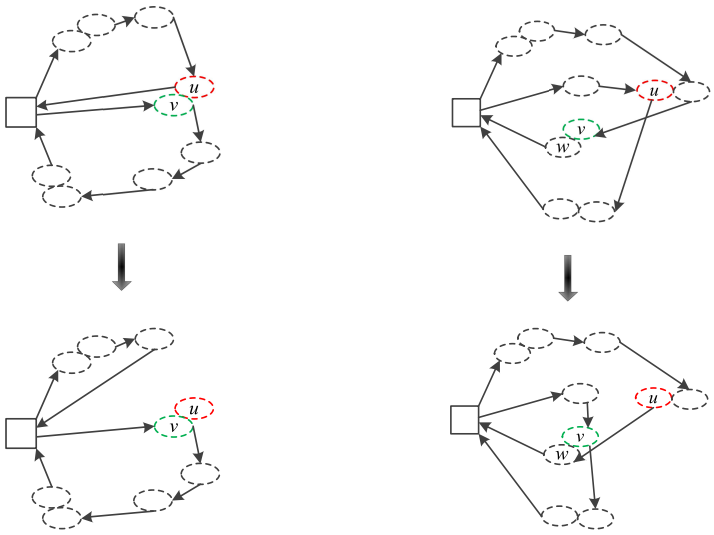
2-opt on customers: if $R(u) = R(v)$, this operator replaces (u, x) and (v, y) by (u, v) and (x, y).

Operators on *customer-commodities*

Here, we present the operators that are defined for *customer-commodities*. The different operators are illustrated in Figure 3.7.

Insert *customer-commodity*: this operator removes a *customer-commodity* u then inserts it after *customer-commodity* v.

Swap *customer-commodities*: this operator swaps *customer-commodity* u and *customer-commodity* v.



(a) Insert *customer-commodity*. (b) Swap *customer-commodities*.

Fig. 3.7. Local search operators with *customer-commodities*.

We can notice that the insert and swap operators for *customer-commodities* allow some moves that are not feasible when we consider only customers. In most cases, the insertion of a customer or of only one of the commodities to be delivered has the same cost. However, during the search, we allow the violation of the vehicle capacity, and the infeasibility is penalized (see Section 3.3.9). In an infeasible solution with some overcapacity in a route, it may be possible that inserting customers in other routes also leads to an infeasibility; while inserting only a *customer-commodity* decreases the infeasibility or leads to a feasible solution. Similarly, a swap of customers may not be feasible (or lead to an increase in the cost) due to the vehicle capacity restriction. At the opposite, when the commodities of some customers are delivered in several routes, swapping *customer-commodities* may be feasible with respect to the capacity and decreases the solution cost. As an example, in Figure 3.7, case (b), v and w

are two commodities for the same customer, while u is a commodity for another customer who requires two commodities. Swapping u with both v and w may not be possible because of vehicle capacity while swapping *customer-commodities* u and v decreases the cost of the solution.

Note that we do not consider a 2-opt operator based on *customer-commodities*. Since this operator works on elements of the same route, it is never beneficial to split apart *customer-commodities* associated to the same customer. It follows that its behavior would be the same as the operator 2-opt on customers.

LS is applied when the split procedure has generated an initial solution or when the removal and insertion heuristics have modified the solution. In the first case, we do not allow the solution to be infeasible with respect to the vehicle capacity. In the second case, the insertion heuristic can produce infeasible solutions. As a consequence, LS (and thus the operators) may have to address the infeasibility of the current solution. LS is depicted in Algorithm 2. Given a solution, first, four local search operators are applied: *insert customer*, *insert customer-commodity*, *swap customers* and *swap customer-commodities*. They are invoked iteratively until there is no further improvement. Then the operator *2-opt of customers* is applied. If it improves the solution, we reiterate with the first four operators. After exploring the neighbor defined by each operator, the move that improves the most is implemented. When all the local search operators fail, routes of the current solution are concatenated and split algorithm is applied again. This strategy is inspired by Prins (2009). If this provides a better solution, the whole procedure is repeated.

3.3.4 Removal heuristics

This section describes the set of removal heuristics we propose to destroy the current solution. Heuristics *Shaw removal* and *worst removal* use the representation of a solution with customers, while *random removal* can be applied to both representations with customers and *customer-commodities*. Another operator that randomly removes one route is also considered.

Shaw removal: this heuristic aims to remove a set of customers which are similar based on a specified criterion (e.g., location or demand). When customers with really different characteristics are removed, it is likely that each customer is

Algorithm 2 Local search procedure.

```
1: Let s be a (feasible or infeasible) solution
2: repeat
3:     repeat
4:         repeat
5:             Obtain s' by applying the insert customer operator; s = s'
6:             Obtain s' by applying the insert customer-commodity operator; s = s'
7:             Obtain s' by applying the swap customers operator; s = s'
8:             Obtain s' by applying the swap customer-commodities operator; s = s'
9:         until No improvement has been obtained
10:        Obtain s' by applying the 2-opt of customers operator; s = s'
11:    until No improvement has been obtained
12:    Obtain a giant tour by concatenation of trips
13:    Apply the split procedure
14: until No improvement has been obtained
```

then reinserted at the same position in the solution. Hence, by removing similar customers, Shaw removal aims to provide a different solution once an insertion heuristic has been applied.

Here, we define similarity between two customers as the distance between these two customers. The heuristic works as follows: a first customer i is randomly selected and removed. We then compute the similarity between customer i and the other customers (here, it is the distance), and sort the customers in a list L, according to the similarity with customer i. A determinism parameter p_d ($p_d \geq 1$) is used to have some randomness in the customer selection to be removed in L, with a higher probability for the firsts customers. The removed customer then plays the role of customer i and the procedure is repeated until ρ customers have been removed. The interested reader can find a detailed Shaw removal pseudocode in Ropke & Pisinger (2006).

Worst removal: this heuristic aims at removing the customers who induce a high cost in the solution. More precisely, at each iteration, we first calculate for each customer the cost decrease if it is removed from the solution. Then customers are sorted in decreasing order according to these values. As in Shaw removal, a determinism parameter p_d controls the randomization in the choice of the worst customer to remove.

Random removal: this removal heuristic randomly chooses ρ customers and removes them from the current solution. It can also be applied with *customer-*

commodities, by randomly removing ρ *customer-commodities*.

Route removal: in this removal heuristic, an entire route from the current solution is randomly selected, and all the *customer-commodities* on this route are removed.

3.3.5 Insertion heuristics

In this section, we describe the insertion heuristics implemented in the proposed ALNS algorithm. In this work, all insertion heuristics consider *customer-commodities*.

Greedy insertion: in this insertion heuristic, at each iteration, for each removed *customer-commodity* i, we first compute the *best insertion cost* Δf_i^1: cost of inserting the *customer-commodity* at its best position into the solution (i.e., the insertion that minimizes the increase of the cost of the solution). Then, the *customer-commodity* with the minimum insertion cost is selected to be inserted at its best position. After each iteration, the insertion costs of the remaining removed *customer-commodities* are recomputed. This process stops when all *customer-commodities* have been inserted.

Regret insertion: this insertion heuristic chooses, at each iteration, the removed *customer-commodity* which produces the biggest regret if it is not inserted at its best position at the current iteration. In regret-k heuristic, at each iteration, we first calculate, for each removed *customer-commodity* i, Δf_i^1 the cost of inserting i at its best position, and Δf_i^η ($\eta \in \{2; ...; k\}$ the cost of inserting i at its η^{th} best position. Then, for each *customer-commodity* i the regret value is computed as: $reg_i = \sum_{\eta=2}^{k} (\Delta f_i^\eta - \Delta f_i^1)$. This represents the extra cost if i is not inserted at the current iteration in its best position. Finally, the *customer-commodity* with highest regret value reg_i is inserted at its best position into the solution. The heuristic continues until all *customer-commodities* have been inserted. For regret-k insertion heuristics in this work, we have considered the values of k to be two and three.

Random insertion: this insertion heuristic randomly chooses a removed *customer-commodity*, and randomly chooses the insertion position in the solution.

Note that when inserting *customer-commodities*, violations of vehicle capacity are allowed and penalized in the cost function (see Section 3.3.9). However, we also impose a maximum capacity violation on each route. Hence, it is possible that a *customer-commodity* cannot be inserted in any route of the current solution. In this case, we create one additional route which includes this *customer-commodity*.

3.3.6 Acceptance and stopping criterion

When the removal, insertion and LS steps have been applied, we use a simulated annealing criterion to determine if the new solution obtained s' is accepted. However, a deterministic decision rule is applied in two cases. At each iteration of the algorithm, if s' has a lower cost than the current solution s ($f(s') < f(s)$), then s' is accepted. The solution s' is rejected if the costs $f(s')$ and $f(s)$ are equal. We reject solutions with the same cost in order to avoid working with equivalent solutions where some *customer-commodities* belonging to the same customer have been exchanged.

When $f(s') > f(s)$, then s' is accepted with probability $e^{-(f(s')-f(s))/T}$, where $T > 0$ is the temperature. As proposed in Ropke & Pisinger (2006), the initial temperature is set such that a solution which is $w\%$ worst than the initial solution s_{init} is accepted with a probability p_{accept}. More formally, T is chosen such that $e^{-(w \cdot f(s_{init}))/T} = p_{accept}$. Then, at each iteration of the ALNS, the temperature T is decreased using the formula $T = T \cdot \gamma$, where $\gamma \in [0, 1]$ is the cooling factor.

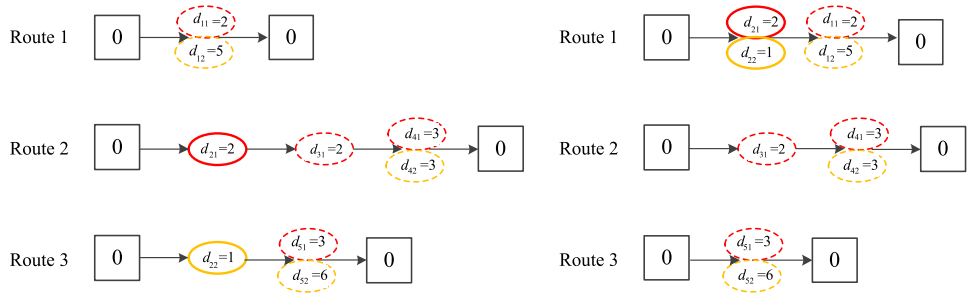
The stopping criterion for the whole procedure is a fixed number of ALNS iterations.

3.3.7 Mathematical Programming based Operator to reassign commodities

When a new best solution is identified, we intensify the search by applying a Mathematical Programming based Operator (MPO). The main purpose is to assign the visits to a specific customer among the solution routes in a different way by solving a capacitated facility location problem.

We use Figures 3.8 (a) and (b) to explain the idea behind MPO. In the example, we assume the vehicle capacity is 10 units. The number d_{im} in the ellipse reflects the demand of commodity m required by customer i. We focus on the commodities of customer 2: the ellipses with a solid line in Figure 3.8. We assume that Figure 3.8 (a) is a solution obtained after LS. Customer 2 has two commodities: the first is delivered on route 2, and the second is delivered on route 3. Inserting or swapping one of these two *customer-commodities* does not provide a better solution. However, we can notice that the deliveries to customer 2 can be reassigned to the first route

and the total cost decreases, as shown in Figure 3.8 (b). The reader can notice that the operators implemented in LS do not consider this kind of movements.



(a) Solution before using MPO. (b) Solution after using MPO.

Fig. 3.8. MPO for reassigning commodities.

Let us introduce the notation that we need to formally present MPO. First, we assume that customer i is considered. We indicate by:

- $\mathcal{M}_i$ the set of commodities required by customer i;
- s_i the solution obtained from the current solution by removing all the visits to customer i;
- $\mathcal{R}_i$ the set of routes in s_i;
- c_r^i the cost to insert (best insertion) customer i in route $r \in \mathcal{R}_i$;
- Q_r^i the remaining capacity in route $r \in \mathcal{R}_i$.

Then, we introduce the following binary decision variables:

$$x_{mr}^i = \begin{cases} 1 & \text{if the delivery of commodity } m \text{ of customer } i \text{ is assigned to route } r \in \mathcal{R}_i; \\ 0 & \text{otherwise.} \end{cases}$$

$$x_r^i = \begin{cases} 1 & \text{if at least the delivery of one commodity required by customer } i \text{ is assigned} \\ & \text{to route } r \in \mathcal{R}_i; \\ 0 & \text{otherwise.} \end{cases}$$

The mathematical program that we solve in order to apply MPO is the following:

$$(IP_{MPO})\ min \sum_{r \in \mathcal{R}_i} c_r^i x_r^i \tag{3.1}$$

$$\text{s.}t. \sum_{r \in \mathcal{R}_i} x_{mr}^i = 1, \qquad \forall\ m \in \mathcal{M}_i \tag{3.2}$$

$$\sum_{m \in \mathcal{M}_i} d_{im} x_{mr}^i \leq Q_r^i x_r^i, \qquad \forall\ r \in \mathcal{R}_i \tag{3.3}$$

$$x_{mr}^i \in \{0,1\}, \qquad \forall\ m \in \mathcal{M}_i, r \in \mathcal{R}_i \tag{3.4}$$

$$x_r^i \in \{0,1\}, \qquad \forall\ r \in \mathcal{R}_i \tag{3.5}$$

This mathematical program corresponds to a capacitated facility location problem, where only the costs related to the inclusion of a new route in the solution (the fixed cost) are taken into account. The objective function (3.1) aims to minimize the total insertion cost. Constraints (3.2) require that the delivery of each commodity (i.e., the delivery of each *customer-commodity*) must be assigned to one route. Constraints (3.3) impose that the total quantity of commodities assigned to a selected vehicle does not exceed its remaining capacity. Constraints (3.4)-(3.5) define the decision variables.

(IP_{MPO}) is solved for each $i \in \mathcal{V}_C$, but only the reassignment of visits associated with the highest cost reduction is effectively implemented.

3.3.8 Adaptive weight adjustment

A roulette wheel is used to give more or less importance to the removal and insertion heuristics to be used. The procedure implemented is based on the principles described in Ropke & Pisinger (2006) and François *et al.* (2016). This last variant includes a normalization process for the score. The main difference in our approach is that removal and insertion heuristics are not selected independently. A pair of removal and insertion heuristics $h_p = \{h_{rem}; h_{ins}\}$ is chosen in each iteration. We denote by $\mathcal{H}_p$ the set of pair of heuristics to be used. Each pair h_p is associated with a weight ω_{h_p}, a score π_{h_p}, and θ_{h_p} the number of times that pair h_p has been used. Initially, all pairs of heuristics have the same weight and $\sum_{p \in \mathcal{H}_p} \omega_{h_p} = 1$.

We define a segment as a fixed number of ALNS iterations in the proposed algorithm. During a segment, the weights of all pairs are kept constant. Before starting a new segment, for each pair $h_p \in \mathcal{H}_p$, the score π_{h_p} and the number of

times the pair is used θ_{h_p} are reset to 0. During a segment, each time a pair h_p is used, θ_{h_p} is increased by 1, and if the new solution s' is accepted after using pair h_p, the score π_{h_p} is updated according to: $\pi_{h_p} \leftarrow \pi_{h_p} + \sigma_\mu$, where σ_μ ($\mu \in \{1; 2; 3\}$) reflects different cases regarding the score change π_{h_p}. That is, the score π_{h_p} is increased by σ_1 when s' is a new best solution, or σ_2 when s' is a new improved solution ($f(s') < f(s)$) but not a new best solution, or σ_3 when s' is not an improved solution but has been accepted according to the simulated annealing criterion. The values σ_μ ($\sigma_\mu \in [0, 1]$) are normalized to satisfy $\sigma_1 + \sigma_2 + \sigma_3 = 1$.

At the end of each segment, we update all the weights of the pairs of heuristics based on the recorded scores. First, the score π_{h_p} is updated as: $\pi_{h_p} \leftarrow \frac{\pi_{h_p}}{\theta_{h_p}}$, where θ_{h_p} is the number of times that pair h_p was used in this segment. If $\theta_{h_p} = 0$, we set π_{h_p} to the same value as in the previous segment. Then, the scores of all pairs of heuristics are normalized:

$$\bar{\pi}_{h_p} = \frac{\pi_{h_p}}{\sum_{h \in \mathcal{H}_p} \pi_h}. \tag{3.6}$$

Let $\omega_{h_p,q}$ be the weight of pair h_p used in segment q, and $\lambda \in [0, 1]$ a factor which determines the change rate in the performance of the pair of heuristics. At the end of segment q, the weight of all pairs of heuristics h_p to be applied in segment $q + 1$ is updated as:

$$\omega_{h_p,q+1} = (1 - \lambda)\omega_{h_p,q} + \lambda\bar{\pi}_{h_p}. \tag{3.7}$$

3.3.9 Infeasibility penalization scheme

In our implementation of the ALNS, we allow some violations of the vehicle capacity in order to reduce the number of routes. Let K_{init} be the number of vehicles used in the initial solution s_{init}. Then, the vehicle capacity Q can be extended by an amount of $Q_{extra} = \frac{Q}{K_{init}}$. If a vehicle delivers more than Q units of products, we penalize the infeasibility by adding to the solution cost a term proportional to the load violation which is $\beta l(s)$, where $l(s)$ is the total load violation of the solution s and β is the penalty rate.

The penalty rate β is related to capacity violations. Initially, β is set equal to a minimum value β_{min} computed as

$$\beta_{min} = 10 \cdot \frac{f(s_{init})}{\sum_{i \in \mathcal{V}_C, m \in \mathcal{M}} d_{im}},$$

where $f(s_{init})$ is the cost of the initial solution obtained after applying the split procedure and LS. This ratio approximates an average transportation cost per unit of demand.

The penalty rate β is dynamically modified during the search since a high rate may eliminate infeasibility quickly, but may also prevent searching other promising regions. We keep track of the number of infeasible solutions obtained during the last consecutive I iterations of the ALNS algorithm. If I_{inf} infeasible solutions are obtained consecutively, the value of β is increased to 2β. Similarly, if I_{feas} feasible solutions are generated consecutively, the value of β is decreased to $\max\{\beta_{min}; \beta/2\}$.

3.4 Computational experiments

In this section, we present the results obtained by the proposed ALNS heuristic. The algorithm was implemented in C++ and ran on an Intel (R) Core(TM) i7-4600U, 2.10GHz, and 16GB of RAM.

We first describe in Section 3.4.1 the test instances on which we evaluate our algorithm. In Section 3.4.2 we report the values of the parameters that we set for the ALNS. Then, in Section 3.4.3 we perform a sensitivity analysis to determine which insertion and removal heuristics lead to the best performance of the proposed algorithm. A set of settings is tested on a subset of instances, and the best is used to perform all other computational tests. In Section 3.4.4, we analyze the effect of the number of iterations on the computational time and the solution quality. Then, Section 3.4.5 reports the results on the testbed. The effectiveness of the MPO is studied in Section 3.4.6, while in Section 3.4.7 we validate the effectiveness of integrating a local search in the large neighborhood search. We examine how the computational time of our method varies according to the instance size in Section 3.4.8. Finally, Section 3.4.9 provides managerial insights about the split customers that are delivered by more than one vehicle.

3.4.1 Instances

To assess the efficiency of our algorithm, we perform computational experiments on the benchmark instances proposed by Archetti *et al.* (2014). These instances are built from the R101 and C101 Solomon instances for the VRP with Time Windows (Solomon (1987)), from which the customer locations are kept. From instances R101

and C101, several instances for the C-SDVRP were generated by considering the n first customers, with $n \in \{15, 20, 40, 60, 80, 100\}$.

In the testbed, up to 3 commodities are taken into account. The number of commodities is indicated by M ($M \in \{2; 3\}$). Each commodity is required by a customer with a probability p of 0.6 (on average each customer needs 2 out of 3 commodities) or with probability p equal to 1 (each customer requires all commodities). The quantity of each commodity required by a customer varies within the intervals $\Delta = [1, 100]$ or $\Delta = [40, 60]$. A last parameter $\alpha \in \{1.1, 1.5, 2, 2.5\}$ is used to determine the vehicle capacity from the original one in Solomon instances. We indicate by $\mathcal{P} = (\mathcal{I}, M, p, \Delta, \alpha)$ (where $\mathcal{I} \in \{R101, C101\}$) the set of parameters used by Archetti *et al.* (2014) to generate instances.

When $n = 15$, 64 instances are available, one for each combination of parameters in set $\mathcal{P}$. These are referred as small instances. When $n \in \{20, 40, 60, 80\}$, 80 mid-size instances are available: 20 instances for each value of n. For the mid-size instances, the combination of parameters in $\mathcal{P}$ is restricted to $M = 3$, $\alpha = 1.5$ and $\Delta = [1, 100]$. Hence $\mathcal{I}$ and p can take two values each, leading four combinations of parameters. For each combination, five instances have been generated, leading to 20 instances in total. When $n = 100$, 320 large instances are available, that means five instances for each combination of parameters in set $\mathcal{P}$.

In the following sections, we present the parameter setting for our algorithm and the results obtained. We provide detailed results on each of the 464 instances in the Appendix A. Due to the high number of instances, we only report average results. In particular, we present results for each *group* of instances, where a group is defined by a quartet $(n, \mathcal{I}, M, p)$. Results for a group are averaged on the values of α and Δ.

We summarize all the notations used to present the results in Table 3.1.

3.4.2 ALNS parameters

In this section, we present the values of the parameters set in our ALNS algorithm. The probabilities to select the pairs of removal and insertion heuristics are updated after a number of iterations called a segment. In our implementation, we define a segment as 100 iterations.

To set the values of σ_1, σ_2, σ_3, preliminary experiments were carried out on instances with $n = 80$ customers for some combinations. The best results were obtained when $\sigma_1 = 0.7$, $\sigma_2 = 0.1$, $\sigma_3 = 0.2$. We set the reaction factor λ that

Table 3.1: Notations for computational results

Symbol	Meaning
C101/R101	Name of the original Solomon instance
n	Number of customers
M	Number of commodities
p	Probability that a customer requires a commodity
id	Instance id (5 mid and large sized instances are generated for each combination of parameters in $\mathcal{P}$)
Δ	Demand range of each customer for each commodity
α	Value for generating the vehicle capacity (see Archetti *et al.* (2015))
CC_m	Total number of customers that require commodity m
n_{cc}	Total number of *customer commodities*
$avg.n_{cc}$	Average total number of *customer commodities* in each group of instances
$nbIns$	Number of instances in each group
OPT	Optimal solution value from literature (when available)
BKS	Best known solution value from literature (when available)
$Cost$	Best solution cost found by the proposed algorithm
Δ_O	Percentage of improvement between Cost and OPT ($\Delta_O = 100 * (Cost - OPT)/OPT$)
Δ_B	Percentage of improvement between Cost and BKS ($\Delta_B = 100 * (Cost - BKS)/BKS$)
$\Delta_{O/B}$	To indicate that the value is Δ_O if an optimal value if available, and Δ_B otherwise
$avg.\Delta_O$	Average percentage of improvement between Cost and OPT ($\Delta_O = 100 * (Cost - OPT)/OPT$)
$avg.\Delta_B$	Average percentage of improvement between Cost and BKS ($\Delta_B = 100 * (Cost - BKS)/BKS$)
$min.\Delta_O$	Minimum percentage of improvement between Cost and OPT ($\Delta_O = 100 * (Cost - OPT)/OPT$)
$max.\Delta_B$	Maximum percentage of improvement between Cost and BKS ($\Delta_B = 100 * (Cost - BKS)/BKS$)
$t(s)$	CPU time in second
$avg.t(s)$	Average CPU time in seconds
$nbOPT$	Number of optimal solutions obtained
nbE	Number of solution values obtained equal to the best known
$nbNBK$	Number of new best known solutions obtained
nbR	Number of routes
$nbMPO$	Number of times that MPO is called for a instance
$nbMPOimp$	Number of times that MPO improves the new best solution during the search
$avg.nbMPO$	Average number of times that the MPO is called for each group of instances
$avg.nbMPOimp$	Average number of times that the MPO improves the current best solution for each group of instances
*	Indicates the solution value is optimal
$nbSplit$	Number of split customers (that are delivered by more than one vehicle)
$nb2\text{-}split$	Number of split customers delivered by exactly two vehicles
$nb3\text{-}split$	Number of split customers delivered by exactly three vehicles
$nbNearDepot$	Number of split customers that are close to the depot
$nbLargeDemand$	Number of split customers with a large demand
$nbCluster$	Number of split customers located inside a cluster of customers

appears in Equation (3.7) to 0.5 as proposed in Masson *et al.* (2013) (instead of $\lambda = 0.1$ as in Ropke & Pisinger (2006)) to ensure higher reactivity when performing fewer iterations. The determinism parameter p_d in Shaw removal and worst removal heuristics is equal to 6 as in Ropke & Pisinger (2006).

The acceptance of a new current solution is based on a simulated annealing criterion. The initial value of the temperature T is set so that a solution that is $w\%$ worse than the current solution is accepted with probability p_{accept} with $w = 0.35$ and $p_{accept} = 0.7$. In addition, we set the cooling factor γ to 0.999.

In order to destroy a solution, we need to determine the number ρ of customers (or *customer-commodities*) to be removed. We follow the scheme proposed by François *et al.* (2016): small moves are applied when a new solution has just been

accepted, while large moves are applied when no new solution has been accepted in the most recent iterations. The value of ρ evolves in the interval $[\rho^{min}, \rho^{max}]$. For small instances ($n = 15$), we set $\rho^{min} = N/2$, $\rho^{max} = N$, while for the other instances, we set $\rho^{min} = N/10$ and $\rho^{max} = N/4$. When the removal heuristic is customer-based, then N represents the number of customers ($N = n$). When the removal heuristics are defined for *customer-commodities*, N represents the number of *customer-commodities* ($N = n_{cc}$).

To update the penalization rate for capacity violations, we consider the number of infeasible and feasible solutions that are obtained consecutively. These values are respectively set to $I_{inf} = 50$ and $I_{feas} = 5$ after performing preliminary experiments.

3.4.3 Efficiency assessment for the removal and insertion heuristics

In Sections 3.3.4 and 3.3.5, we propose several removal and insertion heuristics to be used inside the ALNS framework. Before testing the proposed algorithm on the set of instances, we use a subset of instances in order to determine the removal and insertion heuristics that are used inside the ALNS framework. We also consider configurations with only one removal and one insertion heuristic, which corresponds to designing a Large Neighborhood Search (LNS). This permits to emphasize the efficiency of individual insertion and removal heuristics, and also to point out the benefit of the adaptive approach included in the design of the ALNS when choosing the removal and insertion heuristics.

This analysis is performed to tune the proposed algorithm. We only use the 20 instances with $n = 80$ customers. A summary of these experiments is reported in Table 3.2. The first column shows which configuration is considered (LNS or ALNS). The second column enumerates the configurations that we tested. Columns 3 to 7 indicate which removal heuristics are used in a specific configuration. Among them, 'RandC' and 'RandCC' represent the random removal heuristic considering customers and *customer-commodities*, respectively. Columns 8 to 11 indicate which insertion heuristics are used. In columns 12 to 14 we report the following average statistics for each configuration: the gap with the best-known solutions, the computational time in seconds, the number of new best-known solutions (see Table 3.1).

Table 3.2: LNS configurations compared to ALNS configurations.

		Removal heuristics					Insertion heuristics				Results		
	conf.	Shaw	RandC	RandCC	Worst	Route	Greedy	Regret-2	Regret-3	Rand	$avg.\Delta_B$	$avg.t(s)$	$nbNBK$
LNS	1	×						×			**-0.77**	521.18	**20**
	2		×					×			-0.72	438.34	19
	3			×				×			-0.54	276.76	17
	4				×			×			-0.23	714.33	13
	5					×		×			0.01	333.60	10
	6	×					×				-0.61	533.30	20
	7	×							×		**-0.78**	541.81	**19**
	8	×								×	-0.07	1106.18	11
	9		×							×	-0.10	1146.84	13
ALNS	10	×	×	×	×	×	×	×	×	×	**-0.71**	562.29	**20**
	11	×	×				×	×			-0.72	513.49	19
	12	×	×					×	×		-0.77	500.97	19
	13	×	×				×	×	×		**-0.79**	511.00	**20**
	14	×	×	×			×	×			-0.68	467.99	20
	15	×	×		×		×	×			-0.63	555.34	20
	16	×	×		×		×	×	×		-0.66	530.26	19
	17	×	×	×			×	×	×		-0.79	477.49	19

In the first five configurations, we study the influence of the removal heuristic. To this intent, the insertion heuristic is kept fixed. Using the Shaw removal heuristic, the LNS is able to improve all the best-known solutions.

Configurations 6 to 8 point out the impact of the insertion heuristic. We fixed the removal heuristic to the Shaw removal since it provided the best results in the previous experiments. These tests show that all insertion heuristics perform well except the random insertion. To further evaluate the behavior of the random insertion heuristic we consider configuration 9 where we modify the removal heuristic. It can be observed that the results do not improve. Among the LNS configurations, combining the Shaw removal with the regret-3 insertion provides the best results.

In the second part of Table 3.2, we consider different configurations for the ALNS framework, with several removal and insertion heuristics. Configuration 10 shows the performance of the ALNS algorithm using all proposed heuristics. In other configurations, only a subset of them is invoked in the ALNS. As we can see, using all the proposed removal and insertion heuristics does not provide the best results. The best configuration is configuration number 13: an ALNS with two removal heuristics: Shaw and random removal of customers, and three insertion heuristics: greedy, regret-2 and regret-3. In the following sections, all reported computational experiments have been conducted using this configuration.

3.4.4 Analysis with respect to the number of iterations

We examine the impact of the number of iterations for different instance sizes. Preliminary computational experiments showed that the algorithm converges after a certain number of iterations. Thus, we aim to determine the number of iterations that would be a good compromise between the solution quality and the computational time.

For small instances and medium instances with 20 customers (mid-20), we run our algorithm with the number of iteration *iter* limited to 100, 1000, 3000 and 5000. We compare the results with those reported in Archetti *et al.* (2014) and Archetti *et al.* (2015). In Table 3.3 and Table 3.4 we report average statistics for different values of *iter*: the gap with respect to the optimal values, the computational time in seconds, the number of optimal solutions obtained (see Table 3.1).

Results over the testbed are indicated in bold. Detailed results are provided in Appendix A.

Table 3.3: Impact of the number of iterations on small instances.

instances						ALNS (100 iterations)			ALNS (1000 iterations)		
	n	M	p	$avg.n_{cc}$	$nbIns$	$avg.\Delta_O$	$avg.t(s)$	$nbOPT$	$avg.\Delta_O$	$avg.t(s)$	$nbOPT$
C101	15	2	0.6	22	8	0.00	0.93	8	0.00	2.82	8
C101	15	2	1	30	8	0.01	1.64	7	0.00	4.45	8
C101	15	3	0.6	28	8	0.00	1.13	8	0.00	3.92	8
C101	15	3	1	45	8	0.47	3.72	4	0.33	9.81	7
R101	15	2	0.6	22	8	0.00	1.07	8	0.00	2.89	8
R101	15	2	1	30	8	0.00	2.10	8	0.00	4.75	8
R101	15	3	0.6	28	8	0.00	1.26	8	0.00	3.88	8
R101	15	3	1	45	8	0.35	3.36	6	0.00	8.63	8
		total			64	**0.10**	**1.90**	**57**	**0.04**	**5.14**	**63**
instances						ALNS (3000 iterations)			ALNS (5000 iterations)		
C101	15	2	0.6	22	8	0.00	7.24	8	0.00	11.65	8
C101	15	2	1	30	8	0.00	10.63	8	0.00	16.86	8
C101	15	3	0.6	28	8	0.00	10.40	8	0.00	17.12	8
C101	15	3	1	45	8	0.21	22.26	7	0.21	34.62	7
R101	15	2	0.6	22	8	0.00	7.02	8	0.00	11.05	8
R101	15	2	1	30	8	0.00	10.76	8	0.00	16.99	8
R101	15	3	0.6	28	8	0.00	9.65	8	0.00	15.59	8
R101	15	3	1	45	8	0.00	20.21	8	0.00	31.57	8
		total			64	**0.03**	**12.27**	**63**	**0.03**	**19.43**	**63**

According to Table 3.3, the proposed algorithm solves to optimality 57 out of 64 small instances when 100 iterations are allowed, within an average computational time of 2 seconds. When we increase the number of iterations to 3000, the number of optimal solutions reaches 63 with an average computational time of 12 seconds.

Table 3.4: Impact of the number of iterations on mid-20 instances.

instances						ALNS (100 iterations)			ALNS (1000 iterations)		
	n	M	p	$avg.n_{cc}$	$nbIns$	$avg.\Delta_O$	$avg.t(s)$	$nbOPT$	$avg.\Delta_O$	$avg.t(s)$	$nbOPT$
C101	20	3	0.6	37.4	5	0.26	2.41	3	0.10	8.11	4
C101	20	3	1	60	5	0.66	5.75	1	0.11	21.13	2
R101	20	3	0.6	37.4	5	0.01	3.87	4	0.00	9.60	5
R101	20	3	1	60	5	0.82	10.68	1	0.16	25.66	2
		total			20	**0.44**	**5.68**	**9**	**0.09**	**16.12**	**13**
instances						ALNS (3000 iterations)			ALNS (5000 iterations)		
C101	20	3	0.6	37.4	5	0.10	20.66	4	0.00	33.38	5
C101	20	3	1	60	5	0.04	49.66	4	0.00	77.14	5
R101	20	3	0.6	37.4	5	0.00	22.09	5	0.00	34.87	5
R101	20	3	1	60	5	0.01	54.17	3	0.01	81.72	3
		total			20	**0.04**	**36.65**	**16**	**0.00**	**56.78**	**18**

The average gap with optimal solutions obtained by Archetti *et al.* (2015) varies from 0.10% to 0.03%. Increasing the number of iterations to 5000 does not improve the quality of the results.

When $n = 20$, results reported in Table 3.4 indicate that our method identifies an optimal solution in 5000 iterations for 18 out of the 19 instances for which Archetti *et al.* (2015) provided optimal values. The average gap with the best-known values is less than 0.01%. Over the whole set, the average CPU time is less than 1 minute.

Since large instances usually require more iterations to obtain high-quality results, we compare the ALNS algorithm behavior with 5000 and 10000 iterations on mid-80 instances. In Table 3.5, we report average results and the gap with respect to the best-known values. Detailed results are provided in Appendix A. ALNS can provide best-known solutions for the 20 mid-80 instances in the benchmark. Within 5000 iterations, the best-known solutions can be improved by about 0.79% on average and the CPU times is less than 9 minutes. When 10000 iterations are executed, the solutions are averagely 0.92% better than the best-known values. On the other side, the average computational times are almost doubled.

3.4.5 Computational experiments on the whole testbed

In this section, we consider the results obtained on the whole set of instances for the C-SDVRP with the designed ALNS algorithm. As determined in the previous sections, the removal and insertion heuristics are the ones of configuration 13, and

Table 3.5: Impact of the number of iterations on mid-80 instances.

instances						ALNS (5000 iterations)			ALNS (10000 iterations)		
	n	M	p	$avg.n_{cc}$	$nbIns$	$avg.\Delta_B$	$avg.t(s)$	$nbNBK$	$avg.\Delta_B$	$avg.t(s)$	$nbNBK$
C101	80	3	0.6	150.4	5	-0.77	373.36	5	-0.88	690.87	5
C101	80	3	1	240	5	-0.75	704.20	5	-0.83	1339.55	5
R101	80	3	0.6	150.4	5	-0.84	319.83	5	-1.06	605.74	5
R101	80	3	1	240	5	-0.81	646.61	5	-0.93	1196.69	5
	total				20	**-0.79**	**511.00**	**20**	**-0.92**	**958.21**	**20**

the number of iterations is equal to 3000 (resp. 5000) on small (resp. medium and large) instances. The algorithm is run once on each instance.

The comparison is made with the results reported in Archetti *et al.* (2014) and Archetti *et al.* (2015). Archetti *et al.* (2015) propose a branch-and-price algorithm that can solve to optimality instances with up to 40 customers but that can systematically close instances with up to 20 customers. When the optimal value is not available we compare to the best known solution (BKS) given either by Archetti *et al.* (2014) or Archetti *et al.* (2015).

For the 64 small instances, an optimal solution is known. Results reported in Table 3.3 indicate that our algorithm finds an optimal solution for 63 out of 64 instances. The average optimality gap is 0.03%.

For the mid-20 instances ($n = 20$), 19 out of 20 instances have a known optimal value. The proposed algorithm can find 18 out of 19 optimal values. On average, the gap with respect to BKS (either optimal or feasible) is lower than 0.01%.

For the mid-40 instances ($n = 40$), average results are reported in Table 3.6. Only 5 optimal values are known. On average, the proposed ALNS finds solutions that are 0.26% better than the best-known solutions (either optimal or feasible). Our algorithm finds 4 out of the 5 known optimal solution while 9 new best-known solutions have been identified. The algorithm runs in less than 2 minutes on average.

Table 3.6: Summary of results on mid-40 sized instances.

instances						ALNS results						
	n	M	p	$avg.n_{cc}$	$nbIns$	$avg.\Delta$	$min\Delta$	$max\Delta$	$avg.t(s)$	$nbOPT$	nbE	$nbNBK$
C101	40	3	0.6	76.2	5	-0.12	-0.42	0.15	78.14	2	-	2
C101	40	3	1	120	5	-0.76	-1.91	0.00	161.39	-	-	4
R101	40	3	0.6	76.2	5	0.15	0.00	0.29	80.70	1	-	-
R101	40	3	1	120	5	-0.32	-0.78	0.00	148.33	1	1	3
		total			20	**-0.26**	-1.91	0.29	**117.14**	**4**	**1**	**9**

For instances with $n \geq 60$, no optimal solution is available. For the mid-60 instances ($n = 60$) (see Table 3.7), the ALNS finds solutions that are on average 0.49% better than the BKS. Among them, 15 new best-known values are obtained. The CPU times are less than 5 minutes on average.

Table 3.7: Summary of results on mid-60 sized instances.

instances						ALNS results					
	n	M	p	$avg.n_{cc}$	$nbIns$	$avg.\Delta_B$	$min\Delta_B$	$max\Delta_B$	$avg.t(s)$	nbE	$nbNBK$
C101	60	3	0.6	110	5	-0.56	-1.40	0.00	193.12	1	4
C101	60	3	1	180	5	-0.38	-1.13	0.25	403.78	-	3
R101	60	3	0.6	110	5	-0.41	-0.83	0.00	177.20	1	3
R101	60	3	1	180	5	-0.61	-1.20	-0.14	353.24	-	5
	total				20	**-0.49**	-1.40	0.25	**281.83**	**2**	**15**

As presented in Table 3.5 on mid-80 instances ($n = 80$), the ALNS identifies solutions that are 0.79% better than the BKS, and new best-known values are found for all instances.

Table 3.8 reports the results on the 320 large instances ($n = 100$). The ALNS finds 300 new best-known values with an average improvement of 0.70%. The computational time is around 10 minutes which is very reasonable when considering the instance size.

Table 3.8: Summary of results on large sized instances.

instances						ALNS results				
	n	M	p	$avg.n_{cc}$	$nbIns$	$avg.\Delta_B$	$min\Delta_B$	$max\Delta_B$	$avg.t(s)$	$nbNBK$
C101	100	2	0.6	136.4	40	-0.66	-1.69	0.24	330.02	38
C101	100	2	1	200	40	-0.77	-1.64	0.03	569.11	39
C101	100	3	0.6	188.4	40	-0.77	-2.04	0.73	557.46	39
C101	100	3	1	300	40	-1.17	-2.88	0.32	1106.23	37
R101	100	2	0.6	136.4	40	-0.53	-1.50	1.05	323.48	36
R101	100	2	1	200	40	-0.55	-1.28	0.29	557.74	36
R101	100	3	0.6	188.4	40	-0.53	-1.23	0.02	548.02	39
R101	100	3	1	300	40	-0.62	-1.72	0.25	1078.70	36
		total			320	**-0.70**	-2.88	1.05	**633.85**	**300**

3.4.6 Effectiveness of MPO operator in the ALNS algorithm

As described in Section 3.3.7 we developed a Mathematical Programming based Operator (MPO) to re-assign commodities for one customer to the routes of a solution. Due to the increase observed in computational time consumption, we decided to use

it only to improve a new global best solution further. In this section, we analyze the results on medium and large instances to prove the effectiveness of MPO in our algorithm. In Table 3.9, we indicate the average number of times that MPO is called for each group of instances as well as the average number of times that MPO improves the new best solution for each group of instances. Detailed results are reported in Appendix A (Table A2 and Table A3).

Table 3.9: Effectiveness of MPO in the ALNS algorithm.

instances				results			
	n	M	p	$avg.n_{cc}$	$nbIns$	$avg.nbMPO$	$avg.nbMPOimp$
C101	20	3	0.6	37.4	5	2.60	0.00
C101	20	3	1	60	5	7.00	0.20
R101	20	3	0.6	37.4	5	4.00	0.20
R101	20	3	1	60	5	9.80	0.20
C101	40	3	0.6	76.2	5	8.00	0.20
C101	40	3	1	120	5	16.40	2.00
R101	40	3	0.6	76.2	5	12.40	0.00
R101	40	3	1	120	5	18.60	1.80
C101	60	3	0.6	110	5	20.80	0.20
C101	60	3	1	180	5	30.80	1.60
R101	60	3	0.6	110	5	18.00	0.80
R101	60	3	1	180	5	24.60	1.60
C101	80	3	0.6	150.4	5	29.00	1.60
C101	80	3	1	240	5	32.00	6.40
R101	80	3	0.6	150.4	5	23.80	1.00
R101	80	3	1	240	5	29.60	2.40
C101	100	2	0.6	136.4	40	27.05	0.85
C101	100	2	1	200	40	29.65	2.00
C101	100	3	0.6	188.4	40	31.60	1.33
C101	100	3	1	300	40	37.28	3.33
R101	100	2	0.6	136.4	40	27.15	0.53
R101	100	2	1	200	40	30.15	2.10
R101	100	3	0.6	188.4	40	32.40	2.23
R101	100	3	1	300	40	34.48	3.58

For medium instances (namely for $n = 20, 40, 60, 80$), the impact of MPO on the quality of the solution increases as the instance size increases. Especially, when $p = 1$, MPO has a greater impact than when $p = 0.6$. The main reason is that, when $p = 1$, all customers require all the commodities while a smaller number of commodities are required when $p = 0.6$. As a consequence, for the same number of customers in the instance, the number of *customer-commodities* is larger. Therefore, when $p = 1$, the difficulty of solving the problem increases as well as the possibilities to reassign commodities with MPO. For large instances ($n = 100$), the same conclusions on the performance of MPO can be drawn.

3.4.7 Evaluation of the LS in the ALNS algorithm

In this section, we evaluate the importance of the local search in our ALNS algorithm. We run, on mid-80 instances, the ALNS algorithm without LS and MPO with a limit of 400000 iterations. We then compare the results obtained by the ALNS with LS and MPO on 5000 iterations. This choice is made to have a comparison fair: we remove two optimization components, but we allow a large number of iterations.

We compare the results obtained by the proposed algorithm, indicated by *ALNS+LS+MPO*, with the same algorithm when LS and MPO are deactivated. This last version is indicated as *ALNS-LS-MPO*. The performance comparison of both variants is reported in Table 3.10 for mid-80 and large instances.

It is clear from Table 3.10 that after 400000 iterations the results obtained by the ALNS without LS and MPO are of lower quality than those obtained by the original ALNS. We only obtain 15 new best-known values for the 20 mid-80 instances. Moreover, the average improvement (0.39%) does not compete with the improvement obtained with the proposed algorithm (ALNS+LS+MPO) while the computational times are similar. The same observations can be made from the results for large instances. We then conclude on the importance of the LS and MPO in the proposed ALNS algorithm.

3.4.8 Trend between instance size and computational time

Last, we examine how the CPU time required by the ALNS varies according to the instance size. We consider the results obtained with 5000 iterations (even for the small instances) to perform the analysis. We sort the 464 (small, medium and large) instances according to the number of *customer-commodities* instead of the number of customers. The variation of the computational time $avg.t(s)$ according to the number of *customer-commodities* $avg.n_{cc}$ is depicted on Figure 3.9.

When the size of the instances increases, the average computational time significantly increases (not in a linear fashion). This behavior is not surprising since when the size of the instances increases, it takes more time to operate the LS operators, which computational complexity is $O(n_{cc}^2)$. Nevertheless, the average computational time of our ALNS algorithm for the large instances ($n_{cc} = 300$) remains reasonable with less than 19 minutes.

Table 3.10: Comparison between two ALNS variants on mid-80 and large instances.

instances						ALNS+LS+MPO (5000 iterations)					ALNS-LS-MPO (400000 iterations)				
	n	M	p	$avg.n_{cc}$	$nbIns$	$avg.\Delta_B$	$min\Delta_B$	$max\Delta_B$	$avg.t(s)$	$nbNBK$	$avg.\Delta_B$	$min\Delta_B$	$max\Delta_B$	$avg.t(s)$	$nbNBK$
C101	80	3	0.6	150.4	5	-0.77	-1.48	-0.01	373.36	5	-0.70	-1.50	0.11	330.77	4
C101	80	3	1	240	5	-0.75	-1.36	-0.24	704.20	5	-0.49	-1.08	0.02	686.68	4
R101	80	3	0.6	150.4	5	-0.84	-1.24	-0.05	319.83	5	-0.49	-1.03	0.57	348.14	4
R101	80	3	1	240	5	-0.81	-1.39	-0.50	646.61	5	0.11	-0.83	1.71	712.20	3
		total			20	**-0.79**	-1.48	-0.01	**511.00**	**20**	**-0.39**	-1.50	1.71	**519.45**	**15**
C101	100	2	0.6	136.4	40	-0.66	-1.69	0.24	330.02	38	-0.23	-1.52	1.21	360.30	28
C101	100	2	1	200	40	-0.77	-1.64	0.03	569.11	39	-0.30	-1.58	1.30	606.03	26
C101	100	3	0.6	188.4	40	-0.77	-2.04	0.73	557.46	39	-0.28	-1.66	1.66	576.18	25
C101	100	3	1	300	40	-1.17	-2.88	0.32	1106.23	37	-0.57	-2.47	1.20	1256.61	26
R101	100	2	0.6	136.4	40	-0.53	-1.50	1.05	323.48	36	0.00	-0.75	1.12	366.70	20
R101	100	2	1	200	40	-0.55	-1.28	0.29	557.74	36	-0.08	-1.24	1.83	614.35	25
R101	100	3	0.6	188.4	40	-0.53	-1.23	0.02	548.02	39	0.21	-0.94	1.61	576.27	16
R101	100	3	1	300	40	-0.62	-1.72	0.25	1078.70	36	0.10	-1.78	2.67	1211.77	21
		total			320	**-0.70**	-2.88	1.05	**633.85**	**300**	**-0.14**	-2.47	2.67	**696.03**	**187**

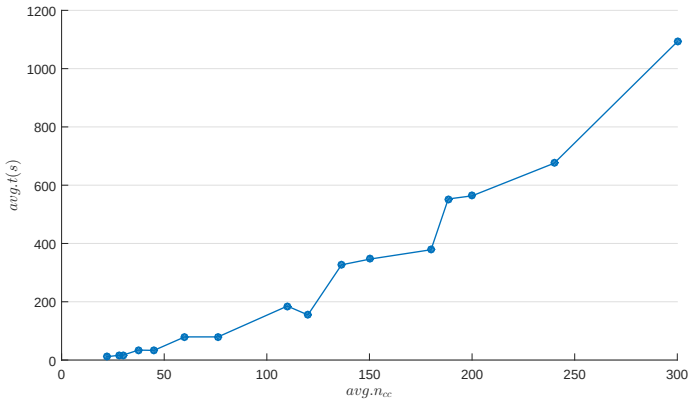


Fig. 3.9. The $avg.t(s)$ of the ALNS with respect to $avg.n_{cc}$

3.4.9 Characteristics of split customers

As mentioned above, considering *customer-commodity* makes the problem more complex with equivalent solutions since several commodities are related to the same customer and then have the same location. Identifying the customers who are good candidates for being split into *customer-commodities* can be beneficial for decision-makers. In this section, we propose to study the characteristics of customers who are delivered by more than one vehicle. We name these customers *split customers*. We provide detailed results on the 20 mid-80 instances with 5000 iterations for the ALNS algorithm.

Table 3.11 reports these detailed results for several characteristics about the split customers in the best solution obtained on each instance. The first three columns report the characteristics of the instance, and the fourth column indicates the identification number of the instance. Column *nbSplit* reports the number of split customers (out of 80), and columns *nb*2-*split* and *nb*3-*split* indicate the number of customers delivered by respectively 2 and 3 vehicles. Columns $nbNearDepot$ and $nbLargeDemand$ report the number of split customers located near the depot and the number of split customers with a large demand respectively. As Nagy *et al.* (2015), we consider that a customer is located near the depot if he or she is one of the 25% of customers closest to the depot; and we consider that a customer has a large demand if he or she is one of the 25% of customers with the largest demand. Note that the demand of a customer is the sum of the demands for the commodities they need. Column $nbCluster$ reports the number of split customers located inside

a cluster of customers. As Nagy *et al.* (2015), we consider a customer to be in a cluster if at least five other customers are inside its neighborhood. Two customers are neighbors if the distance between these two customers is less than 30% of the average distance between the depot and all the customers in the instance. Using this definition, in clustered instances (C101) more than 70% of customers are in a cluster, while in random instance (R101) less than 30% of customers are in a cluster.

Table 3.11: Characteristics of split customers in the best solutions of mid-80 instances (5000 iterations).

instances									
	n	p	id	$nbSplit$	$nb2\text{-}split$	$nb3\text{-}split$	$nbNearDepot$	$nbLargeDemand$	$nbCluster$
C101	80	0.6	1	7	7	0	2	3	7
			2	5	5	0	3	1	5
			3	7	7	0	4	2	5
			4	9	8	1	3	5	7
			5	10	10	0	1	7	6
		1	1	20	19	1	5	5	18
			2	14	14	0	4	4	14
			3	18	18	0	6	4	13
			4	14	14	0	4	4	14
			5	18	18	0	6	2	16
R101	80	0.6	1	6	6	0	4	6	4
			2	5	5	0	3	2	2
			3	7	7	0	2	3	4
			4	5	5	0	3	1	3
			5	6	6	0	6	3	3
		1	1	13	13	0	3	3	7
			2	14	13	1	7	1	4
			3	13	13	0	4	2	6
			4	14	14	0	4	4	6
			5	17	17	0	6	5	7

From the results in Table 3.11, it is clear that there are more split customers when customers require more commodities. p denotes the probability that a customer requires a commodity. When $p = 0.6$, the average number of split customers is 6.7, while when $p = 1$, there are 15.5 split customers on average. Moreover, very few split customers are delivered by three vehicles, i.e., one vehicle for each of the required commodities. In the 20 instances, there are only three split customers in this case. In addition, one important feature of split customers is to be inside a cluster. Indeed, for C101 instances, 86% of split customers are inside a cluster. For R101 instances, 46% of split customers are inside a cluster, while less than 30% of customers are in a cluster in these instances. Proximity to the depot is also an important feature for split customers since 36% of split customers are near the depot. It is less obvious

to link large demands to split customers since 30% of split customers have a large demand.

3.5 Conclusions

In this paper, we presented a dedicated heuristic algorithm for the C-SDVRP. The proposed algorithm is based on an Adaptive Large Neighborhood Search framework introduced by Ropke & Pisinger (2006). This is the first heuristic specifically designed with the aim to provide high-quality solutions for the medium and large size instances. According to the main feature of the C-SDVRP, i.e. the requirement of different commodities, we adapt some classical local search moves to consider either with a customer (i.e., a customer and all its commodities) or a *customer-commodity* (namely, a single commodity required by a customer). We developed a Mathematical Programming based Operator (MPO) to intensify the search and further improve the best solutions. The results show that our ALNS algorithm is very effective in finding high-quality solutions on large size instances. In particular, our method outperforms the algorithms proposed in Archetti *et al.* (2014) and in Archetti *et al.* (2015).

The proposed ALNS algorithm could then be adapted to tackle other variants of routing problems with split deliveries. One of these variants is the case with multiple depots and available quantities at each depot. In this case, because of the limited quantities available for each commodity at each depot, it is worth considering splitting deliveries to find feasible solutions. Another interesting variant of the problem is the VRP with divisible deliveries and pickup (Gribkovskaia *et al.* (2007), Nagy *et al.* (2015)): in this case, delivery and pickup naturally represent two different commodities, but a pickup operation increases the use of the vehicle capacity, and it could then be optimal to visit twice the same customer in one route.

This work is partially supported by the CSC (China Scholarship Council) and by the ELSAT 2020 project. The authors thank C. Archetti and N. Bianchessi for providing the benchmark instances. Thanks are also due to the referees for their valuable comments.

Chapter 4

A decomposition approach to a Multi-Commodity two-echelon Distribution Problem

Contents

Abstract: We address a Multi-Commodity two-echelon Distribution Problem (MC2DP) where three sets of stakeholders are involved: suppliers, distribution centers, and customers. Multiple commodities have to be sent from suppliers to customers, using multiple distribution centers for consolidation purposes. Commodities are collected from the suppliers and delivered to the distribution centers with direct trips, while a fleet of homogeneous vehicles distributes commodities to customers. Commodities are compatible, that is any vehicle can transport any set of commodities as long as its capacity is not exceeded. The goal is to minimize the total transportation cost from suppliers to customers. We present two sequential solution approaches based on the solution, in a different order, of a collection and a delivery subproblem. In both cases, the solution of the first subproblem determines the quantity of each commodity at each distribution center. The second subproblem takes this information as input. We also propose different strategies to guide the solution of the first subproblem in order to take into account the impact of its solution on the second subproblem. The proposed sequential approaches are evaluated and compared both on randomly generated instances and on a case study related to a short and local fresh food supply chain. The results show the impact of problem characteristics on solution strategies.

Keywords: multicommodity, routing problem, local fresh food supply chain, sequential solution.

Introduction

In this paper we study a complex distribution problem in a two-echelon supply chain where three sets of stakeholders are involved: suppliers, distribution centers and customers. Multiple commodities are collected from the suppliers and delivered to the customers through distribution centers for consolidation purposes. Each supplier has a given available quantity for each commodity (possibly 0) and each customer has a demand for each commodity (possibly 0). We consider a single decision maker who

manages all distribution centers and organizes the collection and delivery operations. The commodities are collected from suppliers and delivered to distribution centers through direct trips, and distributed from the distribution centers to customers with a fleet of vehicles performing routes. Direct deliveries from suppliers to customers are not allowed. Commodities are compatible, that is any vehicle can transport any set of commodities as long as its capacity is not exceeded. Multiple visits to a customer are allowed to reduce transportation costs. However, for the sake of customers convenience, a single commodity has to be delivered at once. The problem is named Multi-Commodity two-echelon Distribution Problem (MC2DP). The objective is to find a collection and delivery plan that minimizes the total transportation cost, satisfying customer demands, and not exceeding the available quantities at the suppliers and the vehicle capacities. The study of this problem is motivated by a case study presented in Section 4.6 for the collection and delivery of fresh agri-food products (fruits and vegetables) through a short and local supply chain.

In the following, we detail the application on short and local fresh food supply chain to motivate our work. The production and delivery of fresh food products have undergone important changes in Europe since the 1950's, especially through the modernization of the tools and processes in order to meet the customer demand with low production costs. Multinational companies have played a major role as intermediaries between farmers and consumers (Rucabado-Palomar & Cuéllar-Padilla, 2018). Nowadays, one of the major problems faced by farmers is the shortfall of their incomes: over the last decades they have been encouraged to produce more, while their unit selling price was decreasing. However, in many regions there coexist (1) supplies with medium-sized farms where various products of high quality (freshness, few pesticide) are cultivated and (2) customers with a strong desire for product quality and traceability (King *et al.*, 2015). Hence, the idea has emerged to locally connect suppliers and customers (Berti & Mulligan, 2016), by means of a short (and/or) local food supply chain. The main purpose of this kind of supply chain is to capture more end-use value for the farmers.

Short food supply chains are defined as an opportunity for agricultural products to reach the market either through direct sales or through indirect sales with only one intermediary between producers and consumers. Local food supply chains may involve several intermediaries with all the actors located on a limited area (e.g.

considering geographical or political restrictions). The maximum distance between actors is usually around 80 km (Blanquart *et al.*, 2010). In this paper, we consider short and local food supply chains with indirect sales to canteens, restaurants or supermarkets.

Short and local supply chains involve few intermediaries. Hence, farmers have to take charge of a large part of their products marketing and distribution, which is not their core business. It is feasible for direct sales since the volumes are usually low, and consequently farmers can spend time selling their products. For indirect sales (canteen, restaurants or supermarket), volumes are more important and, therefore, the supply chain has to rely on a set of distribution centers in order to organize product flows and to minimize transportation costs, with the aim to be competitive with conventional food supply chains. Farmers supply these distribution centers by performing direct trips since the volumes are large. The distribution centers are then in charge of consolidation and delivery of the products to customers. In a local supply chain context, all the actors (farmers, distribution centers and customers) are located in a restricted area. A single decision maker manages all the distribution centers, and coordinates the transportation planning for both collection and delivery operations. This decision maker can be an association of farmers, or a local political authority. These distribution centers are considered as the only intermediary in the supply chain.

The Multi-Commodity two-echelon Distribution Problem belongs to the broad class of two-echelon routing problems, which are distribution problems where transportation activities take place in two echelons of a supply chain. There exists a wide literature on this class of problems. We refer to Cuda *et al.* (2015) for a recent survey on two-echelon routing problems and to Guastaroba *et al.* (2016) for a more general survey on transportation problems with intermediate facilities.

The problem we study here contributes to the literature on two-echelon routing problems by explicitly considering multiple commodities that are required by final customers. To the best of our knowledge, the multiple commodities aspect has not been considered before in two-echelon distribution systems. On the other side, two-echelon location problems that deal with multiple commodities have already been tackled (see, for example, Hinojosa *et al.* (2000) and Sadjady & Davoudpour (2012)). These problems do not involve routing decisions but evaluate the cost of assigning final customers to a specific distribution center.

The explicit consideration of different commodities is essential in the agri-food supply chains since availability at the producer depends on the production of farmers and requirements made by customers concern specific commodities.

There is a large literature on planning problems in agri-food supply chains. The main focus is, in general, the integration of degradation and quality of products inside the planning models. When focusing on short and local food supply chain, the literature is very scarce. As pointed out by Flores & Villalobos (2018), there is a lack of supply chain planning tools for local fresh food supply chains. Flores & Villalobos (2018) develop an agricultural planning framework that determines an optimal production planning of vegetables for local supply chains. Bosona & Gebresenbet (2011) address a case study in Sweden. They first study the location of distribution centers using the centre-of-gravity technique. Then, farmers are assigned to one distribution center, and can bring their products to the distribution center, or the latter can perform collection by grouping farmers into routes. Routes are then optimized using a dedicated software (Route LogiX). Ogier *et al.* (2013) propose a mixed integer linear programming model for service network design of a short and local supply chain.

Even though the application presented in this paper deals with the agri-food supply chain, the methods we propose can be applied to MC2DPs that consider direct delivery trips in the first echelon and allow splitting the delivery of different commodities in the second level.

In this paper, we present a solution approach to the MC2DP. Due to the complexity of the problem, we propose a decomposition approach of the MC2DP in two subproblems, associated with the collection and delivery phases, respectively, and the sequential solution of the subproblems. Two sequential approaches to the solution of the MC2DP are presented, depending on which of the two subproblems is solved first. In both cases, the solution of the first subproblem determines the quantity of each commodity at each distribution center. The second subproblem takes this information as input. We also propose different strategies to guide the solution of the first subproblem in order to take into account the impact of its solution on the second subproblem. It is worth noting that the subproblem associated with the delivery phase is itself a new problem. It is a Vehicle Routing Problem (VRP) with multiple commodities and multiple depots, with a maximum available quantity of each commodity at the depots. VRP with multiple depots is a well-studied problem

in the literature (Montoya-Torres *et al.*, 2015). In this paper, the delivery subproblem is an extension of the VRP with multiple depots where several commodities are available at each depot with given quantities. A further contribution of this paper is a solution method for the delivery subproblem, based on the Adaptive Large Neighborhood Search (ALNS) algorithm proposed for the one depot case in Chapter 3. The two proposed sequential approaches and the different strategies are evaluated and compared both on randomly generated instances, with different characteristics (supplier locations, customer locations, maximum supply quantities, number of distribution centers), and on a case study for the collection and delivery of fresh food products (fruits and vegetables) through a short and local supply chain using a set of distribution centers located in the French department of Isère. The size of the instances for this case study is large, and two kinds of customers are considered: school canteens and supermarkets. The computational results show the impact of the instance characteristics on the solution approaches and strategies.

The remainder of this paper is organized as follows. A definition of the MC2DP is given in Section 4.1. The decomposition approach and the collection and delivery subproblems are presented in Section 4.2. Section 4.3 provides a theoretical analysis of the benefits of distribution centers, on the complexity of the problem and on the sequential solution approaches. Then, the sequential approaches are described in Section 4.4. Section 4.5 provides the computational results. The case study on short and local fresh food supply chains in the French department of Isère is discussed in Section 4.6. Finally, conclusions and prospects are presented in Section 4.7.

4.1 Problem definition

The Multi-Commodity two-echelon Distribution Problem (MC2DP) is defined on a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, in which $\mathcal{V}$ is the set of vertices and $\mathcal{A}$ is the set of arcs. More precisely, $\mathcal{V}$ is defined as $\mathcal{V}_S \bigcup \mathcal{V}_D \bigcup \mathcal{V}_C$ where $\mathcal{V}_S = \{1, \dots N_S\}$ represents the set of suppliers, $\mathcal{V}_D = \{N_S + 1, \dots, N_S + N_D\}$ is the set of distribution centers and $\mathcal{V}_C = \{N_S + N_D + 1, \dots, N_S + N_D + N_C\}$ represents the set of customers. We only consider direct trips from suppliers to distribution centers. Direct deliveries from suppliers to customers are not allowed. Moreover, transfers of commodities between distribution centers are not considered. Thus, $\mathcal{A} = \{(i,j),(j,i) | i \in \mathcal{V}_S, j \in \mathcal{V}_D\} \cup \{(i,j),(j,i) | i \in \mathcal{V}_D, j \in \mathcal{V}_C\} \cup \{(i,j) | i,j \in \mathcal{V}_C\}$ is the set of arcs.

Suppliers provide a set of commodities $\mathcal{M}$, which are transported to the distribution centers by an unlimited fleet of homogeneous vehicles of capacity Q_S. Each supplier $s \in \mathcal{V}_S$ is associated with a maximum available quantity O_{sm} (possibly equal to zero) of commodity $m \in \mathcal{M}$. Each distribution center has its own unlimited fleet of homogeneous vehicles of capacity Q_D that are used to deliver the commodities to the customers. Customer $i \in \mathcal{V}_C$ requires a quantity D_{im} (possibly equal to zero) of each commodity $m \in \mathcal{M}$. Commodities are compatible, i.e., they can be transported on the same vehicle. The demand of a customer can be split, that is the customer can be served by several vehicles. However, for the sake of customer convenience, the split policy is constrained: each commodity has to be delivered by one vehicle only.

A cost c_{ij} is associated with each arc $(i, j) \in \mathcal{A}$ and represents the non-negative cost of traversing arc (i, j). In the MC2DP, the decision maker is the logistic provider who manages all the distribution centers, decides how to collect commodities from the suppliers and how to distribute the commodities from the distribution centers to the customers. The objective is to find a collection and delivery plan that minimizes the total transportation cost, satisfying customer demands, not exceeding the available quantities at the suppliers and the vehicle capacities.

In the following, we will call *collection* the transportation of commodities from suppliers to distribution centers and *delivery* the distribution of commodities from distribution centers to customers. The collection and delivery phases of the MC2DP are connected through the quantities of commodities at the distribution centers. We denote by U_{dm} the unknown quantity of commodity $m \in \mathcal{M}$ at distribution center d.

An example of an instance of the MC2DP is depicted in Figure 4.1. There are two commodities, and the vehicle capacities are $Q_S = 8$ and $Q_D = 10$. A feasible solution of this instance is shown in Figure 4.2. In the solution, five vehicles are used to supply the distribution centers, each vehicle performing a direct trip. The numbers on arcs representing direct trips report the quantities transported for each commodity, where the number corresponding to the commodity is reported in parentheses. Note that two vehicles are used from supplier 1 to distribution center 4. Quantities U_{dm} are as follows: $U_{41} = 4$, $U_{42} = 14$, $U_{51} = 5$ and $U_{52} = 3$. For the distribution to customers, two vehicles leave distribution center 4, while one vehicle leaves distribution center 5. Note that customer 9 is visited twice: one vehicle delivers commodity 1 while another vehicle delivers commodity 2.

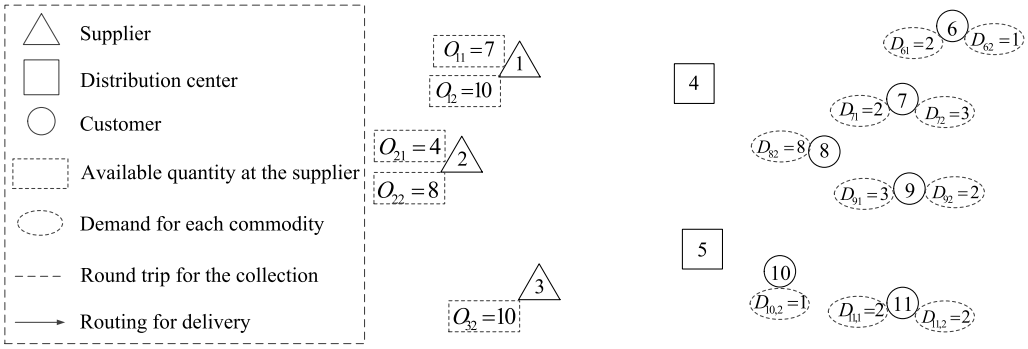


Fig. 4.1. An instance of the MC2DP.

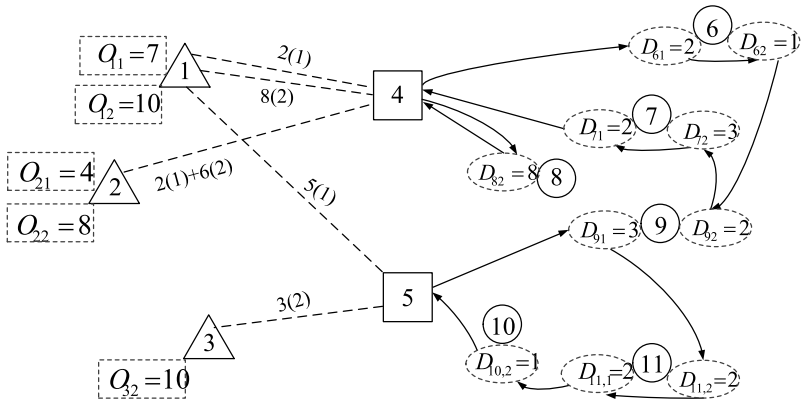
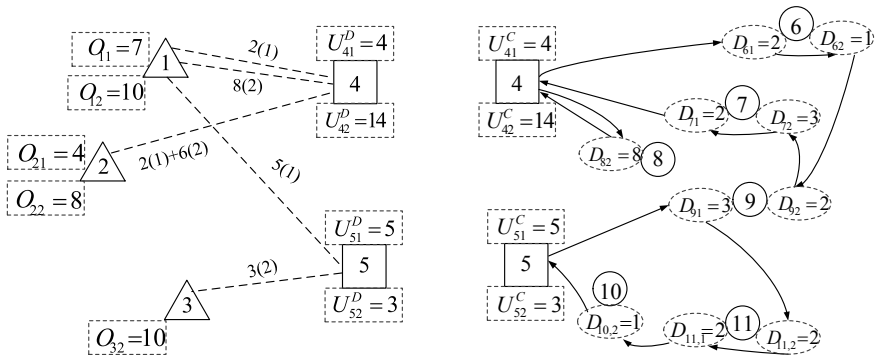


Fig. 4.2. A feasible solution of the MC2DP instance.

4.2 A decomposition approach

The MC2DP integrates two subproblems, one considering the collection of commodities, that is the transportation of the commodities from the suppliers to the distribution centers, the other considering the delivery of commodities from the distribution centers to the customers. In this section, we present the decomposition approach to the solution of the MC2DP and define the two subproblems: the SPC (SubProblem Collection) and the SPD (SubProblem Delivery). The connection of these two problems is made by quantities U_{dm} that represent the available quantity of commodity m at distribution center d. For the sake of clarity, when necessary, we will denote by U^D_{dm} the quantity of commodity m that is delivered from the distribution center d to customers, and by U^C_{dm} the quantity of commodity m that is collected at the suppliers and delivered to the distribution center d. We provide an example of this decomposition approach in Figure 4.3. From the solution of MC2DP in Figure 4.2, we provide the related solutions of the SPC and the SPD with the associated U_{dm} variables to link the two subproblems.



(a) Solution of the SPC. (b) Solution of the SPD.

Fig. 4.3. Solutions of the two subproblems based on the solution of the MC2DP in Figure 4.2.

We now provide a formal definition of SPC and SPD. Note that each subproblem is concerned with the optimization of the related operations, i.e., SPC minimizes the cost of collection operations only while SPD minimizes the cost of delivery operations only.

4.2.1 The Collection Subproblem (SPC)

The SPC is defined on a graph $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{A}_1)$, where $\mathcal{V}_1 = \mathcal{V}_S \bigcup \mathcal{V}_D$ and $\mathcal{A}_1 = \{(i,j),(j,i) | i \in \mathcal{V}_S, j \in \mathcal{V}_D\}$. In the following, we will use the word *truck* to indicate a vehicle used in the SPC. The SPC consists in determining a set of direct trips for trucks between suppliers and distribution centers with the associated quantities for each commodity. The objective is to minimize the transportation cost, defined as the total cost for the direct trips, that is independent of the quantity transported on each truck. For each commodity, the quantity transported to distribution centers has to be sufficient to satisfy customer demands in the SPD. Moreover, the solution of the SPC must satisfy the following constraints:

(1) the total quantity of commodities transported by each truck does not exceed the truck capacity Q_S;

(2) for each supplier s, the quantity of each commodity m that is transported to distribution centers must be at most equal to the available quantity O_{sm};

(3) the quantity of each commodity m transported to each distribution center d is greater than the required quantity U^D_{dm}.

When the SPC is solved first, the values U^D_{dm} are not known and have to be set to a valid lower bound. Next, the values U^C_{dm} are computed from the solution of the SPC and used as input to the SPD.

The SPC is related to the Multi-commodity Capacitated fixed-charge Network Design problem (MCND) (Magnanti & Wong, 1984). The MCND problem is a discrete optimization problem in which a set of commodities has to be routed through a directed network. Each commodity has a demand to be transported from an origin to a destination. Each arc has a limited capacity, a unit cost flow, and a fixed cost if the arc is used (the flow is positive). The SPC differs from the MCND problem because of the cost structure. In the SPC, the cost of an arc depends on the number of trucks used: it is a step-wise cost function defined by a unitary cost associated with each truck used. Thus, steps are defined as multiples of the truck capacity Q_S.

4.2.2 The Delivery Subproblem (SPD)

The SPD is defined on a graph $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{A}_2)$ where $\mathcal{V}_2 = \mathcal{V}_D \bigcup \mathcal{V}_C$ and $\mathcal{A}_2 = \{(i,j),(j,i)|i \in \mathcal{V}_D, j \in \mathcal{V}_C\} \cup \{(i,j)|i,j \in \mathcal{V}_C\}$. In the following, we will use the word *vehicle* to indicate a vehicle used in the SPD.

The SPD consists in assigning commodities to vehicles and in determining a set of vehicle routes to meet all customer demands. The solution must satisfy the following constraints:

(1) the total quantity of commodities delivered by each vehicle does not exceed the vehicle capacity Q_D;

(2) each commodity requested by each customer is delivered by a single vehicle;

(3) the demand of all customers is satisfied;

(4) the quantity of each commodity m distributed from each distribution center d does not exceed the available quantity U^C_{dm};

(5) each vehicle starts and ends its route at the same distribution center.

When the SPD is solved first, the values U^C_{dm} are not known and have to be set to a valid upper bound. Next, the values U^D_{dm} are computed from the solution of the SPD and used as input to the SPC.

An instance of the SPD with two distribution centers, six customers and two commodities is illustrated in Figure 4.4. Each vehicle has a capacity Q_D of 10 units. Figure 4.4 also shows a feasible solution.

The SPD is the multi-depot case of the Commodity constrained Split Delivery Vehicle Routing Problem (C-SDVRP)(Archetti *et al.*, 2014). The C-SDVRP is a problem where customers require multiple commodities. Each customer can be served by different vehicles, but each commodity has to be delivered at once by a single vehicle. The C-SDVRP considers only one distribution center with sufficient quantity for each commodity to satisfy all the customer demands.

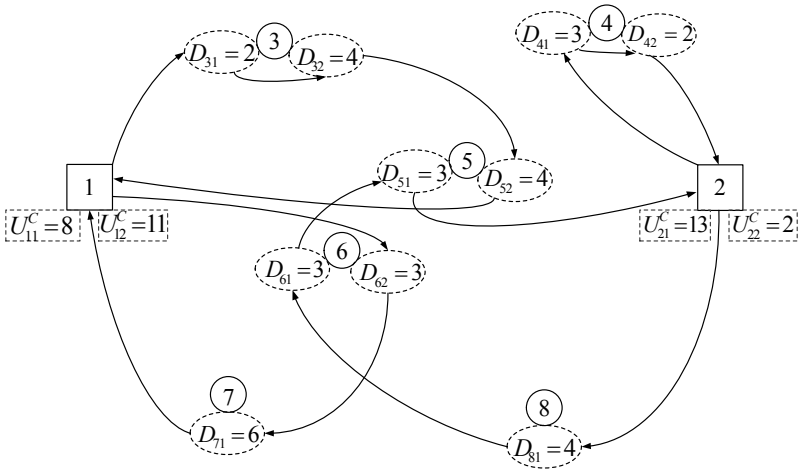


Fig. 4.4. An instance with feasible solution of the SPD.

4.3 Analysis of the MC2DP

This section provides an analysis of the properties of the MC2DP. In particular, Section 4.3.1 focuses on the role of distribution centers on the structure of optimal solutions. Section 4.3.2 presents an analysis of cases in which the sequential solution of the SPC and SPD provides an optimal solution to MC2DP. Finally, Section 4.3.3 studies the complexity of special cases of the MC2DP.

4.3.1 On the benefit of distribution centers

The MC2DP consists in the collection of commodities from suppliers and in the delivery to customers via consolidation at distribution centers. In order to illustrate the benefits of having one or several distribution centers, let us introduce the following example:

Example 4.1 *Consider the case with one commodity and one distribution center, i.e.,* $|\mathcal{M}| = |\mathcal{V}_D| = 1$. *Suppose that* $\mathcal{V}_S$ *is composed of two suppliers* 1 *and* 2 *that offer 9 and 6 units of commodity, respectively, and* $\mathcal{V}_C$ *contains three customers,* $4, 5$ *and* 6*, that respectively require 4, 5 and 6 units of the commodity. The example is illustrated in Figure 4.5. It provides locations of suppliers, the distribution center, customers and distances. Let us now suppose that direct deliveries from suppliers to customers, i.e., without passing through distribution centers, are allowed. The best solution making direct deliveries from suppliers to customers is depicted in Figure 4.6 and has a cost of* $4a + 6a\sqrt{2}$. *The best solution in which consolidation at*

distribution centers is mandatory, i.e., the optimal solution of the MC2DP, is depicted in Figure 4.7 and has a cost of $8a\sqrt{2}$. *Thus, the latter solution is cheaper than the one performing direct deliveries.*

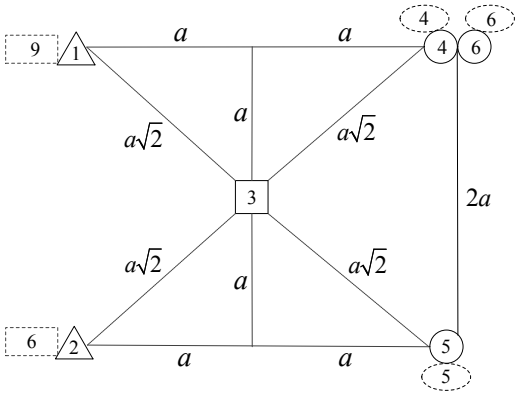


Fig. 4.5. Illustration of Example 4.1.

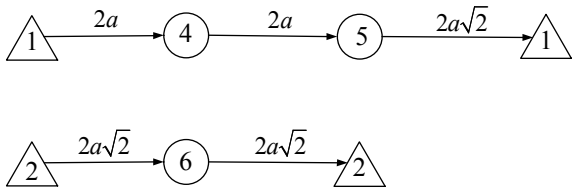


Fig. 4.6. The best solution that does not consider consolidation at the distribution center.

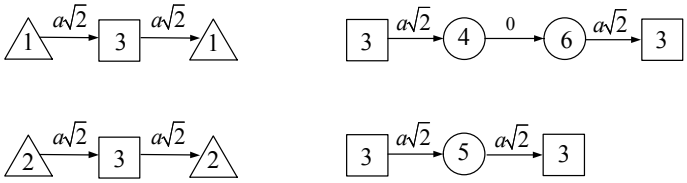


Fig. 4.7. The optimal solution considers consolidation at the distribution center.

Example 4.1 aims at showing that consolidation at distribution centers might be beneficial even in basic settings like the one considered in the example ($|\mathcal{M}| = |\mathcal{V}_D| = 1$).

In the MC2DP consolidation at distribution centers is mandatory. The following proposition shows what happens in case one distribution center only is considered.

Proposition 4.2 *Let us consider the MC2DP with* $|\mathcal{V}_\mathcal{D}| = 1$. *In this case, the MC2DP can be decomposed in the SPC and the SPD, i.e. solving sequentially to optimality the SPC and the SPD solves the MC2DP to optimality.*

From Proposition 4.2 it follows that the interesting cases when studying the MC2DP are those with $|\mathcal{V}_D| > 1$ and $|\mathcal{M}| \geq 1$. These are the cases on which we focus our study.

4.3.2 Sequential solution of the MC2DP

The sequential solution of the SPC and the SPD provides a heuristic solution to the MC2DP. The first solved subproblem determines the quantity of each commodity at each distribution center. The second solved subproblem uses this information as an input.

When the SPC and the SPD are sequentially solved in this order, we obtain a solution $s' = s'_{SPC} \oplus s'_{SPD}$ with cost $c'_{SPC} + c'_{SPD}$, where c'_{SPC} and c'_{SPD} are the costs of each subproblem. Let $U^{C'}_{dm}$ be the quantity of commodity $m \in \mathcal{M}$ brought to each distribution center $d \in \mathcal{V}_D$.

When the SPD and the SPC are sequentially solved in this order we obtain a solution $s'' = s''_{SPC} \oplus s''_{SPD}$ with cost $c''_{SPC} + c''_{SPD}$, where c''_{SPC} and c''_{SPD} are the costs of each subproblem. Let $U^{D''}_{dm}$ be the quantity of commodity $m \in \mathcal{M}$ brought to each distribution center $d \in \mathcal{V}_D$.

Proposition 4.3 *If* $U^{C'}_{dm} \geq U^{D''}_{dm}$, *for all* $d \in \mathcal{V}_D$ *and for all* $m \in \mathcal{M}$, *then the solution* $s = s'_{SPC} \oplus s''_{SPD}$ *is optimal.*

Proof 4.4 *Suppose there exists a feasible solution* s^* *associated with a cost* c^* *such that* $c^* = c^*_{SPC} + c^*_{SPD} < c'_{SPC} + c''_{SPD}$. *Since* $s = s'_{SPC} \oplus s''_{SPD}$ *is feasible* ($U^{D''}_{dm} \leq U^{C'}_{dm}$, *for each* $m \in \mathcal{M}$ *and* $d \in \mathcal{V}_D$) *and* s'_{SPC} *is optimal for the SPC while* s''_{SPD} *is optimal for the SPD, then* $c'_{SPC} \leq c^*_{SPC}$ *for any solution of the SPC. Similarly,* $c''_{SPD} \leq c^*_{SPD}$ *for any solution of the SPD. Thus,* $c'_{SPC} + c''_{SPD} \leq c^*_{SPC} + c^*_{SPD}$.

Proposition 4.5 *Suppose that* $c'_{SPC} + c'_{SPD} = c''_{SPC} + c''_{SPD}$. *Then there is no guarantee that* s' *and* s'' *are optimal.*

Proof 4.6 *Let us consider an instance with one commodity, two distribution centers, two suppliers, and two customers, as depicted in Figure 4.8. The demand of*

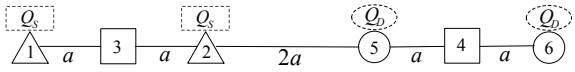


Fig. 4.8. Instance considered for the proof.

each customer is equal to the vehicle capacity Q_D, and the available quantity at each supplier is equal to the truck capacity Q_S, with $Q_D = Q_S$.

When solving first the SPC and then the SPD, we obtain the solution depicted in Figure 4.9. Both suppliers deliver to distribution center 3*, and, as a consequence, both customers are served by this distribution center. For SPC this solution costs* $4a$*, and for SPD it costs* $16a$*. Hence the total cost is* $20a$*.*

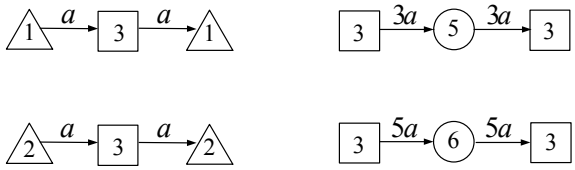


Fig. 4.9. Solution when solving the SPC followed by the SPD.

When solving first the SPD and then the SPC, we obtain the solution depicted in Figure 4.10. Both customers are served by distribution center 4*. Hence both suppliers deliver their commodities to distribution center* 4*. This solution has a cost of* $16a$ *for the SPC and* $4a$ *for the SPD. The total cost is* $20a$*, as for the previous solution. The solution is illustrated in Figure 4.10.*

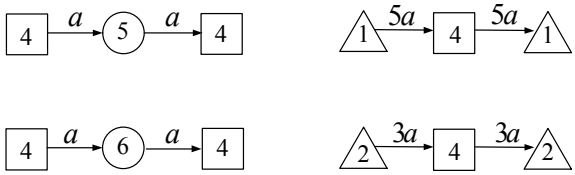


Fig. 4.10. Solution when solving first the SPD then the SPC.

The optimal solution of the MC2DP is depicted in Figure 4.11. The cost is $8a$ *for the SPC and* $8a$ *for the SPD. So the total cost is* $16a$*, that is lower than the cost of the two solutions described above.*

4.3.3 Complexity of special cases of the MC2DP

Proposition 4.7 *The MC2DP is NP-hard when $|\mathcal{M}| = |\mathcal{V}_D| = 1$ and the vehicle capacity is unlimited ($Q_D = \infty$).*

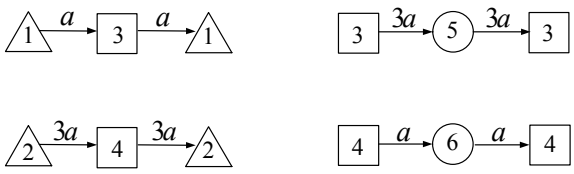


Fig. 4.11. Optimal solution for the instance depicted in Figure 4.8.

Proof 4.8 *When* $|\mathcal{M}| = |\mathcal{V}_D| = 1$ *and vehicles have unlimited capacity, the SPD reduces to the Traveling Salesman Problem (TSP).*

Let us suppose that customers in the SPD are served through direct trips.

Proposition 4.9 *The MC2DP is NP-hard when* $|\mathcal{M}| = |\mathcal{V}_S| = 1$ *and the truck and vehicle capacities are unlimited* ($Q_S = Q_D = \infty$).

Proof 4.10 *The problem reduces to the Uncapacitated Facility Location Problem (UFLP) where the facilities correspond to the distribution centers, the fixed costs of facilities are the direct trip costs between the supplier and the distribution center, and the costs of assigning customers to facilities correspond to the direct trip costs between the distribution centers and the customers. UFLP is shown to be NP-hard by Mirchandani & Francis (1990).*

4.4 Solution approach

In this section we describe the sequential solution approach for the MC2DP. We first describe a solution method for the SPC based on the solution of a mathematical programming model. Then, we propose a solution method for the SPD. It is an Adaptive Large Neighborhood Search (ALNS) algorithm that extends the approach described in Chapter 3. As the solution of the first subproblem solved, whatever it is, has an impact on the solution of the second subproblem, we describe, in Section 4.4.3, different strategies for taking into account this impact when solving the first subproblem.

We will make use of the concept of *customer-commodity*. A *customer-commodity* represents the demand of a customer for a single commodity. A customer may be seen as a union of *customer-commodities*.

4.4.1 Solution of the SPC

The SPC is modelled as a Mixed Integer linear Program (MIP). The formulation, that will be solved to optimality, is based on the following decision variables:

- $x_{sd} \in \mathbb{N}$: the number of trucks sent from supplier $s \in \mathcal{V}_S$ to distribution center $d \in \mathcal{V}_D$;
- y_{di}^m: is equal to 1 if the demand of commodity $m \in \mathcal{M}$ of customer $i \in \mathcal{V}_C$ is assigned to distribution center $d \in \mathcal{V}_D$, and 0 otherwise;
- $q_{sd}^m \in \mathbb{R}_+$: the quantity of commodity $m \in \mathcal{M}$ that supplier $s \in \mathcal{V}_S$ provides to distribution center $d \in \mathcal{V}_D$.

In addition to the notation introduced in Section 4.1, we define as $\bar{c}_{sd} = c_{sd} + c_{ds}$ the direct trip cost between supplier $s \in \mathcal{V}_S$ and distribution center $d \in \mathcal{V}_D$.

The mathematical formulation that follows determines the quantities to transport from suppliers to distribution centers in such a way that all commodities requested by customers are shipped, and the availabilities at the suppliers and the truck capacity are not violated:

$$\min \sum_{s \in \mathcal{V}_S} \sum_{d \in \mathcal{V}_D} \bar{c}_{sd} x_{sd} \tag{4.1}$$

$$\text{s.}t. \sum_{d \in \mathcal{V}_D} y_{di}^m = 1, \qquad \forall\, i \in \mathcal{V}_C, m \in \mathcal{M}, \tag{4.2}$$

$$\sum_{i \in \mathcal{V}_C} D_{im} y_{di}^m \leq \sum_{s \in \mathcal{V}_S} q_{sd}^m, \qquad \forall\, d \in \mathcal{V}_D, m \in \mathcal{M}, \tag{4.3}$$

$$\sum_{s \in \mathcal{V}_S} q_{sd}^m \geq U_{dm}^D, \qquad \forall\, d \in \mathcal{V}_D, m \in \mathcal{M}, \tag{4.4}$$

$$\sum_{m \in \mathcal{M}} q_{sd}^m \leq Q_S x_{sd}, \qquad \forall\, s \in \mathcal{V}_S, d \in \mathcal{V}_D, \tag{4.5}$$

$$\sum_{d \in \mathcal{V}_D} q_{sd}^m \leq O_{sm}, \qquad \forall\, s \in \mathcal{V}_S, m \in \mathcal{M}, \tag{4.6}$$

$$x_{sd} \in \mathbb{N}, \qquad \forall\, s \in \mathcal{V}_S, d \in \mathcal{V}_D, \tag{4.7}$$

$$y_{di}^m \in \{0, 1\}, \qquad \forall\, i \in \mathcal{V}_C, d \in \mathcal{V}_D, m \in \mathcal{M}, \tag{4.8}$$

$$q_{sd}^m \geq 0, \qquad \forall\, s \in \mathcal{V}_S, d \in \mathcal{V}_D, m \in \mathcal{M}. \tag{4.9}$$

The objective function (4.1) minimizes the transportation cost, that is the cost of the direct trips of the trucks used to supply the distribution centers. Constraints (4.2) impose that the demand of each commodity of each customer is assigned to one distribution center. Constraints (4.3) impose that the quantity of each commodity delivered to each distribution center satisfies the demand of the customers assigned to the same distribution center. Note that constraints (4.2) and (4.3) ensure that any solution of the SPC is feasible for the SPD. Constraints (4.4) ensure that the quantity transported to each distribution center is greater than the the lower bound U^D_{dm}. Constraints (4.5) impose that the total volume of all commodities transported from a supplier to a distribution center cannot exceed the capacity of the trucks used on that arc. Constraints (4.6) impose not to exceed the available quantities of commodities at the suppliers. Constraints (4.7)–(4.9) define the variables.

The above formulation may have several equivalent optimal solutions. To break ties among solutions with the same cost but delivering different quantities, we consider a small value ε and add to the objective function (4.1) the term:

$$\varepsilon \sum_{d\in\mathcal{V}_D} \sum_{i\in\mathcal{V}_C} \sum_{m\in\mathcal{M}} c_{di} y^m_{di}. \tag{4.10}$$

With this term, solutions with different assignments of *customer-commodities* to distribution centers are evaluated differently. This term favours solutions that minimize the total distance between *customer-commodities* and the distribution center to which they are assigned.

We call $\mathcal{F}_{SPC}$ the formulation (4.2)-(4.9) with objective function:

$$\min \sum_{s\in\mathcal{V}_S} \sum_{d\in\mathcal{V}_D} \bar{c}_{sd} x_{sd} + \varepsilon \sum_{d\in\mathcal{V}_D} \sum_{i\in\mathcal{V}_C} \sum_{m\in\mathcal{M}} c_{di} y^m_{di}. \tag{4.11}$$

An optimal solution to the formulation $\mathcal{F}_{SPC}$ gives the available quantities U^C_{dm} of each commodity m at distribution center d as:

$$U^C_{dm} = \sum_{s\in\mathcal{V}_S} q^m_{sd}.$$

We will see in Section 4.4.3, how such quantities are used in a sequential approach to the solution of the MC2DP where the SPC is solved first. When the SPD is solved

first, the assignment of *customer-commodities* to distribution centers is known, that is the values of the y variables are known. In this case, constraints (4.2) and (4.3) can be removed from $\mathcal{F}_{SPC}$, and only constraints (4.4) are used to link the two subproblems. It is noteworthy that when the SPC is solved first, the values U^D_{dm} are not set according to the solution of the SPD. Then, constraints (4.2) and (4.3) are necessary to ensure that the solution of the SPC is feasible for the SPD.

4.4.2 Solution of the SPD

The SPD deals with the delivery of commodities from distribution centers to customers. As already mentioned, when the SPD is solved after the SPC, the values U^C_{dm} are known. Instead, when the SPD is solved first, the values U^C_{dm} have to be set. Different strategies to determine these values are proposed in Section 4.4.3.2. In the remaining of this section we assume that these quantities are given, either from the solution of the SPC or from one of the strategies described in Section 4.4.3.1.

In the following, we describe the solution approach we propose for the SPD, which is an Adaptive Large Neighborhood Search (ALNS). Section 4.4.2.1 describe how we build an initial solution while the ALNS is described in Section 4.4.2.2.

4.4.2.1 Initial solution

To solve the SPD, we start by assigning *customer-commodities* to distribution centers. To this end, we solve a Generalized Assignment Problem (GAP) (Ross & Soland (1975), Cattrysse & Van Wassenhove (1992)) which aims at finding the minimum cost assignment of *customer-commodities* to distribution centers so that each *customer-commodity* is assigned to exactly one distribution center. The assignment is subject to the availability of each commodity at the distribution centers. The assignment cost of commodity m for customer i to distribution center d is c_{id} that represents the traveling cost from customer i to distribution center d.

We now present the formulation of the GAP. For each customer $i \in \mathcal{V}_C$, $\mathcal{M}_i = \{m \in \mathcal{M} | D_{im} > 0\}$ represents the set of commodities required by customer i. Let x^m_{id} be the binary variables equal to 1 if and only if commodity $m \in \mathcal{M}_i$ required by customer $i \in \mathcal{V}_C$ is served from distribution center $d \in \mathcal{V}_D$, and 0 otherwise. The GAP is formulated as follows:

$$\min \sum_{i \in \mathcal{V}_C} \sum_{m \in \mathcal{M}_i} \sum_{d \in \mathcal{V}_D} c_{id} x_{id}^m \tag{4.12}$$

$$\text{s.t.} \sum_{i \in \mathcal{V}_C} D_{im} x_{id}^m \leq U_{dm}^C, \qquad \forall\, m \in \mathcal{M}, d \in \mathcal{V}_D, \tag{4.13}$$

$$\sum_{d \in \mathcal{V}_D} x_{id}^m = 1, \qquad \forall\, i \in \mathcal{V}_C, m \in \mathcal{M}_i, \tag{4.14}$$

$$x_{id}^m \in \{0,1\}, \qquad \forall\, i \in \mathcal{V}_C, m \in \mathcal{M}_i, d \in \mathcal{V}_D. \tag{4.15}$$

The objective function (4.12) minimizes the total assignment cost of *customer-commodities* to distribution centers. Constraints (4.13) impose not to exceed the quantity of commodity available at each distribution center. Constraints (4.14) ensure that each *customer-commodity* is assigned to exactly one distribution center. Constraints (4.15) define decision variables.

Given the assignment of *customer-commodities* to distribution centers provided by the solution of the GAP, the set of initial routes is obtained by applying the split algorithm to each distribution center. The split algorithm (Beasley (1983), Prins (2004)) starts from a giant-tour that visits all the customers associated with a distribution center and decomposes the visiting sequence into a set of feasible routes. The solution obtained is the initial solution for the ALNS.

4.4.2.2 The Adaptive Large Neighborhood Search

The basic idea of the ALNS is to improve the current solution by destroying and rebuilding it. We adapt the ALNS algorithm described in Chapter 3 that was designed to solve the Commodity constrained Split Delivery VRP with a single depot (distribution center). The basic scheme of the algorithm is given in Algorithm 3, where $f(s)$ is the objective value of solution s.

The changes made in each of the procedures are described in the following. We refer the reader to Chapter 3 for more details on the original algorithm. As done in Chapter 3, violations of vehicle capacity are allowed and penalized in the cost function when *customer-commodities* are inserted into existing routes. However, a maximum capacity violation on each route is imposed. Hence, it is possible that a *customer-commodity* cannot be inserted in any route of the current solution. In this case, we select one distribution center from which serving this *customer-commodity*

Algorithm 3 The ALNS algorithm.

```
1: s ← initial solution
2: s_best ← initial solution
3: repeat
4:     s ←Destroy and Repair(s)
5:     s ←Local Search(s)
6:     if f(s) < f(s_best) then
7:         s ←Intensification(s)
8:         s ← s_best
9:     end if
10: until stopping criterion is met
```

is feasible with respect to the availability, and we create an additional route. If several distribution centers may be chosen, the one that minimizes the delivery cost is selected. In addition to what done in Chapter 3, we allow violations of the limited quantities U^C_{dm} of commodities available at each distribution center. This violation is penalized in the objective function proportionally to the violation. The penalty rate is indicated by γ. Hence, if a solution violates the available quantities U^C_{dm} by an amount v, then a term γv is added to the cost function. Initially, γ is set to a minimum value γ_{min} that is equal to the cost of the initial solution. The penalty rate γ is then dynamically modified during the search as follows. We keep track of the number of consecutive feasible and infeasible solutions visited during the ALNS algorithm. If E_{inf} infeasible solutions are obtained consecutively, the value of γ is increased to 2γ. Similarly, if E_{feas} feasible solutions are generated consecutively, the value of γ is decreased to $\max\{\gamma_{min}; \gamma/2\}$.

Destroy and Repair This procedure aims at diversifying the search. It relies on a set of removal and insertion operators which iteratively destroy and repair solutions. The removal and insertion operators are selected using a roulette wheel mechanism. The probability of selecting an operator is dynamically influenced by its performance in past iterations (Ropke & Pisinger, 2006). It uses two removal operators (Shaw removal and random removal of customers), and three insertion heuristics of *customer-commodity* based on greedy, regret-2 and regret-3 insertion paradigms.

Local search The Local Search procedure (LS) considers classical operators: insertion, swap and 2-opt for customers or *customer-commodities*. Note that when,

a move is applied on a customer (and not on a *customer-commodity*), it means that all commodities required by the customer are involved in the move. These operators consider moves in the same route, moves between different routes assigned to the same distribution centerand moves between routes assigned to different distribution centers.

Intensification When a new best solution is found, we intensify the search by applying a Mathematical Programming based Operator (MPO). The main goal is to define a new assignment of the visits to a customer i by solving a capacitated facility location problem. The MPO has been introduced in Chapter 3 and we modified it for the solution of the MC2DP. For the sake of clarity, we provide in the following the formulation of the MPO. We introduce the following notation:

- s_i: solution obtained from the current solution by removing all the visits to customer i;
- $\mathcal{R}_i$: set of routes in s_i;
- C_r^i: cost for inserting customer i in route $r \in \mathcal{R}_i$ (cheapest insertion);
- Q_r^i: remaining capacity in route $r \in \mathcal{R}_i$.

The binary decision variables are the following:

$$x_{mr}^i = \begin{cases} 1 & \text{if the delivery of commodity } m \text{ of customer } i \text{ is assigned to route } r \in \mathcal{R}_i; \\ 0 & \text{otherwise.} \end{cases}$$

$$x_r^i = \begin{cases} 1 & \text{if customer } i \text{ is visited in route } r \in \mathcal{R}_i; \\ 0 & \text{otherwise.} \end{cases}$$

The formulation of MPO is the following:

$$(IP_{MPO})\ min \sum_{r\in\mathcal{R}_i} C_r^i x_r^i \tag{4.16}$$

$$s.t. \sum_{r\in\mathcal{R}_i} x_{mr}^i = 1, \qquad \forall\ m \in \mathcal{M}_i \tag{4.17}$$

$$\sum_{m\in\mathcal{M}_i} D_{im} x_{mr}^i \leq Q_r^i x_r^i, \qquad \forall\ r \in \mathcal{R}_i \tag{4.18}$$

$$x_{mr}^i \in \{0,1\}, \qquad \forall\ m \in \mathcal{M}_i, r \in \mathcal{R}_i \tag{4.19}$$

$$x_r^i \in \{0,1\}, \qquad \forall\ r \in \mathcal{R}_i \tag{4.20}$$

The objective function (4.16) aims at minimizing the total insertion cost. Constraints (4.17) require that each commodity is assigned to one route. Constraints (4.18) impose that the total quantity of commodities assigned to a selected vehicle does not exceed its capacity. Constraints (4.19)-(4.20) define the decision variables. (IP_{MPO}) is solved for each $i \in \mathcal{V}_C$ and only the reassignment of visits of customer i associated with the largest cost reduction is implemented.

Formulation (4.16)–(4.20) corresponds to the one proposed in Chapter 3. In order to solve the MC2DP, we add another constraint ensuring that the distribution centers are not overloaded. Specifically, let Q_{dm}^i represent the remaining quantity of commodity m at depot $d \in \mathcal{V}_D$ in s_i. Then, the following constraints are added:

$$D_{im} x_{mr}^i \leq Q_{dm}^i, \quad \forall\ m \in \mathcal{M}_i, r \in \mathcal{R}_i, \tag{4.21}$$

which ensure that the quantity of commodity assigned to each distribution center does not exceed its remaining availability.

4.4.3 Sequential solution approaches

In the previous two sections, we presented how the two subproblems, SPC and SPD, are solved. We now show how we combine the two approaches in order to obtain a solution method for the MC2DP. Different strategies are proposed, all based on the sequential solution of the two subproblems: the SPD and the SPC. In particular, we propose strategies where the SPD is solved first and the SPC second (indicated as SPD $\rightarrow$ SPC), or, vice-versa, first we solve the SPC and then we solve the SPD (indicated as SPC $\rightarrow$ SPD). In both cases, the solution of the first

subproblem determines the quantity of each commodity at each distribution center. The second subproblem takes this information and deals with delivery or collection accordingly. Note that when the first subproblem is solved, the solution is such that the minimization of the transportation cost of that specific problem only is considered, regardless of the other subproblem. This may lead to solutions of poor quality for the MC2DP. As a consequence, we propose different strategies to guide the solution of the first subproblem to obtain better solutions of the MC2DP.

4.4.3.1 Sequential solution: SPD → SPC

When the SPD is solved first, we consider three strategies to determine the values of quantities U^C_{dm} available at the distribution centers. The SPD is then solved through the algorithm presented in Section 4.4.2. Note that a solution of the SPD gives the required quantities U^D_{dm} of each commodity m at each distribution center d, computed as the sum of the demands of the *customer-commodities* assigned to this distribution center. Afterward, SPC is solved by fixing the values of y variables and the required quantities at the depots U^D_{dm} in $\mathcal{F}_{SPC}$ according to the solution of SPD. The three strategies are as follows.

SPD infinite → SPC. In this strategy, the values U^C_{dm} involved in the GAP formulation (Section 4.4.2.1) and in the ALNS algorithm (Section 4.4.2.2) are set to a valid upper bound as follows:

$$U^C_{dm} = \sum_{i \in \mathcal{V}_C} D_{im}, \quad \forall d \in \mathcal{V}_D, m \in \mathcal{M}$$

and do not restrict the search.

SPD finite balanced → SPC. In this strategy, the values U^C_{dm} are restrictive and aim at balancing the quantity of commodities distributed from each distribution center. As a consequence, the quantity of each commodity available at each distribution center is determined as:

$$U^C_{dm} = \frac{\sum_{i \in \mathcal{V}_C} D_{im}}{|\mathcal{V}_D|} + \max_{i \in \mathcal{V}_C} \{D_{im}\}, \quad \forall d \in \mathcal{V}_D, m \in \mathcal{M}.$$

Note that the second term involved in the computation of U^C_{dm} guarantees that the SPD has a feasible solution and it also allows some flexibility in the assignment

of *customer-commodities* to distribution centers.

SPD finite supplier based → SPC. In this strategy, the values U^C_{dm} are determined taking into account the location of the distribution centers and the suppliers. The idea is to compute the available quantities U^C_{dm} at distribution center d based on the available quantities of the suppliers located near to d. First, we assign each supplier to its k closest distribution centers. Then, the quantity U^C_{dm} of commodity m available at distribution center d is computed as the sum of the available quantities O_{sm} for commodity m over all suppliers assigned to distribution center d. Note that if there are few distribution centers, k will take value 1, but when there are several distribution centers, k can be larger than 1 to consider larger values for U^C_{dm}, providing more flexibility for the solution of the SPD.

4.4.3.2 Sequential solution: SPC → SPD

When the SPC is solved first, we consider three strategies. Based on the strategy chosen, we define different values for U^D_{dm}, and then the formulation $\mathcal{F}_{SPC}$ is solved to obtain a solution for SPC. Then, in order to solve the SPD, the U^C_{dm} values are defined from the solution of $\mathcal{F}_{SPC}$ as follows:

$$U^C_{dm} = \sum_{s \in \mathcal{V}_S} q^m_{sd}, \quad \forall d \in \mathcal{V}_D, m \in \mathcal{M}. \tag{4.22}$$

Finally, the SPD is solved. The three strategies considered when the SPC is solved first are listed in the following.

SPC not full truck → SPD. In this strategy, the SPC is first solved using the model $\mathcal{F}_{SPC}$ presented in Section 4.4.1 with values U^D_{dm} set to a non-binding value, i.e.:

$$U^D_{dm} = 0, \quad \forall d \in \mathcal{V}_D, m \in \mathcal{M}.$$

SPC full truck → SPD. In this strategy, the SPC is first solved using the model $\mathcal{F}_{SPC}$ presented in Section 4.4.1 with values U^D_{dm} set to 0 as in the former strategy. Then, the solution of the SPC is updated by using the remaining capacity of the trucks to increase the quantities q^m_{sd} brought to the distribution centers. This does not modify the cost of the SPC, but offers more flexibility when solving the SPD

due to larger availabilities (U^C_{dm}) at distribution centers. More precisely, for each supplier s and each distribution center d, we divide the remaining capacity of the trucks traveling from s to d by the number of commodities available at supplier s. This gives an equal maximum quantity q_{fill} for each commodity in order to fill the trucks. If the availability of some commodity is less than q_{fill}, we fill the trucks as much as we can and we possibly repeat the procedure. Details of the procedure are given in Algorithm 10 in the Appendix. This procedure increases the values of variables q^m_{sd} in formulation $\mathcal{F}_{SPC}$. Then, in order to solve the SPD, U^C_{dm} values are defined as in Equation (4.22).

SPC full truck customer based → SPD. This strategy solves the model $\mathcal{F}_{SPC}$ in which quantities U^D_{dm} are determined on the basis of the locations of the customers. The idea is to compute the quantities U^D_{dm} required at distribution center d based on the demands of the customers located near to the distribution center d and far from other distribution centers. Customers close to several distribution centers are not included in the computation of U^D_{dm} to ensure some flexibility.

Given two distribution centers d_1 and d_2 with a distance a between d_1 and d_2, we say that a customer i is d_1-d_2 compatible if one of the following two conditions is satisfied: (1) the distance between i and d_1 is less than $a/3$; or (2) the distance between i and d_1 is less than a, and the distance between i and d_2 is greater than a. An example is given in Figure 4.12. For each distribution center d, we say that a customer i is assigned to d if for all other distribution centers $d' \in \mathcal{V}_D \setminus \{d\}$, i is d-d' compatible. With the choice of $a/3$, customers at similar distance from two distribution centers are not assigned to any distribution center, as illustrated in Figure 4.12.

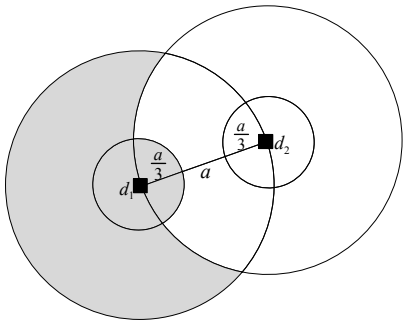


Fig. 4.12. All customers in the grey zone are d_1-d_2 compatible.

Then, the quantity U^D_{dm} is computed as the sum of the demands of commodity $m \in \mathcal{M}$ of customers assigned to $d \in \mathcal{V}_D$. After solving $\mathcal{F}_{SPC}$, the strategy *SPC full truck* $\rightarrow$ *SPD* is applied to increase the quantities q^m_{sd}. Finally, values U^C_{dm} are calculated as in (4.22).

4.5 Computational experiments

In this section, we compare the solution algorithms proposed in Section 4.4.3 to solve the MC2DP sequentially. An algorithm is determined by the sequence in which the two subproblems are solved and the strategy to solve the first subproblem.

In Section 4.5.1, we describe the generation of the sets of instances for the MC2DP based on instances for the C-SDVRP. Then, in Section 4.5.2, we report the results obtained when solving the instances with the different sequential algorithms.

The algorithms have been implemented in C++ and run on an Intel (R) Core(TM) i7-4600U, 2.10GHz, and 16GB of RAM. We summarize the notation used to present the instances and the results in Table 4.1.

Table 4.1: Notation for computational results.

Symbol	Meaning
$nbIns$	Number of instances in each group
$SPDcost$	Best solution cost for SPD
$SPCcost$	Best solution cost for SPC
$Cost$	Total cost for MC2DP
$avg.t(s)$	Average CPU time for computing the MC2DP solution (in seconds)
$avg.SPDt(s)$	Average CPU time for computing the SPD solution (in seconds)
$avg.SPCt(s)$	Average CPU time for computing the SPC solution (in seconds)

4.5.1 Instances

We generate 9 sets of instances, all with 30 customers and 8 suppliers. First, we create a base set of instances that is indicated as $\mathcal{S}$. Then, the 8 other sets are generated by modifying one of the characteristics of set $\mathcal{S}$.

4.5.1.1 Generation of the base set of instances $\mathcal{S}$

The delivery subproblem (SPD) of the MC2DP extends the C-SDVRP by considering multiple distribution centers. Hence, we build the base set of instances from

instances of the C-SDVRP. More precisely, we consider the 64 small instances with 15 customers proposed by Archetti *et al.* (2014). These instances are built from the customer locations of the R101 and C101 Solomon instances for the VRP with Time Windows (Solomon, 1987). Each instance is characterized by five parameters listed thereafter. $I \in \{R101, C101\}$ indicates if the instance is based on R101 or C101. Only the depot and the first 15 customer locations are considered from the Solomon instances. The number of commodities, indicated by M, is equal to 2 or 3. A customer requires a commodity with probability p equal to 0.6 or 1. The quantity of each commodity required by a customer varies within the intervals $\Delta = [1, 100]$ or $\Delta = [40, 60]$. Last, parameter $\alpha \in \{1.1, 1.5, 2, 2.5\}$ determines the vehicle capacity by multiplying the vehicle capacity in the original Solomon instances. We indicate by $\mathcal{P} = (I, M, p, \Delta, \alpha)$ the set of parameters listed above.

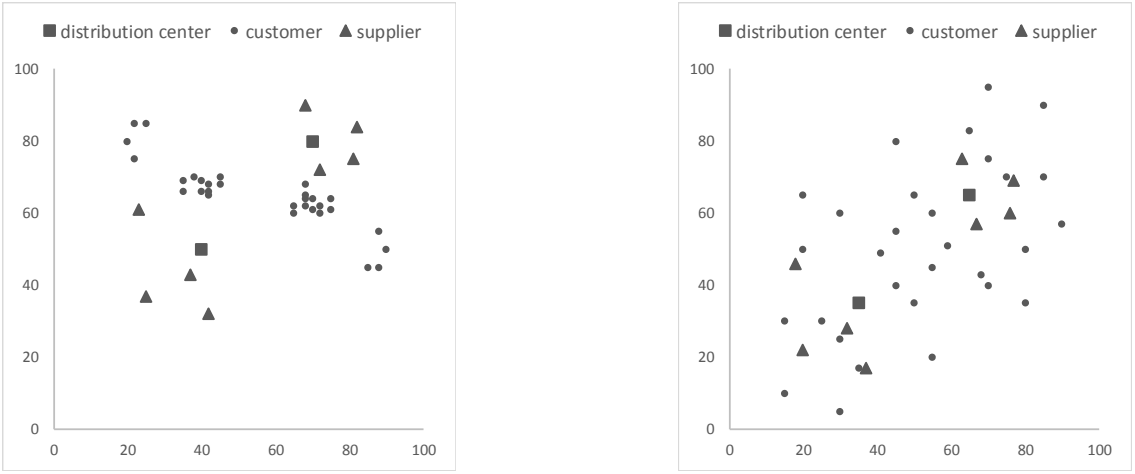
For each instance $\mathcal{I}$ of the C-SDVRP, we create an instance $\mathcal{I}_{MC2DP}$ for the MC2DP as follows. Given the coordinates (x_d, y_d) of the distribution center in $\mathcal{I}$, one distribution center of $\mathcal{I}_{MC2DP}$ is located in (x_d, y_d), while another one is located in $(x_d + \delta, y_d + \delta)$. Given the coordinates (x, y) of a customer in $\mathcal{I}$, one customer of $\mathcal{I}_{MC2DP}$ is located in (x, y), while another customer is located in $(-x + 2x_d + \delta, -y + 2y_d + \delta)$. Both customers have the same demand as in $\mathcal{I}$. Hence, the instance $\mathcal{I}_{MC2DP}$ contains one distribution center d_1 and a set $\mathcal{V}_C^1$ of 15 customers with the same locations as instance $\mathcal{I}$. It contains as well another distribution center d_2 and a second set $\mathcal{V}_C^2$ of 15 customers. Customers in $\mathcal{V}_C^2$ are transposed by (δ, δ) from the original locations, and are also rotated by 180 degrees around the distribution center d_2. After some preliminary experiments detailed in Appendix B.2, we set $\delta = 30$.

To locate the 8 suppliers we proceeded as follows. We consider a circle of radius r centered at each distribution center. We randomly created inside each circle four suppliers. Should the two circles intersect, the intersection is not considered as a potential zone to locate suppliers. The value of the radius r has been fixed to 30. Note that the cost c_{ij} associated with each arc $(i, j) \in \mathcal{A}$ is equal to the Euclidean distance between i and j.

The quantity of commodity $m \in \mathcal{M}$ available at each supplier is calculated as $O_{sm} = \lceil \zeta \cdot \frac{\sum_{i \in \mathcal{V}_C} D_{im}}{|\mathcal{V}_S|} \rceil$, where ζ is a parameter that has been fixed to 1.2. Hence, all suppliers produce all commodities with the same amount. Globally, the total supply is 20% more than the total demand.

The capacities Q_S and Q_D of the trucks and vehicles are equal to the capacity of vehicles in $\mathcal{J}$.

Figure 4.13 shows the locations of customers, distribution centers and suppliers for C101 and R101 configurations.



(a) Locations in instances obtained from C101. (b) Locations in instances obtained from R101.

Fig. 4.13. Locations in instances in $\mathcal{S}$.

In order to ensure the diversity of the instances to evaluate the proposed sequential approaches, we construct 8 additional sets of instances based on instances in $\mathcal{S}$. Each of the 8 sets differs from $\mathcal{S}$ by the modification of one characteristic.

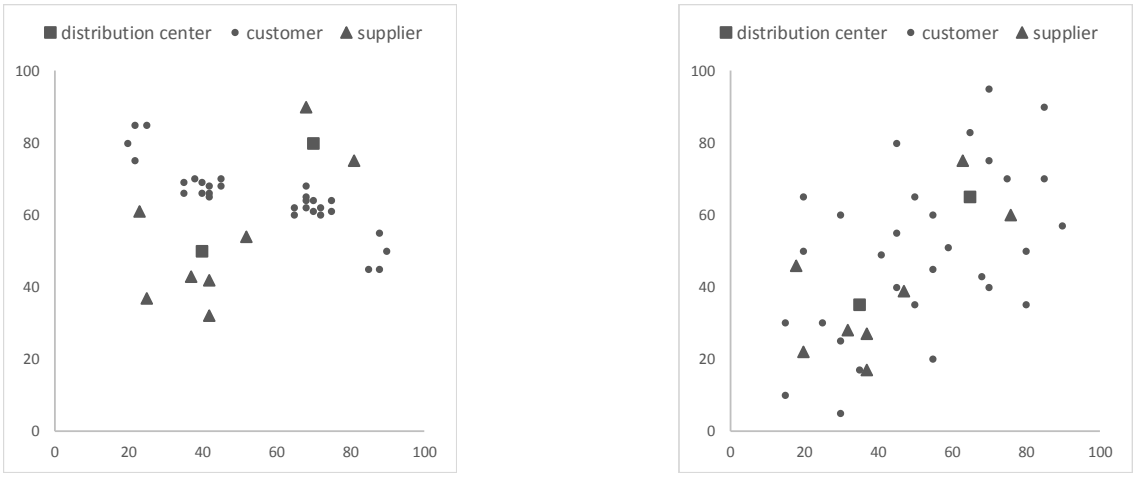
4.5.1.2 Modification of the supplier locations

In set $\mathcal{S}$, four suppliers are generated around each distribution center. We generate two other sets of instances where we modify the location of suppliers by unbalancing the number of suppliers generated around each distribution center. These two sets of instances are named $\mathcal{S}_1^S$ and $\mathcal{S}_2^S$. As for instance set $\mathcal{S}$, we consider a circle of radius $r = 30$ centered at each distribution center, and no supplier is located in the intersection of the two circles. The characteristics of the new sets of instances are the following.

$\mathcal{S}_1^S$: Around distribution center d_1 we randomly locate 6 suppliers. The other two suppliers are randomly located around d_2.

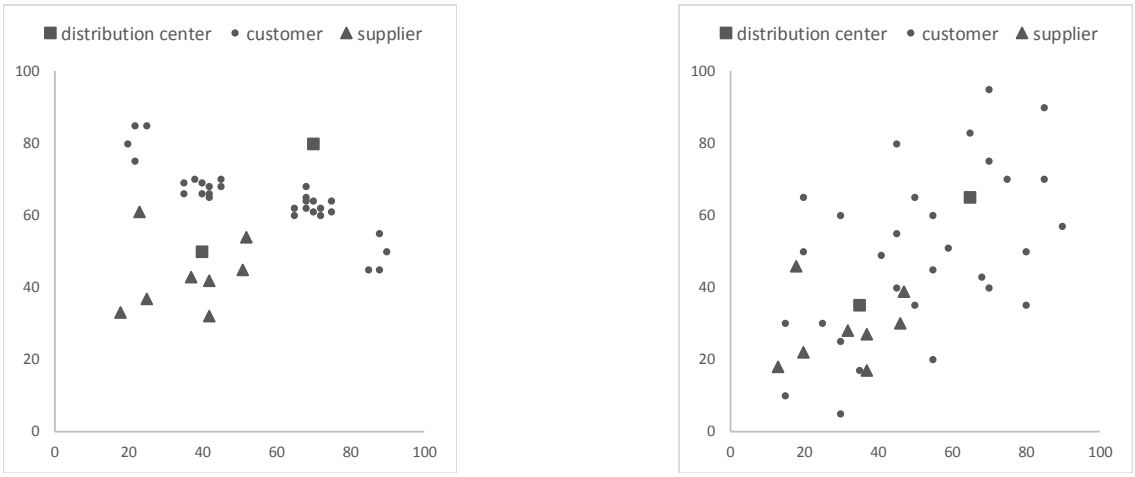
$\mathcal{S}_2^S$: Around distribution center d_1 we randomly locate 8 suppliers. No supplier is generated around d_2.

Examples of the two configurations are presented in Figures 4.14 and 4.15.



(a) Locations in instances obtained from C101. (b) Locations in instances obtained from R101.

Fig. 4.14. Locations in instances of $\mathcal{S}_1^S$.



(a) Locations in instances obtained from C101. (b) Locations in instances obtained from R101.

Fig. 4.15. Locations in instances of $\mathcal{S}_2^S$.

4.5.1.3 Modification of the customer locations

In set $\mathcal{S}$, each distribution center is surrounded by 15 customers. This makes the instances *balanced* from the point of view of the customer locations with respect

to the distribution centers. We generate four sets of instances where we modified the location of customers by unbalancing the number of customers around each distribution center. These four sets of instances are named $\mathcal{S}_1^C$, $\mathcal{S}_2^C$, $\mathcal{S}_3^C$ and $\mathcal{S}_4^C$. As with instance set $\mathcal{S}$, given the coordinates (x, y) of a customer in $\mathcal{I}$, one customer of $\mathcal{I}_{MC2DP}$ is located in (x, y), while another customer is located in $(-x+2x_d+\delta, -y+2y_d+\delta)$. For instances in $\mathcal{S}$, the value $\delta = 30$ has been considered. For the new sets of instances, the locations of the 15 duplicated customers are defined as follows.

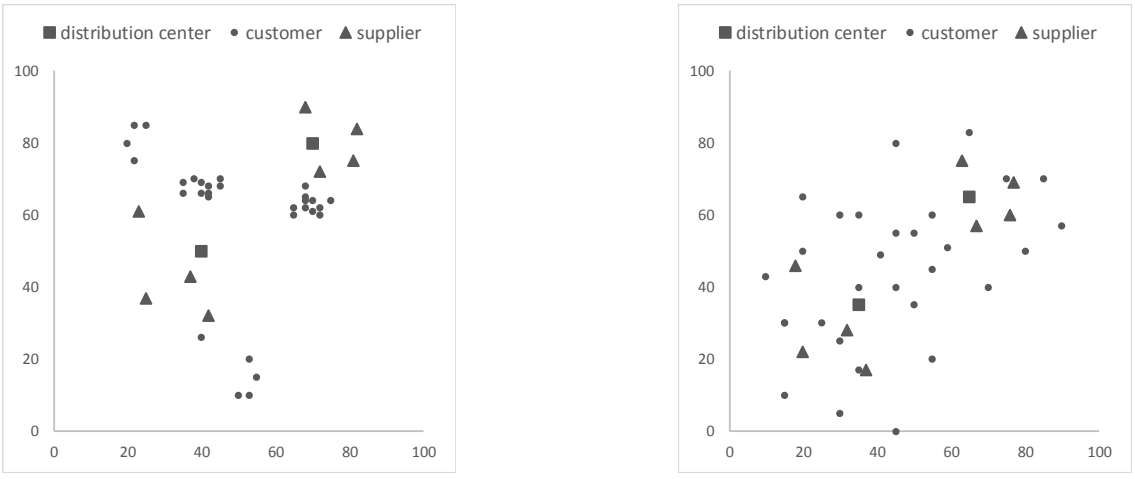
$\mathcal{S}_1^C$: For the first 5 customers in $\mathcal{I}$, $\delta = -5$; for the remaining 10 customers, $\delta = 30$.

$\mathcal{S}_2^C$: For the first 5 customers in $\mathcal{I}$, $\delta = 10$; for the remaining 10 customers, $\delta = 30$.

$\mathcal{S}_3^C$: For the first 10 customers in $\mathcal{I}$, $\delta = -5$; for the remaining 5 customers, $\delta = 30$.

$\mathcal{S}_4^C$: For the first 10 customers in $\mathcal{I}$, $\delta = 10$; for the remaining 5 customers, $\delta = 30$.

An illustration of instances in $\mathcal{S}_1^C$ is given in Figure 4.16.



(a) Locations in instances obtained from C101. (b) Locations in instances obtained from R101.

Fig. 4.16. Locations in instances of $\mathcal{S}_1^C$.

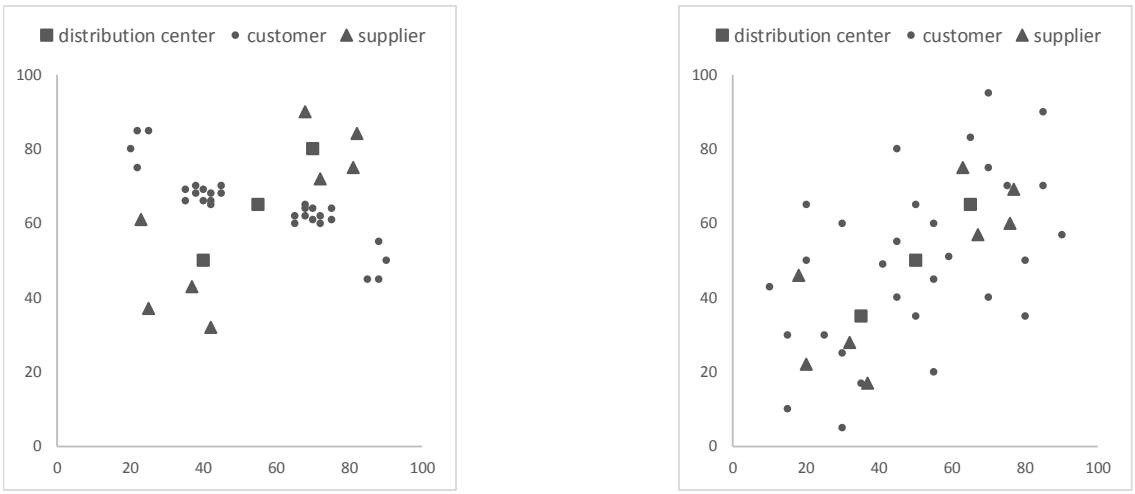
4.5.1.4 Modification of the available quantities at suppliers

In set $\mathcal{S}$, given a commodity $m \in \mathcal{M}$, all suppliers $s \in \mathcal{V}_S$ have the same available quantity $O_{sm} = \lceil \zeta \cdot \frac{\sum_{i \in \mathcal{V}_C} D_{im}}{|\mathcal{V}_S|} \rceil$, with $\zeta = 1.2$. We generate a set of instances $\mathcal{S}^O$

where we unbalance the available quantities at suppliers. For this set of instances, we only consider the 32 instances where the number of commodities is $|\mathcal{M}|$=2. Half of the suppliers, all located around the same distribution center, have $O_{s1} = \lceil \zeta \cdot \frac{1.8 \sum_{i \in \mathcal{V}_C} D_{i1}}{|\mathcal{V}_S|} \rceil$ units of commodity 1 and $O_{s2} = \lceil \zeta \cdot \frac{0.2 \sum_{i \in \mathcal{V}_C} D_{i2}}{|\mathcal{V}_S|} \rceil$ units of commodity 2. On the contrary, the other suppliers, all located around the other distribution center, have $O_{s1} = \lceil \zeta \cdot \frac{0.2 \sum_{i \in \mathcal{V}_C} D_{i1}}{|\mathcal{V}_S|} \rceil$ units of commodity 1 and $O_{s2} = \lceil \zeta \cdot \frac{1.8 \sum_{i \in \mathcal{V}_C} D_{i2}}{|\mathcal{V}_S|} \rceil$ units of commodity 2. ζ is still fixed to 1.2.

4.5.1.5 Modification of the number of distribution centers

The instances in $\mathcal{S}$ contain two distribution centers. For each instance in $\mathcal{S}$, we create an instance in $\mathcal{S}^D$ by adding a third distribution center located in the middle of the segment joining the two distribution centers. More precisely, the two distribution centers are located in (x_d, y_d) and $(x_d + \delta, y_d + \delta)$. Hence, the third distribution center is located in $(x_d + 0.5\delta, y_d + 0.5\delta)$, as illustrated in Figure 4.17.



(a) Locations in instances obtained from C101. (b) Locations in instances obtained from R101.

Fig. 4.17. Locations in instances of $\mathcal{S}^D$

4.5.2 Comparison of the sequential approaches to solve the MC2DP

The six algorithms presented in Section 4.4.3 for the solution of the MC2DP are tested on the 9 instance sets presented in Section 4.5.1. The mathematical programs presented in Section 4.4.2 are solved using Cplex 12.6. The ALNS algorithm

is run with a limit of 5000 iterations and the values of the parameters used in Chapter 3. The additional parameters introduced in Section 4.4.2.2 are set to $E_{inf} = 50$ and $E_{feas} = 50$ after preliminary experiments. We recall that the value of γ_{min} equals the cost of the initial solution. Moreover, for the algorithm $SPD\ finite\ supplier\ based \rightarrow SPC$, we set $k = 1$ since the instances contain few distribution centers.

In order to compare the algorithms, for each instance set we report:

- $\Delta avg.SPCcost$: percentage difference between the average cost for collection ($avg.SPCcost$) and the minimum average cost for collection ($minavg.SPCcost$), i.e. $avg.\Delta_{SPCcost} = 100*(avg.SPCcost - minavg.SPCcost)/minavg.SPCcost$;
- $\Delta avg.SPDcost$: percentage difference between the average cost for delivery ($avg.SPDcost$) and the minimum average cost for delivery ($minavg.SPDcost$), i.e. $avg.\Delta_{SPDcost} = 100*(avg.SPDcost - minavg.SPDcost)/minavg.SPDcost$;
- $\Delta avg.Cost$: percentage difference between the average total cost ($avg.Cost$) and the minimum average total cost ($minavg.Cost$), i.e. $\Delta avg.Cost = 100 * (avg.Cost - minavg.Cost)/minavg.Cost$.

We also report the minimum average results for the collection cost, the delivery cost and the total cost in the last line of each table.

Table 4.2 reports the results for the instance set $\mathcal{S}$. The best value obtained for MC2DP, SPC and SPD are provided in bold. The table shows that the best algorithm is $SPD\ finite\ balanced \rightarrow SPC$. The other algorithms that solve first SPD provide also good results, always better than the algorithms where SPC is solved first. The worst algorithm is $SPC\ not\ full\ truck \rightarrow SPD$, with an average increase in total cost of 2.35%. We also note that, when solving SPD first, we have an increase in the cost of SPC of more than 8%. Instead, when solving first SPC, the increase on SPD cost goes from 4.49% to 7.54%. However, the algorithms solving SPD first provide better values of total cost. This is due to the unbalance in the total cost of SPC and SPD. Finally, it is interesting to note that the algorithm providing the best total cost is not the best for none of the two subproblems. On average, the CPU times are around 1 minute. Since computational times on the other sets of instances are similar, we do not report them in the following Tables.

Table 4.2: Average results on instance set $\mathcal{S}$.

Strategy	$\Delta avg.SPCcost$	$\Delta avg.SPDcost$	$\Delta avg.Cost$	$avg.SPCt(s)$	$avg.SPDt(s)$	$avg.t(s)$
SPD infinite → SPC	8.60	0.01	0.02	0.15	46.84	46.99
SPD finite balanced → SPC	8.55	0.01	**0.00**	0.15	54.27	54.42
SPD finite supplier based → SPC	8.71	**0.00**	0.04	0.16	50.59	50.75
SPC not full truck → SPD	**0.00**	7.54	2.35	0.26	81.23	81.49
SPC full truck → SPD	**0.00**	4.74	0.49	0.27	73.21	73.47
SPC full truck customer based → SPD	**0.00**	4.49	0.33	0.27	74.96	75.23
minimum average costs	**331.92**	**711.99**	**1072.34**	-	-	-

Table 4.3: Average results on instance sets $\mathcal{S}_1^S$ and $\mathcal{S}_2^S$.

Instance set	$\mathcal{S}_1^S$			$\mathcal{S}_2^S$		
Strategy	$\Delta avg.SPCcost$	$\Delta avg.SPDcost$	$\Delta avg.Cost$	$\Delta avg.SPCcost$	$\Delta avg.SPDcost$	$\Delta avg.Cost$
SPD infinite → SPC	45.36	0.00	6.55	100.57	**0.00**	9.78
SPD finite balanced → SPC	45.05	**0.00**	6.45	98.33	**0.00**	9.15
SPD finite supplier based → SPC	3.94	9.01	**0.00**	**0.00**	34.86	**0.00**
SPC not full truck → SPD	**0.00**	17.57	4.27	**0.00**	34.86	**0.00**
SPC full truck → SPD	**0.00**	13.44	1.65	**0.00**	34.89	0.01
SPC full truck customer based → SPD	7.01	9.04	0.93	42.52	12.62	0.14
minimum average costs	**331.92**	**712.04**	**1121.17**	**376.85**	**712.08**	**1337.16**

Table 4.4: Average results on instance sets $\mathcal{S}_1^C$ and $\mathcal{S}_2^C$.

Instance set	$\mathcal{S}_1^C$			$\mathcal{S}_2^C$		
Strategy	$\Delta avg.SPCcost$	$\Delta avg.SPDcost$	$\Delta avg.Cost$	$\Delta avg.SPCcost$	$\Delta avg.SPDcost$	$\Delta avg.Cost$
SPD infinite → SPC	28.61	**0.00**	3.24	25.47	**0.00**	2.78
SPD finite balanced → SPC	11.75	2.85	**0.00**	11.77	2.30	0.04
SPD finite supplier based → SPC	14.78	1.48	0.03	13.64	1.35	**0.00**
SPC not full truck → SPD	**0.00**	14.62	4.05	**0.00**	12.82	3.15
SPC full truck → SPD	**0.00**	10.11	1.14	**0.00**	8.42	0.33
SPC full truck customer based → SPD	0.84	8.85	0.58	0.23	8.24	0.29
minimum average costs	**331.92**	**710.99**	**1102.21**	**331.92**	**690.85**	**1077.35**

Table 4.5: Average results on instance sets $\mathcal{S}_3^C$ and $\mathcal{S}_4^C$.

Instance set	$\mathcal{S}_3^C$			$\mathcal{S}_4^C$		
Strategy	$\Delta avg.SPCcost$	$\Delta avg.SPDcost$	$\Delta avg.Cost$	$\Delta avg.SPCcost$	$\Delta avg.SPDcost$	$\Delta avg.Cost$
SPD infinite → SPC	65.70	**0.00**	9.63	55.71	**0.00**	8.95
SPD finite balanced → SPC	12.46	9.85	0.46	11.87	6.84	**0.00**
SPD finite supplier based → SPC	15.79	7.56	**0.00**	15.30	5.22	0.03
SPC not full truck → SPD	**0.00**	23.99	5.67	**0.00**	18.89	3.90
SPC full truck → SPD	**0.00**	19.44	2.84	**0.00**	14.63	1.26
SPC full truck customer based → SPD	10.20	12.15	1.25	0.72	13.58	0.82
minimum average costs	**331.92**	**717.99**	**1156.61**	**331.92**	**684.00**	**1102.15**

Results on instance sets $\mathcal{S}_1^S$ and $\mathcal{S}_2^S$ with modification of the supplier locations are reported in Table 4.3. We note that, among the three algorithms where the SPD is solved first, the algorithm $SPD\ finite\ supplier\ based \rightarrow SPC$ provides the highest cost for the SPD. However this is compensated by the lowest cost for the SPC resulting in the lowest cost for the whole problem. When supplier locations around the distribution centers are unbalanced, the algorithm $SPD\ finite\ supplier\ based \rightarrow SPC$ is very efficient in comparison with the other algorithms that solve first the SPD. We also note that, for this set of instances, the sequence in which the subproblems are solved has a significant impact with respect to the instances in set $\mathcal{S}$, providing a larger increase in the cost of the second subproblem, especially on instances $\mathcal{S}_2^S$. The best algorithm is $SPD\ finite\ supplier\ based \rightarrow SPC$. On $\mathcal{S}_2^S$, $SPC\ not\ full\ truck \rightarrow SPD$ has the same performance as $SPD\ finite\ supplier\ based \rightarrow SPC$. However, we note that the remaining algorithms in which SPC is solved first perform better than the remaining algorithms where SPD is solved first, especially on $\mathcal{S}_2^S$.

Results on sets $\mathcal{S}_1^C$, $\mathcal{S}_2^C$, $\mathcal{S}_3^C$, and $\mathcal{S}_4^C$ are reported in Tables 4.4 and 4.5. These instances have unbalanced customer locations around the distribution centers. For these four sets, the algorithms $SPD\ finite\ balanced \rightarrow SPC$ and $SPD\ supplier\ based \rightarrow SPC$ provide the best results for the whole problem. The total costs achieved with these two algorithms are very similar. It can be noticed that when applying the algorithm $SPD\ infinite \rightarrow SPC$, the results are not competitive with the two other algorithms where the SPD is solved first: the cost of the SPD is smaller but the cost of the SPC increases a lot, leading to a high value of the total cost. Moreover, for the three algorithms where the SPC is solved first, we observe a similar behaviour for these sets of instances and the base set $\mathcal{S}$. The only difference is that the algorithm $SPC\ full\ truck\ customer\ based \rightarrow SPD$ provides the best results among the three algorithms where the SPC is solved first. Note that when the SPC is solved first and the customer locations are not taken into account (algorithms $SPC\ not\ full\ truck \rightarrow SPD$ and $SPC\ full\ truck \rightarrow SPD$), the solutions that are obtained for the SPC are always the same and have the lowest SPC cost for all the considered instances. However, the overall solution is never the best. This clearly shows the importance of taking into account the location of final customers when bringing commodities to distribution centers.

Table 4.6: Average results on instance set $\mathcal{S}^O$.

Strategy	$\Delta avg.SPCcost$	$\Delta avg.SPDcost$	$\Delta avg.Cost$
$SPD\ infinite \rightarrow SPC$	112.24	**0.00**	9.06
$SPD\ finite\ balanced \rightarrow SPC$	113.50	0.03	9.40
$SPD\ finite\ supplier\ based \rightarrow SPC$	4.25	38.77	2.09
$SPC\ not\ full\ truck \rightarrow SPD$	**0.00**	45.59	4.68
$SPC\ full\ truck \rightarrow SPD$	**0.00**	43.39	3.49
$SPC\ full\ truck\ customer\ based \rightarrow SPD$	40.16	17.74	**0.00**
minimum average costs	**345.76**	**722.98**	**1335.83**

Table 4.7: Average results on instance set $\mathcal{S}^D$.

Instance set	$\mathcal{S}^D$			Difference between $\mathcal{S}^D$ and $\mathcal{S}$ (%)		
Strategy	$\Delta avg.SPCcost$	$\Delta avg.SPDcost$	$\Delta avg.Cost$	$\Delta avg.SPCcost$	$\Delta avg.SPDcost$	$\Delta avg.Cost$
$SPD\ infinite \rightarrow SPC$	64.09	**0.00**	11.47	51.09	-8.43	11.57
$SPD\ finite\ balanced \rightarrow SPC$	40.12	1.45	4.94	29.08	-7.09	5.06
$SPD\ finite\ supplier\ based \rightarrow SPC$	8.93	9.19	**0.00**	0.20	0.00	0.07
$SPC\ not\ full\ truck \rightarrow SPD$	**0.00**	20.00	3.81	0.00	2.20	1.53
$SPC\ full\ truck \rightarrow SPD$	**0.00**	15.30	0.95	0.00	0.81	0.56
$SPC\ full\ truck\ customer\ based \rightarrow SPD$	**5.88**	14.68	2.39	5.88	0.52	2.17
minimum average costs	**331.92**	**652.05**	**1073.55**	-	-	-

[1] The last three columns represent the percentage of the corresponding difference between the instance set $\mathcal{S}^D$ and instance set $\mathcal{S}$.

Table 4.6 presents the results obtained on set $\mathcal{S}^O$ where the quantities available at the suppliers are unbalanced. The results indicate that in this case, the algorithm $SPC\ full\ truck\ customer\ based \rightarrow SPD$ provides the lowest total cost. Hence, when the suppliers have unbalanced available quantities, starting by solving the SPC in a sequential approach is the best choice.

The results on set $\mathcal{S}^D$ are provided in Table 4.7. In these instances, a third distribution center is added in between the other two distribution centers. In addition to the measures provided in previous tables, in Table 4.7 we report, in the last three columns, the percentage difference between the cost of the solution with three distribution centers with respect to the one with two distribution centers for SPC, SPD and total cost, respectively. The lowest total cost is achieved with the algorithm $SPD\ finite\ supplier\ based \rightarrow SPC$. Note that, compared to the results on the base instance set $\mathcal{S}$, adding a distribution center is beneficial to reduce the cost of SPD. This is the case for the algorithms $SPD\ infinite \rightarrow SPC$ and $SPD\ finite\ balanced \rightarrow SPC$. However, for these algorithms, the cost of SPC increases remarkably, which leads to higher total cost. As shown in Figure 4.17, this is due to the distance of the third depot from all suppliers. Thus, when solving

SPD first, the solution might assign some customers to the distribution center in the middle, and this goes to the detriment of the solution cost of SPC. Thus, a sequential solution approach shows, in this case, its limitations. We will further comment about this issue in the conclusions.

Results on the different sets of randomly generated instances reveal that $SPD\ finite\ supplier\ based \rightarrow SPC$ and $SPC\ full\ truck\ customer\ based \rightarrow SPD$ are the best algorithms. As a rule, $SPD\ finite\ supplier\ based \rightarrow SPC$ identifies the best solutions in most cases except when the quantities available at the suppliers are unbalanced. In the latter case, $SPD\ finite\ supplier\ based \rightarrow SPC$ has a better performance.

4.6 A case study

In this section, we present a study on instances generated from a real case application. The aim is to show how the sequential solution approaches behave on real-case applications.

4.6.1 Context

The case study was proposed by local authorities of the French department of Isère (General Council and Chamber of Agriculture). The aim is to increase the volume of fresh food products sold through short and local supply chains in the department of Isère. The department of Isère is an interesting location for fresh food supply chains since: i) there are many farmers producing a variety of fruits and vegetables, ii) these farmers have very low revenues when selling their products through classical distribution channels, iii) the inhabitants of the department are deeply interested in buying local fruits and vegetables through short and local supply chains.

Local authorities of Isère identified two kinds of customers for short and local supply chains: school canteens and supermarkets. For such customers, considering direct deliveries from the farmers is not acceptable as farmers should devote a significant part of their working time to carrying out deliveries. Hence, the idea is to use a set of distribution centers which would be jointly managed by associations of farmers.

4.6.2 Description of the data sets

In this case study, the commodities are fresh fruits and vegetables: apple, pear, kiwi, strawberry, carrot, lettuce, tomato, zucchini, cucumber, potato. Because of the seasonality of these products, demand and supply are not constant over the year. Hence, we consider a set of patterns for demand and supply.

Two sets of customers are considered: school canteens and supermarkets. A study conducted with the General Council of Isère has permitted to collect and estimate actual demands for school canteens. The demand associated with supermarkets has instead been generated according to the following procedure. The selected supermarkets are the ones selling fresh food products with a sales area of at least $300m^2$. Demands for each supermarket have been generated under the following hypotheses: i) the demands for each product has the same proportion as for school canteens (i.e. people have the same kind of food consumption at home and in school canteens), ii) demands are proportional to the sales area of the supermarket, and iii) the global demand of all commodities is proportional to the global revenue generated by the sales of the considered fresh products through supermarkets, for which the data are provided by a statistical study from the General Council of Isère. For school canteens, 8 patterns of demands and supply are considered, and for supermarkets 10 patterns are considered. Hence, there is a set of 18 instances for the case study.

The instances are divided in two sets: one considering canteens and one considering supermarkets.

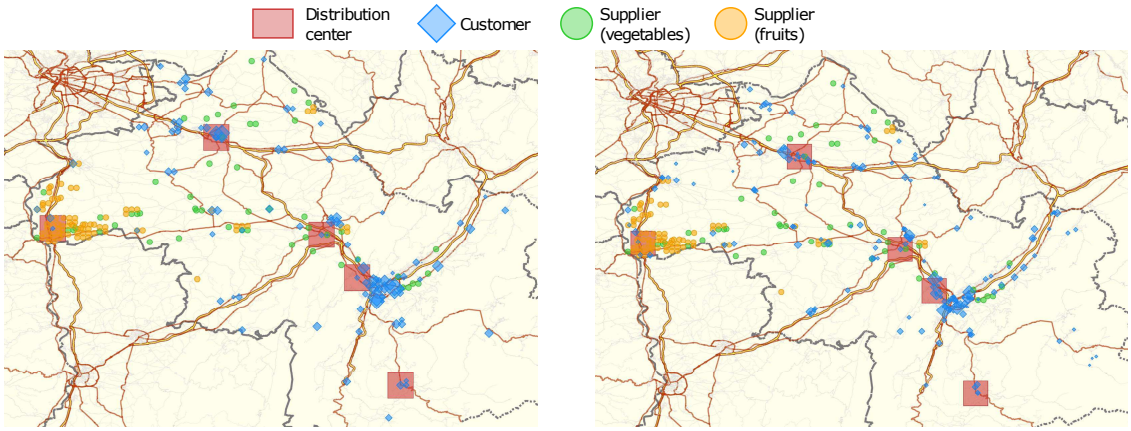
In the school canteens instances, there are 103 customers and up to 61 suppliers. For each pattern of supply and demand, the number of commodities ranges from 5 to 8, and the number of suppliers producing at least one commodity ranges from 54 to 61. In the supermarket instances, there are 188 customers and up to 61 suppliers. For each pattern of supply and demand, the number of commodities ranges from 5 to 8, and the number of suppliers producing at least one commodity ranges from 45 to 61. In all instances, there are 5 distribution centers. Their locations have been proposed by the General Council and the Chamber of Agriculture of Isère. The sizes of the instances are reported in Tables 4.8 and 4.9. The locations of customers, suppliers and distribution centers in the department of Isère are shown in Figure 4.18. The capacity of the vehicles is set such that $Q_D = 2Q_S$, since farmers in short and local supply chains generally do not have large capacity vehicles while distribution centers may invest in vehicles with bigger capacity to visit several customers.

instance id	$\lvert\mathcal{V}_C\rvert$	$\lvert\mathcal{V}_S\rvert$	$\lvert\mathcal{V}_D\rvert$	$\lvert\mathcal{M}\rvert$
1	103	61	5	8
2	103	61	5	7
3	103	61	5	6
4	103	61	5	5
5	103	61	5	6
6	103	61	5	5
7	103	61	5	6
8	103	54	5	7

Table 4.8: School canteens instances.

instance id	$\lvert\mathcal{V}_C\rvert$	$\lvert\mathcal{V}_S\rvert$	$\lvert\mathcal{V}_D\rvert$	$\lvert\mathcal{M}\rvert$
1	188	61	5	6
2	188	61	5	5
3	188	61	5	6
4	188	54	5	7
5	188	45	5	5
6	188	58	5	6
7	188	61	5	8
8	188	61	5	7
9	188	61	5	7
10	188	61	5	6

Table 4.9: Supermarkets instances.



(a) Locations in school canteens instances (b) Locations in supermarkets instances

Fig. 4.18. Locations in the case study instances.

4.6.3 Analysis of the results

The instances are very large with respect to the ones described in Section 4.5.1 since there are up to 1500 *customer-commodities*. The computational time of the ALNS significantly increases with the size of the instances. Hence, when solving the SPD, the ALNS algorithm is run with a limit of 100 iterations. Moreover, due to the size of the instances, we set a time limit for solving Mixed Integer linear Programs. More specifically, we set a 5 minutes time limit to solve the GAP model presented in Section 4.4.2.1 to obtain an initial solution of the SPD, and a 10 minutes time limit to solve the formulation $\mathcal{F}_{SPC}$. As there are 5 distribution centers, when applying the algorithm $SPD\ finite\ supplier\ based \rightarrow SPC$, we set $k = 2$, i.e., each supplier is assigned to its two nearest distribution centers.

The results obtained for each sequential approach are reported in Table 4.10 for the school canteens and in Table 4.11 for the supermarkets. The columns nbT and nbV indicate the numbers of trucks used for collection operations and the number of vehicles used for delivery operations, respectively. The columns $\Delta SPCcost$, $\Delta SPDcost$ and $\Delta Cost$ report the percentage difference between the SPC cost and the minimum SPC cost, the SPD cost and the minimum SPD cost, the total cost and the minimum total cost, respectively.

When customers are school canteens, two algorithms outperform the others, namely: $SPC\ full\ truck \rightarrow SPD$ and $SPD\ finite\ supplier\ based \rightarrow SPC$. It is interesting to note that $SPC\ full\ truck \rightarrow SPD$ provides the best results on average. However, this algorithm gives low collection costs and high delivery costs. This is an interesting observation: in fact, local authorities might judge as inappropriate a solution where the cost is highly unbalanced, even if it provides the best cost for the overall system. On the contrary, the algorithm $SPD\ finite\ supplier\ based \rightarrow SPC$ is a bit more costly on average, but has the advantage of balancing the costs between collection and delivery. It can also be noticed that the algorithms $SPD\ finite\ balanced \rightarrow SPC$ and $SPC\ full\ truck\ customer\ based \rightarrow SPD$ do not provide good solutions. This could be explained by the fact that suppliers and customers are not located homogeneously around the distribution centers.

For the instances with supermarkets, two algorithms outperform the others, namely: $SPD\ finite\ supplier\ based \rightarrow SPC$ and $SPD\ infinite \rightarrow SPC$. $SPD\ finite\ supplier\ based \rightarrow SPC$ provides the best results on 7 instances out of 10, and has the advantage of balancing the costs between collection and delivery operations. It

Table 4.10: Detailed results for school canteens instances.

instance id	Strategy	$\Delta SPCcost$	$\Delta SPDcost$	$\Delta Cost$	$SPCt(s)$	$SPDt(s)$	$t(s)$	nbT	nbV
1	$SPD\ infinite \rightarrow SPC$	144.24	**0.00**	9.76	9.57	683.38	693.41	45	15
	$SPD\ finite\ balanced \rightarrow SPC$	144.73	6.62	11.96	219.67	677.39	898.69	45	16
	$SPD\ finite\ supplier\ based \rightarrow SPC$	78.58	38.27	0.40	4.94	1256.32	1262.17	44	15
	$SPC\ not\ full\ truck \rightarrow SPD$	**0.00**	174.72	17.23	7.42	4763.64	4771.05	41	15
	$SPC\ full\ truck \rightarrow SPD$	**0.00**	119.01	**0.00**	6.81	1981.80	1988.61	41	16
	$SPC\ full\ truck\ customer\ based \rightarrow SPD$	18.42	121.12	6.60	46.42	2463.34	2509.75	41	15
	minimum costs	**928.66**	**889.69**	**2877.15**	-	-	-	-	-
2	$SPD\ infinite \rightarrow SPC$	143.59	**0.00**	13.71	36.51	397.44	434.31	45	14
	$SPD\ finite\ balanced \rightarrow SPC$	155.49	8.91	20.54	118.95	490.81	611.13	45	14
	$SPD\ finite\ supplier\ based \rightarrow SPC$	73.51	41.52	3.24	4.23	844.34	849.31	42	14
	$SPC\ not\ full\ truck \rightarrow SPD$	**0.00**	188.16	24.87	7.30	2883.20	2890.50	41	16
	$SPC\ full\ truck \rightarrow SPD$	**0.00**	109.56	**0.00**	6.62	1576.24	1582.87	41	14
	$SPC\ full\ truck\ customer\ based \rightarrow SPD$	15.80	104.43	3.70	10.53	1562.47	1573.00	42	15
	minimum costs	**928.66**	**872.17**	**2756.37**	-	-	-	-	-
3	$SPD\ infinite \rightarrow SPC$	199.86	**0.00**	11.48	2.56	373.58	376.46	32	8
	$SPD\ finite\ balanced \rightarrow SPC$	245.65	1.43	23.07	3.57	270.39	275.09	35	9
	$SPD\ finite\ supplier\ based \rightarrow SPC$	108.46	26.80	**0.00**	1.80	671.94	674.45	33	9
	$SPC\ not\ full\ truck \rightarrow SPD$	**0.00**	166.24	28.66	3.03	2324.59	2327.62	30	11
	$SPC\ full\ truck \rightarrow SPD$	**0.00**	93.72	0.17	3.09	1518.40	1521.50	30	9
	$SPC\ full\ truck\ customer\ based \rightarrow SPD$	46.56	92.17	10.78	6.93	1575.55	1582.48	30	10
	minimum costs	**498.48**	**813.24**	**2070.29**	-	-	-	-	-
4	$SPD\ infinite \rightarrow SPC$	169.70	**0.00**	16.60	3.00	312.69	315.97	36	8
	$SPD\ finite\ balanced \rightarrow SPC$	199.26	2.33	26.31	4.44	192.39	197.71	38	9
	$SPD\ finite\ supplier\ based \rightarrow SPC$	98.44	29.40	5.66	2.52	471.45	474.94	37	8
	$SPC\ not\ full\ truck \rightarrow SPD$	**0.00**	153.19	20.14	3.19	1887.38	1890.56	35	8
	$SPC\ full\ truck \rightarrow SPD$	**0.00**	96.58	**0.00**	3.22	787.23	790.45	35	8
	$SPC\ full\ truck\ customer\ based \rightarrow SPD$	36.04	87.90	7.73	6.30	942.31	948.61	35	9
	minimum costs	**644.11**	**763.10**	**2144.25**	-	-	-	-	-
5	$SPD\ infinite \rightarrow SPC$	133.83	**0.00**	19.27	22.02	301.71	324.09	49	12
	$SPD\ finite\ balanced \rightarrow SPC$	126.99	6.84	18.50	6.87	380.67	388.94	50	12
	$SPD\ finite\ supplier\ based \rightarrow SPC$	61.94	37.71	1.70	6.93	596.74	604.61	49	13
	$SPC\ not\ full\ truck \rightarrow SPD$	**0.00**	189.16	19.75	3.86	2227.06	2230.91	46	12
	$SPC\ full\ truck \rightarrow SPD$	**0.00**	118.38	**0.00**	3.80	1272.64	1276.44	46	11
	$SPC\ full\ truck\ customer\ based \rightarrow SPD$	20.44	121.19	8.77	6.86	1327.52	1334.38	46	12
	minimum costs	**1135.42**	**810.68**	**2905.79**	-	-	-	-	-
6	$SPD\ infinite \rightarrow SPC$	168.15	**0.00**	28.23	4.58	262.93	267.80	41	9
	$SPD\ finite\ balanced \rightarrow SPC$	160.32	6.14	27.50	7.85	142.78	151.62	42	9
	$SPD\ finite\ supplier\ based \rightarrow SPC$	69.19	34.51	4.66	2.89	345.44	348.85	42	7
	$SPC\ not\ full\ truck \rightarrow SPD$	**0.00**	175.68	27.12	3.06	1520.67	1523.72	38	8
	$SPC\ full\ truck \rightarrow SPD$	**0.00**	94.15	**0.00**	3.11	931.38	934.48	38	8
	$SPC\ full\ truck\ customer\ based \rightarrow SPD$	24.68	103.07	11.71	5.31	738.85	744.16	38	9
	minimum costs	**790.98**	**742.95**	**2233.42**	-	-	-	-	-
7	$SPD\ infinite \rightarrow SPC$	164.61	**0.00**	21.63	1.70	447.42	449.44	40	9
	$SPD\ finite\ balanced \rightarrow SPC$	147.56	1.22	16.34	4.61	287.01	292.67	42	9
	$SPD\ finite\ supplier\ based \rightarrow SPC$	73.91	26.23	**0.00**	3.56	861.23	865.45	41	10
	$SPC\ not\ full\ truck \rightarrow SPD$	**0.00**	167.72	22.18	4.27	1938.72	1942.99	38	11
	$SPC\ full\ truck \rightarrow SPD$	**0.00**	112.05	3.73	4.37	1570.88	1575.25	38	9
	$SPC\ full\ truck\ customer\ based \rightarrow SPD$	32.95	89.43	7.25	9.25	1793.77	1803.02	38	8
	minimum costs	**790.98**	**784.04**	**2365.34**	-	-	-	-	-
8	$SPD\ infinite \rightarrow SPC$	101.10	**0.00**	11.30	16.75	552.88	569.94	37	11
	$SPD\ finite\ balanced \rightarrow SPC$	111.95	5.87	17.49	2.34	371.81	375.29	36	9
	$SPD\ finite\ supplier\ based \rightarrow SPC$	68.41	15.25	5.24	0.89	779.23	780.76	36	9
	$SPC\ not\ full\ truck \rightarrow SPD$	**0.00**	172.27	40.32	3.37	3942.81	3946.19	34	10
	$SPC\ full\ truck \rightarrow SPD$	**0.00**	66.68	**0.00**	3.34	1570.57	1573.91	34	9
	$SPC\ full\ truck\ customer\ based \rightarrow SPD$	19.16	77.19	10.98	5.91	1494.92	1500.83	34	11
	minimum costs	**727.33**	**763.74**	**2000.32**	-	-	-	-	-

Table 4.11: Detailed results for supermarkets instances.

instance id	Strategy	$\Delta SPCcost$	$\Delta SPDcost$	$\Delta Cost$	$SPCt(s)$	$SPDt(s)$	$t(s)$	nbT	nbV
1	$SPD\ infinite \rightarrow SPC$	214.00	**0.00**	3.48	611.18	1820.26	2432.16	77	36
	$SPD\ finite\ balanced \rightarrow SPC$	245.69	20.25	16.42	606.65	2482.62	3091.35	76	36
	$SPD\ finite\ supplier\ based \rightarrow SPC$	108.98	93.16	**0.00**	51.29	2713.73	2767.41	76	35
	$SPC\ not\ full\ truck \rightarrow SPD$	**0.00**	265.70	15.04	6.97	8391.10	8398.06	77	36
	$SPC\ full\ truck \rightarrow SPD$	**0.00**	234.38	7.34	6.97	6459.01	6465.98	77	37
	$SPC\ full\ truck\ customer\ based \rightarrow SPD$	10.69	224.58	7.62	15.73	10286.01	10301.74	76	36
	minimum costs	**1620.35**	**1585.66**	**6449.01**	-	-	-	-	-
2	$SPD\ infinite \rightarrow SPC$	228.00	**0.00**	4.24	605.75	1277.51	1883.88	75	31
	$SPD\ finite\ balanced \rightarrow SPC$	268.59	24.63	20.04	605.28	2116.40	2723.28	73	32
	$SPD\ finite\ supplier\ based \rightarrow SPC$	125.48	88.55	**0.00**	40.10	2066.67	2108.86	73	31
	$SPC\ not\ full\ truck \rightarrow SPD$	**0.00**	250.33	7.44	4.89	5610.75	5615.64	71	33
	$SPC\ full\ truck \rightarrow SPD$	**0.00**	222.78	0.92	5.05	4205.70	4210.75	71	30
	$SPC\ full\ truck\ customer\ based \rightarrow SPD$	11.45	226.89	4.71	10.20	5946.75	5956.95	71	34
	minimum costs	**1519.26**	**1462.82**	**6183.81**	-	-	-	-	-
3	$SPD\ infinite \rightarrow SPC$	205.51	**0.00**	**0.00**	610.75	1931.09	2542.59	73	34
	$SPD\ finite\ balanced \rightarrow SPC$	252.42	23.57	17.42	612.11	2658.14	3272.19	73	35
	$SPD\ finite\ supplier\ based \rightarrow SPC$	116.08	90.33	0.76	33.61	2905.92	2940.85	72	34
	$SPC\ not\ full\ truck \rightarrow SPD$	**0.00**	242.07	10.40	6.78	8684.81	8691.59	71	36
	$SPC\ full\ truck \rightarrow SPD$	**0.00**	224.71	6.04	7.28	6999.63	7006.91	71	37
	$SPC\ full\ truck\ customer\ based \rightarrow SPD$	16.20	208.08	5.84	15.25	9249.36	9264.61	71	36
	minimum costs	**1512.12**	**1548.85**	**6168.55**	-	-	-	-	-
4	$SPD\ infinite \rightarrow SPC$	96.70	**0.00**	3.88	608.26	2972.43	3581.45	86	37
	$SPD\ finite\ balanced \rightarrow SPC$	116.11	22.50	18.04	603.28	4000.20	4605.78	85	38
	$SPD\ finite\ supplier\ based \rightarrow SPC$	48.34	45.47	**0.00**	155.32	3876.63	4033.38	84	37
	$SPC\ not\ full\ truck \rightarrow SPD$	**0.00**	231.68	39.65	8.27	14438.65	14446.91	84	38
	$SPC\ full\ truck \rightarrow SPD$	**0.00**	160.36	17.59	8.42	10476.51	10484.93	84	37
	$SPC\ full\ truck\ customer\ based \rightarrow SPD$	11.55	163.04	22.71	22.65	12376.38	12399.03	84	40
	minimum costs	**1985.06**	**1654.69**	**5351.62**	-	-	-	-	-
5	$SPD\ infinite \rightarrow SPC$	72.48	**0.00**	4.77	605.59	1257.91	1864.09	77	34
	$SPD\ finite\ balanced \rightarrow SPC$	110.85	21.16	27.70	604.33	1563.74	2169.92	77	34
	$SPD\ finite\ supplier\ based \rightarrow SPC$	47.10	16.78	**0.00**	604.88	1559.48	2165.50	77	33
	$SPC\ not\ full\ truck \rightarrow SPD$	**0.00**	178.09	35.17	7.99	5495.13	5503.12	74	34
	$SPC\ full\ truck \rightarrow SPD$	**0.00**	102.42	9.58	7.95	4345.60	4353.55	74	36
	$SPC\ full\ truck\ customer\ based \rightarrow SPD$	4.93	102.26	11.56	12.16	4415.38	4427.54	74	38
	minimum costs	**1816.33**	**1493.05**	**4415.45**	-	-	-	-	-
6	$SPD\ infinite \rightarrow SPC$	154.81	**0.00**	1.63	606.33	1554.77	2161.85	98	48
	$SPD\ finite\ balanced \rightarrow SPC$	193.37	22.16	18.83	606.05	2318.84	2927.02	98	44
	$SPD\ finite\ supplier\ based \rightarrow SPC$	85.71	72.96	**0.00**	175.93	2385.50	2562.88	95	43
	$SPC\ not\ full\ truck \rightarrow SPD$	**0.00**	238.19	17.34	10.68	8621.81	8632.49	96	45
	$SPC\ full\ truck \rightarrow SPD$	**0.00**	193.98	5.89	10.54	6007.16	6017.70	96	47
	$SPC\ full\ truck\ customer\ based \rightarrow SPD$	9.58	187.48	7.05	18.90	7724.43	7743.33	96	46
	minimum costs	**2091.43**	**1823.58**	**7037.98**	-	-	-	-	-
7	$SPD\ infinite \rightarrow SPC$	162.14	**0.00**	3.96	605.93	2970.73	3577.50	109	51
	$SPD\ finite\ balanced \rightarrow SPC$	199.68	22.26	20.81	605.41	6528.79	7136.93	109	50
	$SPD\ finite\ supplier\ based \rightarrow SPC$	80.68	83.98	**0.00**	610.58	6411.95	7024.80	108	49
	$SPC\ not\ full\ truck \rightarrow SPD$	**0.00**	258.95	18.69	14.14	16299.01	16313.14	107	49
	$SPC\ full\ truck \rightarrow SPD$	**0.00**	214.87	7.83	14.03	20570.22	20584.25	107	53
	$SPC\ full\ truck\ customer\ based \rightarrow SPD$	7.90	265.78	22.76	66.09	16548.57	16614.66	106	53
	minimum costs	**2374.80**	**1933.26**	**7847.62**	-	-	-	-	-
8	$SPD\ infinite \rightarrow SPC$	170.10	**0.00**	**0.00**	605.34	2497.02	3103.21	105	48
	$SPD\ finite\ balanced \rightarrow SPC$	206.99	25.41	16.43	607.10	5173.62	5783.09	107	46
	$SPD\ finite\ supplier\ based \rightarrow SPC$	104.89	81.77	0.84	604.99	4574.01	5181.14	105	46
	$SPC\ not\ full\ truck \rightarrow SPD$	**0.00**	258.53	12.85	17.90	10955.42	10973.32	102	48
	$SPC\ full\ truck \rightarrow SPD$	**0.00**	220.44	3.87	17.56	8705.85	8723.41	102	52
	$SPC\ full\ truck\ customer\ based \rightarrow SPD$	8.34	194.36	0.08	20.73	12724.18	12744.91	101	51
	minimum costs	**2250.39**	**1876.13**	**7954.41**	-	-	-	-	-
9	$SPD\ infinite \rightarrow SPC$	197.45	**0.00**	**0.00**	605.49	2294.35	2900.67	93	44
	$SPD\ finite\ balanced \rightarrow SPC$	235.90	27.01	16.25	606.94	3506.61	4115.86	91	45
	$SPD\ finite\ supplier\ based \rightarrow SPC$	126.24	83.97	1.48	605.33	3923.75	4534.08	93	41
	$SPC\ not\ full\ truck \rightarrow SPD$	**0.00**	265.42	11.78	9.30	12015.01	12024.31	87	42
	$SPC\ full\ truck \rightarrow SPD$	**0.00**	239.52	5.68	9.30	11836.48	11845.78	87	45
	$SPC\ full\ truck\ customer\ based \rightarrow SPD$	11.59	218.23	3.65	17.14	9850.93	9868.07	87	45
	minimum costs	**1911.03**	**1751.77**	**7436.06**	-	-	-	-	-
10	$SPD\ infinite \rightarrow SPC$	214.00	**0.00**	3.48	611.83	1812.64	2425.15	77	36
	$SPD\ finite\ balanced \rightarrow SPC$	246.19	20.25	16.55	606.66	2481.68	3090.51	76	36
	$SPD\ finite\ supplier\ based \rightarrow SPC$	108.98	93.16	**0.00**	55.61	2827.97	2887.82	76	35
	$SPC\ not\ full\ truck \rightarrow SPD$	**0.00**	245.62	10.11	6.88	8623.59	8630.47	77	36
	$SPC\ full\ truck \rightarrow SPD$	**0.00**	234.38	7.34	6.94	6372.42	6379.35	77	37
	$SPC\ full\ truck\ customer\ based \rightarrow SPD$	10.69	239.23	11.22	15.91	8250.89	8266.80	76	37
	minimum costs	**1620.35**	**1585.66**	**6449.01**	-	-	-	-	-

can be noticed that some sequential algorithms lead to a very high cost, namely $SPD\ finite\ balanced \rightarrow SPC$ and $SPC\ not\ full\ truck \rightarrow SPD$.

We note that the algorithm $SPD\ finite\ supplier\ based \rightarrow SPC$ again provides very good results on these additional sets of instances. However, depending on the instance setting, some approaches based on the SPC solution first can be competitive.

From a computational point of view, solving the SPD first takes much less time than solving the SPC first. Moreover, most of the time is spent on solving the SPD. When solving the SPC first, the available quantities at the distribution centers are very limited, which may generate a significant infeasibility when applying destroy and repair moves in the ALNS algorithm. This makes it more difficult to obtain good solutions with the local search.

When customers are school canteens, few trucks are used. Their number is less than the number of suppliers. This means that some suppliers do not provide any commodity since they are far from each distribution center. When customers are supermarkets, the total supply is not very large compared to the total demand. Hence, more trucks are used for the delivery and several suppliers perform more than one trip to a distribution center.

4.7 Conclusion and future research

In this paper, we presented a new and complex problem which occurs in local agri-food supply chains, the MC2DP. The problem concerns the collection of commodities from suppliers to distribution centers and the delivery from distribution centers to final customers. The objective is to jointly optimize the transportation plan for the collection and delivery operations. In order to tackle this complex problem, we proposed two sequential approaches based on the decomposition of the problem in two subproblems: collection and delivery. For each sequential approach, three strategies are considered in order to take into account, in the first subproblem solved, the impact of its solution on the second subproblem solved.

The proposed algorithms have been compared on several sets of instances derived from instances from the literature. The best algorithm usually does not provide the lowest cost neither for the collection nor for the delivery subproblem. The algorithms are also tested on a case study of a short and local fresh food supply chain.

Large size instances are considered with two types of customers, school canteens and supermarkets. We compared the different algorithms and provides some managerial insights about cost balancing between collection and delivery. The strategy leading to the lowest transportation cost provides very unbalanced collection and delivery costs, while the second best strategy is slightly more costly but has a much better balancing of costs between collection and delivery.

The MC2DP is a complex problem to solve. The proposed sequential approaches have the advantage of being easy to design and to be understood by non-expert decision makers. However, a sequential solution approach may provide largely sub-optimal solutions in some cases. The main future research direction is to develop a integrated solution approach solving the MC2DP as a whole instead of decomposing it in subproblems.

The project presented in this paper is partially supported by the CSC (China Scholarship Council). The authors thank *Conseil Général de l'Isère* and *Chambre d'Agriculture de l'Isère* for providing the case study instances.

Chapter 5

An integrated optimization approach for a Multi-Commodity two-echelon Distribution Problem

Contents

5.1 Introduction

The Multi-Commodity two-echelon Distribution Problem (MC2DP) is a complex distribution problem in a two-echelon supply chain. The problem has been introduced in Chapter 4 where a two-phase approach for the MC2DP is presented. In this chapter, we propose an integrated heuristic approach to solve this problem.

In the MC2DP, three sets of stakeholders are involved: suppliers, distribution centers, and customers. Multiple commodities are collected from the suppliers and delivered to the customers through distribution centers for consolidation purposes. Each supplier has a given available quantity for each commodity (possibly 0) and each customer has a demand for each commodity (possibly 0). We consider a single decision maker who manages all the distribution centers and organizes the collection and delivery operations. The commodities are collected from suppliers and delivered to distribution centers through direct trips and distributed from the distribution centers to customers with a fleet of vehicles performing routes. Direct deliveries from suppliers to customers are not allowed. Commodities are compatible, that is any vehicle can transport any set of commodities as long as its capacity is not exceeded. Multiple visits to a customer are allowed to reduce transportation costs. However, for the sake of customer convenience, a single commodity has to be delivered at once. The MC2DP consists in finding a collection and delivery plan that minimizes the total transportation cost, satisfying customer demands, not exceeding the available quantities at the suppliers and the vehicle capacities.

In Chapter 4, we proposed a decomposition of the MC2DP in two subproblems, associated with the collection and delivery phases. In order to solve the problem, we proposed to solve the subproblems sequentially. The collection subproblem is modeled as a Mixed Integer linear Program. The delivery problem is the multi-depot case of the Commodity constrained Split Delivery Vehicle Routing Problem (C-SDVRP). Hence the Adaptive Large Neighborhood Search (ALNS) for the C-SDVRP presented in Chapter 3 was extended to the multi-depot case for the delivery subproblem.

The sequential approaches proposed in Chapter 4 have the advantage of being easy to design and to be understood by non-expert decision makers. However, a sequential solution approach may provide largely suboptimal solutions in some cases. The goal of this chapter is to use a more sophisticated integrated solution method to improve the results obtained by the sequential approaches.

In this chapter, the ALNS proposed in Chapter 4 to solve the delivery subproblem is improved by considering the modification of the solution of the collection subproblem. More precisely, we add new operators able to modify both the collection and the delivery, by moving some quantities from one distribution center to another. Some of these operators are based on mathematical programming in order to change the current solution of the collection subproblem. This integrated approach is evaluated and compared to the best solutions obtained with the sequential approaches.

This chapter is organized as follows. In Section 5.2, we give a description of the MC2DP. The proposed integrated approach to solve the MC2DP is described in Section 5.3. Section 5.4 reports the computational results. Section 5.5 concludes the chapter and provides some perspectives in order to improve the proposed solving approach.

5.2 Problem definition

In the following of this section, we define the MC2DP. A more detailed description can be found in Chapter 4 (in Section 4.1 and 4.2). As described in Chapter 4, the MC2DP can be decomposed into two subproblems, one that considers the collection of commodities, that is the transportation of the commodities from the suppliers to the distribution centers, the other that considers the delivery of commodities from the distribution centers to the customers. In Section 5.2.1, we first describe the whole problem MC2DP, and then we define the two subproblems: the SPC (SubProblem Collection) and SPD (SubProblem Delivery) in Section 5.2.2.

5.2.1 The Multi-Commodity two-echelon Distribution Problem

The Multi-Commodity two-echelon Distribution Problem (MC2DP) is defined on a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, in which $\mathcal{V}$ is the set of vertices and $\mathcal{A}$ is the set of arcs. More precisely, $\mathcal{V}$ is defined as $\mathcal{V}_S \bigcup \mathcal{V}_D \bigcup \mathcal{V}_C$ where $\mathcal{V}_S = \{1, \dots N_S\}$ represents the set of suppliers, $\mathcal{V}_D = \{N_S + 1, \dots, N_S + N_D\}$ is the set of distribution centers and $\mathcal{V}_C = \{N_S + N_D + 1, \dots, N_S + N_D + N_C\}$ represents the set of customers. We only consider direct trips from suppliers to distribution centers. Direct deliveries from suppliers to customers are not allowed. Moreover, transfers of commodities

between distribution centers are not considered. Thus, $\mathcal{A} = \{(i,j),(j,i)|i \in \mathcal{V}_S, j \in \mathcal{V}_D\} \cup \{(i,j),(j,i)|i \in \mathcal{V}_D, j \in \mathcal{V}_C\} \cup \{(i,j)|i,j \in \mathcal{V}_C\}$ is the arc set.

Suppliers provide a set $\mathcal{M}$ of commodities, which are transported to the distribution centers using an unlimited fleet of homogeneous vehicles of capacity Q_S. Each supplier $s \in \mathcal{V}_S$ is associated with a maximum available quantity O_{sm} (possibly equal to zero) of commodity $m \in \mathcal{M}$. Each distribution center has an unlimited fleet of homogeneous vehicles of capacity Q_D that are used to deliver the commodities to the customers. Customer $i \in \mathcal{V}_C$ requires a quantity D_{im} (possibly equal to zero) of each commodity $m \in \mathcal{M}$. Commodities are compatible, i.e., they can be transported in the same vehicle. The demand of a customer can be split, that is, the customer can be served by several vehicles. However, for the sake of customer convenience, the split policy is constrained: each commodity has to be delivered to each customer by one vehicle only. A cost c_{ij} is associated with each arc $(i,j) \in \mathcal{A}$ and represents the non-negative cost of traversing arc (i,j).

5.2.2 Subproblems

In the following, we will call *collection* the transportation of commodities from suppliers to distribution centers and *delivery* the distribution of commodities from distribution centers to customers.

The collection and delivery phases of the MC2DP are connected through the quantities of commodities brought to and delivered from the distribution centers. We denote by U_{dm} the quantity of commodity $m \in \mathcal{M}$ at distribution center d. For the sake of clarity, when necessary, we will denote by U^D_{dm} the quantity of commodity m that is delivered from the distribution center d to customers, and by U^C_{dm} the quantity of commodity m that is collected at the suppliers and brought to the distribution center d.

We now provide a formal definition of the SPC and the SPD. Note that each subproblem is concerned with the optimization of the related operations, i.e., the SPC minimizes the cost of collection operations while the SPD minimizes the cost of delivery operations.

5.2.2.1 The Collection Subproblem (SPC)

The SPC is defined on a graph $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{A}_1)$, where $\mathcal{V}_1 = \mathcal{V}_S \bigcup \mathcal{V}_D$ and $\mathcal{A}_1 = \{(i,j),(j,i)|i \in \mathcal{V}_S, j \in \mathcal{V}_D\}$. In the following, we will use the word *truck* to indicate

a vehicle used in the SPC. The SPC consists in determining a set of direct trips for trucks between suppliers and distribution centers with the associated quantities for each commodity. The objective is to minimize the transportation cost, defined as the total cost for the direct trips, that is independent of the quantity transported on each truck. For each commodity, the quantity transported to distribution centers has to be sufficient to satisfy the customer demands in the SPD. Moreover, the solution of the SPC must satisfy the following constraints:

(1) the total quantity of commodities transported by each truck does not exceed the truck capacity Q_S;

(2) the quantity of each commodity m that is transported from a supplier s to distribution centers must be at most equal to the available quantity O_{sm};

(3) the quantity of each commodity m transported to each distribution center d is greater than or equal to the required quantity U^D_{dm}.

5.2.2.2 The Delivery Subproblem (SPD)

The SPD is defined on a graph $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{A}_2)$ where $\mathcal{V}_2 = \mathcal{V}_D \bigcup \mathcal{V}_C$ and $\mathcal{A}_2 = \{(i,j),(j,i) | i \in \mathcal{V}_D, j \in \mathcal{V}_C\} \cup \{(i,j) | i,j \in \mathcal{V}_C\}$. In the following, we will use the word *vehicle* to indicate a vehicle used in the SPD.

The SPD consists in determining a set of vehicle routes such that all the customer demands are satisfied in assigning commodities to vehicles. The solution must satisfy the following constraints:

(1) the total quantity of commodities delivered by each vehicle does not exceed the vehicle capacity Q_D;

(2) each commodity requested by each customer is delivered by a single vehicle;

(3) the demands of all customers are satisfied;

(4) the quantity of each commodity m distributed from each distribution center d does not exceed the available quantity U^C_{dm};

(5) each vehicle starts and ends its route at the same distribution center.

5.3 An integrated optimization algorithm

In Chapter 4, a solution method for the SPD has been developed, based on the ALNS proposed for the one depot case in Chapter 3.

In this section, we propose to improve this ALNS in order to modify also the current solution of the SPC. This is done by proposing dedicated optimization operators that modify both the delivery and the collection parts of the current solution. This results in an integrated optimization algorithm to tackle the whole MC2DP.

As in previous chapters, we use the concept of *customer-commodity* that denotes the demand of a customer for a single commodity. A customer may be treated as a union of *customer-commodities*.

5.3.1 General framework

With the aim of developing an integrated solution approach, the main drawback of the ALNS developed to solve the SPD is that it never modifies of the current SPC solution, and is not able to update the available quantities U^C_{dm} at the distribution centers. Moreover, some moves defined for the SPD may not be feasible because (1) they concern customers or *customer-commodities* in routes assigned to different distribution centers, and (2) the available quantities U^C_{dm} at these distribution centers do not enable to implement the considered move (i.e., the quantity of commodity m distributed from one distribution center d would exceed U^C_{dm}). A modification of the SPC would update the available quantities at the distribution centers U^C_{dm}, and then a move concerning the SPD could then become feasible.

Moreover, note that the SPC has a special cost structure: the collection cost depends on the number of trucks from suppliers to distribution centers, and does not depend on the transported quantities. This means that a modification of the available quantities at the distribution centers U^C_{dm} may even be performed by simply modifying the transported quantities in the SPC, without modifying the collection cost.

In the following, we present the general framework of the proposed ALNS to solve the whole MC2DP. The framework is given in Algorithm 4. The ALNS relies on a set of removal and insertion heuristics, which iteratively destroy and repair solutions. The removal and insertion heuristics are selected using a roulette wheel mechanism (see Chapter 3 of this document). The probability of selecting a heuristic

is dynamically influenced by its performance in past iterations (Pisinger & Ropke, 2010). Moreover, an intensification procedure permits to improve the solutions obtained after applying removal and insertion heuristics.

Note that in the following *local operators* will denote the operators that only modify the SPD, and *large operators* will denote the operators that modify the whole MC2DP, i.e. the SPC and the SPD.

The main changes from the ALNS developed to solve the SPD are the following:

- we consider the total cost for the MC2DP to compare solutions;
- the initial solution is a solution of the MC2DP;
- we do not consider infeasible solutions that would be produced by the application of an insertion heuristic due to the limited availabilities of commodities at the distribution centers. Instead, we consider the use of large operators that modify the collection part of the current solution;
- the intensification phase considers both local and large operators.

Algorithm 4 Framework of the ALNS to solve the MC2DP.

```
1: Generate an initial solution for the MC2DP (see Section 5.3.2)
2: repeat
3:     Roulette wheel to select destroy and removal heuristics
4:     Destroy the SPD
5:     Repair the SPD, with possible use of large operators (see Section 5.3.4)
6:     Intensification with local and large operators (see Section 5.3.5)
7:     if a new best know solution is obtained then
8:         Intensification with MPO for the SPD (see Section 4.4.2.2 of Chapter 4)
9:     end if
10:    Acceptance criterion to update the current solution
11: until stopping criterion is met
12: return the best known solution
```

In the following, we detail the main components of the algorithm: the generation of an initial solution for the MC2DP, the large operators that modify both the SPC and the SPD, and their use in destroy and repair heuristics, as well as in the intensification procedure.

5.3.2 Initial solution

In order to provide an initial solution for the MC2DP, we apply the sequential approaches presented in Chapter 4 (see Section 4.4.3). They are based on the decomposition of the problem in the two subproblems SPC and SPD. For each sequential approach, three strategies have been proposed in order to take into account, in the first subproblem solved, the impact of its solution on the second subproblem. We recall here the six strategies:

- $SPD\ infinite \rightarrow SPC$: the SPD is solved first, then the SPC is solved;
- $SPD\ finite\ balanced \rightarrow SPC$: the SPD is solved first, balancing the available quantities at the distribution centers , then the SPC is solved;
- $SPD\ finite\ supplier\ based \rightarrow SPC$: the SPD is solved first, taking into account the location of the suppliers to determine available quantities at the distribution centers, then the SPC is solved;
- $SPC\ not\ full\ truck \rightarrow SPD$: the SPC is solved first, then the SPD is solved;
- $SPC\ full\ truck \rightarrow SPD$: the SPC is solved first, and the capacity of the trucks is fully used, then the SPD is solved;
- $SPC\ full\ truck\ customer\ based \rightarrow SPD$: the SPC is solved first, taking into account the location of the customers, then the SPD is solved.

In order to start with a good initial solution, all these six strategies are run, so that we obtain six solutions of the MC2DP. The solution with the lowest total cost is kept as the initial solution for the integrated algorithm. Based on the results obtained in Chapter 4, it seems interesting to apply the six strategies and keep the best solutions since, depending on the instance configuration, the best strategy is not always the same.

Note that, each sequential strategy requires the execution of one ALNS call to solve the SPD. The stopping criterion of the ALNS algorithm can be set such that the computation time to get a feasible solution remains low.

5.3.3 Large operators

Large operators Φ modify the current solution of the MC2DP on both the SPD and the SPC. They are combination of one operator Φ_{SPD} for the SPD and one operator Φ_{SPC} for the SPC. Φ is applied when the application of Φ_{SPD} would result in an infeasible solution due to the limited available quantities U_{dm} at the distribution centers. The role of Φ_{SPC} is then to modify the current solution of the SPC (and thus the available quantities U_{dm} at the distribution centers) with the global objective to make the application of Φ_{SPD} feasible in the hope that the application of the large operator Φ improves the current solution. A particular application of an operator is also called *move*.

In the following, we assume that a nonfeasible application of an operator for the SPD would require quantities U^D_{dm} to be available at the distribution centers. We first provide a Mixed Integer linear Programming (MIP) formulation for the SPC in order to satisfy the required quantities U^D_{dm}. Then, we propose three operators for the SPC.

The cost structure of the SPC is such that a cost is incurred for each truck between a supplier and a distribution center, but the cost does not depend on the quantities transported on these trucks. Let us define as *structure of the network* the decisions related to the number of trucks circulating between the suppliers and the distribution centers, and *flow* the decisions related to the flow of commodities circulating in the network. Hence, changing the *flow* without modifying the *structure of the network* can be done at zero cost. The three operators are the following:

- `heuristic flow`: the flow of a single commodity m is modified, without modifying the structure of the network;
- `multi-commodity flow`: the flow of all the commodities can be modified, without modifying the structure of the network;
- `network`: the structure of the network and the flow of the commodities are modified.

5.3.3.1 A MIP formulation for the SPC

In this section we propose a MIP formulation for the SPC such that the optimal solution minimizes the transportation cost to bring the quantities U^D_{dm} required at

the distribution centers. Moreover, it ensures the respect of the available quantities at the suppliers and the truck capacities.

We recall that for the large operators, these quantities U^D_{dm} provide from an infeasible local move for the SPD, and this local move would become feasible if quantities U^D_{dm} can be available at the distribution centers. The formulation is based on the MIP formulation for the SPC presented in Chapter 4 (see Section 4.4.1). We define the following decision variables:

- $x_{sd} \in \mathbb{N}$: the number of trucks sent from supplier $s \in \mathcal{V}_S$ to distribution center $d \in \mathcal{V}_D$;
- $q^m_{sd} \in \mathbb{R}_+$: the quantity of commodity $m \in \mathcal{M}$ that supplier $s \in \mathcal{V}_S$ provides to distribution center $d \in \mathcal{V}_D$.

In addition to the notation introduced in Section 5.2, we define as $\bar{c}_{sd} = c_{sd} + c_{ds}$ the direct trip cost between supplier $s \in \mathcal{V}_S$ and distribution center $d \in \mathcal{V}_D$.

The formulation is the following:

$$\min \sum_{s \in \mathcal{V}_S} \sum_{d \in \mathcal{V}_D} \bar{c}_{sd} x_{sd} \tag{5.1}$$

$$\text{s.t.} \sum_{s \in \mathcal{V}_S} q^m_{sd} \geq U^D_{dm}, \qquad \forall\, d \in \mathcal{V}_D, m \in \mathcal{M}, \tag{5.2}$$

$$\sum_{m \in \mathcal{M}} q^m_{sd} \leq Q_S x_{sd}, \qquad \forall\, s \in \mathcal{V}_S, d \in \mathcal{V}_D, \tag{5.3}$$

$$\sum_{d \in \mathcal{V}_D} q^m_{sd} \leq O_{sm}, \qquad \forall\, s \in \mathcal{V}_S, m \in \mathcal{M}, \tag{5.4}$$

$$x_{sd} \in \mathbb{N}, \qquad \forall\, s \in \mathcal{V}_S, d \in \mathcal{V}_D, \tag{5.5}$$

$$q^m_{sd} \geq 0, \qquad \forall\, s \in \mathcal{V}_S, d \in \mathcal{V}_D, m \in \mathcal{M}. \tag{5.6}$$

The objective function (5.1) minimizes the transportation cost, which is the cost of the direct trips of the trucks used to supply the distribution centers. Constraints (5.2) ensure that the quantity transported to each distribution center are greater than or equal to the required quantities U^D_{dm}. Constraints (5.3) impose that the capacity of the truck is not exceeded. Constraints (5.4) impose to not exceed the quantity of each commodities available at the suppliers. Constraints (5.5)–(5.6) define the variables.

It should be noticed that the variables x_{sd} represent the decisions about the structure of the network, while the variables q_{sd}^m are the decisions about the flow circulating in the network.

5.3.3.2 The operators for the SPC

In the following, we describe the operators for the SPC. We use U_{dm} to indicate the values of the available quantities at the distribution centers in the current solution of the MC2DP, while U_{dm}^D indicate the values of the required quantities at the distribution centers such that an application of an operator Φ_{SPD} would become feasible.

The network operator

In order to provide a solution of the SPC that satisfies the required quantities U_{dm}^D at the distribution centers, the network operator aims at optimizing the whole SPC, by modifying the structure of the network, and in consequence the flow circulating in the network. To apply this operator, we propose to solve the MIP formulation (5.1)-(5.6).

Hence, the network operator is a very large operator that reoptimizes the SPC in order to satisfy the required quantities U_{dm}^D at the distribution centers. Note that the network operator always finds a feasible solution, since all the suppliers are able to serve all the distribution centers (with the assumption that suppliers provide sufficient quantities to satisfy the demand of the customers. This assumption is not restrictive since otherwise the problem is infeasible). The application of the network operator adapts the structure of the network to the required quantities U_{dm}^D. Since these quantities differ from U_{dm}, the application of network operator may change the cost of the solution of the SPC.

The multi-commodity flow operator

The multi-commodity flow operator consists in solving the MIP formulation (5.1)-(5.6) where the decisions variables x_{sd} are fixed in order to represent the current SPC network. Thus, only the values of variables q_{sd}^m need to be determined. Since q_{sd}^m are continuous, the model becomes a Linear Program (LP) that is usually fast to solve. Note that the application of the multi-commodity flow operator does not change the

cost of the current solution of the SPC, and aims at finding a multi-commodity flow that satisfies the required quantities U^D_{dm}.

An example of an instance is given in Figure 5.1. Triangles represent the locations of suppliers, and squares indicate the locations of distribution centers. Commodities are in the dotted ellipses. To increase readability, a different colored ellipse represents each commodity, and the same colored line indicates the flow of this commodity. For example, blue ellipses and lines represent commodity m_1 and its flow. The graph is bipartite with suppliers on one side and distribution centers on the other side. For each supplier, there is an incoming arc for each commodity with a capacity O_{sm}. For each distribution center, there is an outgoing arc for each commodity with a capacity equal to the required quantities U^D_{dm}. Between suppliers and distribution centers, there are arcs with multiple commodities and a capacity equal to $Q_S x_{sd}$. If a maximum multi-commodity flow of value $\sum_{m\in\mathcal{M};d\in\mathcal{V}_D} U^D_{dm}$ can be found then the SPC has a feasible solution with the same cost as the current one and such that it brings the required quantities at the distribution centers.

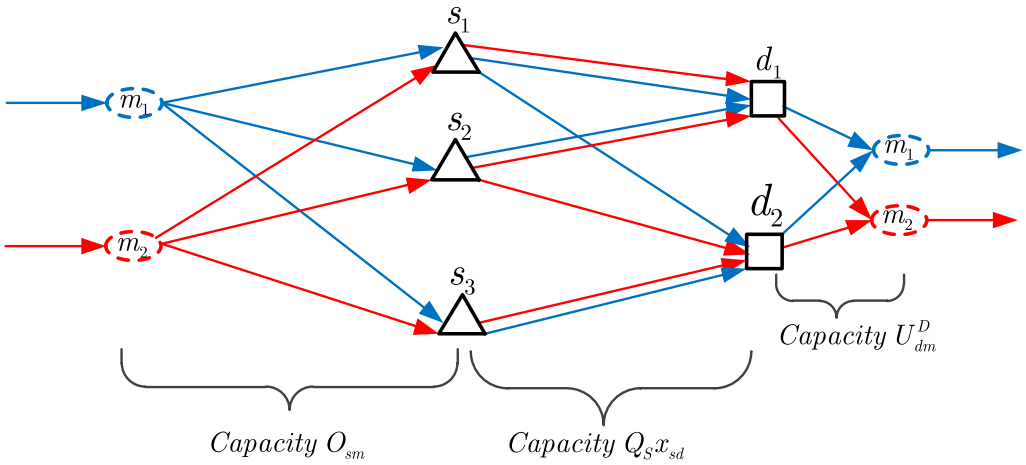


Fig. 5.1. An example of an instance for maximum multi-commodity flow.

5.3.3.3 The heuristic flow operator

The heuristic flow operators aim at heuristically find a solution for the SPC, by modifying only the flow decisions. The idea of the heuristic flow operator is to iteratively change the flow of the current solution for the SPC in order to get a solution with the requested U^D_{dm} quantities at the distribution centers. At each iteration, the changes concern the flow of a single commodity. This operator only

modifies the flow while maintaining the same network structure, so the cost of the SPC solution does not change.

More precisely, for each commodity m and each distribution center d, we first compute the value $q_{md} = U^D_{dm} - U_{dm}$. It represents the surplus of commodity m that has to be delivered to the distribution center d. If $q_{md} = 0$, nothing has to be changed. Otherwise, if $q_{md} > 0$ or $q_{md} < 0$, we have to respectively increase or decrease the flow of commodity m that arrives at distribution center d. Let us denote by $\mathcal{F}^+$ the set of flows to increase, and by $\mathcal{F}^-$ the set of flows to decrease. The flow of the current solution (values q^m_{sd}) is then updated by iteratively decreasing the value for each flow in $\mathcal{F}^-$, and then by iteratively increasing the value for each flow in $\mathcal{F}^+$. If increasing a flow in $\mathcal{F}^+$ has not been possible because of the constraints imposed by the capacity of the trucks or the available quantities at the suppliers, then the procedure stops, and the associated move is not feasible. Otherwise, if increasing all the flows in $\mathcal{F}^+$ is feasible, then we obtain a new feasible solution for the SPC with available quantities U^D_{dm} at the distribution centers. A detailed description is provided in Algorithm 5. Algorithms 6 and 7 present how the current solution is updated, when we remove or add some units of flow.

It is note worthy that the algorithms used to modify the flows are heuristics. However, they have the advantage to be very fast. The complexity of the whole algorithm is $\mathcal{O}(|\mathcal{M}| \cdot |\mathcal{V}_D| \cdot |\mathcal{V}_S|)$. Note that the operators that modify the SPD (see Section 5.3.5.1) only change the required quantities for two distribution centers, and some operators for the SPD change the required quantities for a single commodity. In the latter case, the complexity reduces to $\mathcal{O}(|\mathcal{V}_S|)$.

5.3.4 Destroy and repair heuristics

This procedure is aimed at diversifying the search. It is based on a set of removal and insertion operators which iteratively destroy and repair solutions of the SPD. However, when applying the insertion operator for the SPD, it may be possible that no feasible solution is found because of the quantities U^C_{dm} supplied to the distribution centers. In that case, we modify the SPC in order to get a feasible solution.

The removal and insertion operators are chosen using a roulette wheel mechanism. The probability of selecting an operator is dynamically modified by its performance in past iterations (Ropke & Pisinger, 2006). There are two removal

Algorithm 5 Algorithm of the heuristic flow operator

Input: values U_{dm} and U^D_{dm}.
$\mathcal{F}^+ \leftarrow \emptyset$, $\mathcal{F}^- \leftarrow \emptyset$
for each commodity $m \in \mathcal{M}$ **do**
  for each distribution center $d \in \mathcal{V}_D$ **do**
    $q_{dm} = U^D_{dm} - U_{dm}$
    if $q_{dm} > 0$ **then**
      $\mathcal{F}^+ \leftarrow \mathcal{F}^+ \cup \{(m, d, q_{dm})\}$
    end if
    if $q_{dm} < 0$ **then**
      $\mathcal{F}^- \leftarrow \mathcal{F}^- \cup \{(m, d, q_{dm})\}$
    end if
  end for
end for
for each $(m, d, q) \in \mathcal{F}^-$ **do**
  From the current solution, remove q units of commodity m to distribution center d
end for
for each $(m, d, q) \in \mathcal{F}^+$ **do**
  From the current solution, add q units of commodity m to distribution center d
  if no feasible solution has been found **then**
    return no feasible solution found
  end if
end for
return the current feasible solution with available quantities U^D_{dm} at the distribution centers

operators (Shaw removal and random removal of customers), and three insertion heuristics of *customer-commodity* based on greedy, regret-2 and regret-3 insertion paradigms. More details are provided in Chapter 3.

In the ALNS algorithm, violations of vehicle capacity are allowed and penalized in the cost function when *customer-commodities* are inserted into existing routes. However, we impose a maximum capacity violation on each route. Hence, it may happen that a *customer-commodity* cannot be inserted in any route of the current SPD solution. In this case, we select one distribution center from which serving this *customer-commodity* is feasible with respect to the availability of the involved commodity and we create one additional route. If several distribution centers can be considered, we select the one that minimizes the delivery cost.

Algorithm 6 Updating the current solution when removing q^{rem} units of flow.

Input: a quantity q^{rem} of commodity m_1 to remove from distribution center d_1

$q \leftarrow q^{rem}$ the remaining quantity to remove
for each supplier $s \in \mathcal{V}_S$ **do**
  if $q_{sd_1}^{m_1} > 0$ **then**
    $q_s \leftarrow \min\{q_{sd_1}^{m_1}; q\}$ the quantity that can be removed in a truck from supplier s
    $q \leftarrow q - q_s$
    $q_{sd_1}^{m_1} \leftarrow q_{sd_1}^{m_1} - q_s$ update of the solution
  end if
end for

Furthermore, since the problem considers multiple distribution centers, it may happen that a *customer-commodity* cannot be assigned to any distribution centers in the current solution. In order to face this situation, in Chapter 4 (see Section 4.4.2.2), we allow violations of the limited quantities U_{dm}^C of commodities available at each distribution center and penalize the violation in the objective function. In this work, we consider large operators that permit to modify the SPC (see Section 5.3.3). Thus, violations of the limited quantities U_{dm}^C of commodities available at each distribution center are not allowed anymore, and we use large operators to modify the SPC. When a *customer-commodity* cannot be inserted in the solution of the SPD due to the available quantities U_{dm}^C, we insert this *customer-commodity* in a route from the distribution center with the largest remaining quantity (calculated as the available quantity minus the quantity delivered to the customers from the distribution center). Then, in order to recover a feasible solution for the MC2DP, we apply the large operators detailed in Section 5.3.3 to get a feasible solution for the SPC. The required quantities U_{dm}^D at the distribution centers are updated based on the current solution of the SPD. Then, the large operators are called in the following order: heuristic flow operator, multi-commodity flow operator, and network operator. A first-improvement strategy is employed, namely whenever an operator finds a feasible solution it is implemented, and the other operators are not called. Note that the network operator always finds a feasible solution. However, it may increase or decrease the cost of the SPC.

If the solution of the SPC has been modified by implementing a large operator, then the *full truck* strategy proposed in the sequential approaches (see Algorithm 10 in Chapter 4) is applied on the updated solution of the SPC. The idea of the *full*

Algorithm 7 Updating the current solution when adding q^{add} units of flow.

Input: a quantity q^{add} of commodity m_1 to add to distribution center d_1
Output: true if the feasible solution have been found, false otherwise.

$q \leftarrow q^{add}$ the remaining quantity to add
for each supplier $s \in \mathcal{V}_S$ **do**
  if $q_{sd_1}^{m_1} > 0$ **then**
    $q_s \leftarrow \min\{q; Q_1 x_{sd_1} - \sum_{m \in \mathcal{M}} q_{sd_1}^m; O_{sm_1} - \sum_{d \in \mathcal{V}_D} q_{sd}^{m_1}\}$ the quantity that can be added in a truck from supplier s
    $q \leftarrow q - q_s$
    $q_{sd_1}^{m_1} \leftarrow q_{sd_1}^{m_1} + q_s$ update of the solution
    if $q = 0$ **then**
      return true
    end if
  end if
end for
if $q > 0$ **then return** false
end if

truck strategy is to use the remaining capacity of the trucks in the SPC to bring more products than required at the distribution centers. This does not modify the cost of the SPC but allows more flexibility when applying operators on the SPD.

5.3.5 Intensification procedure for the MC2DP

The intensification procedure for the MC2DP considers local operators that only modify delivery routes of the SPD and large operators that modify both the SPC and the SPD. Figure 5.2 presents these operators from the more local ones (on the left) to the more global ones (on the right).

The scheme of the intensification procedure is presented in Algorithm 8. It consists in a local search procedure where local operators are applied until a local minimum is reached. Then, in order to escape from this local minimum, a large operator is applied.

When the application of an operator Φ_{SPD} is infeasible, a large operator Φ that composes an operator Φ_{SPC} for the SPC and Φ_{SPD} is applied.

The whole procedure (local search plus application of large operators) is repeated until no improvement is obtained. In the following, we present the local search procedure for the SPD and the method to select a large operator to improve the MC2DP.

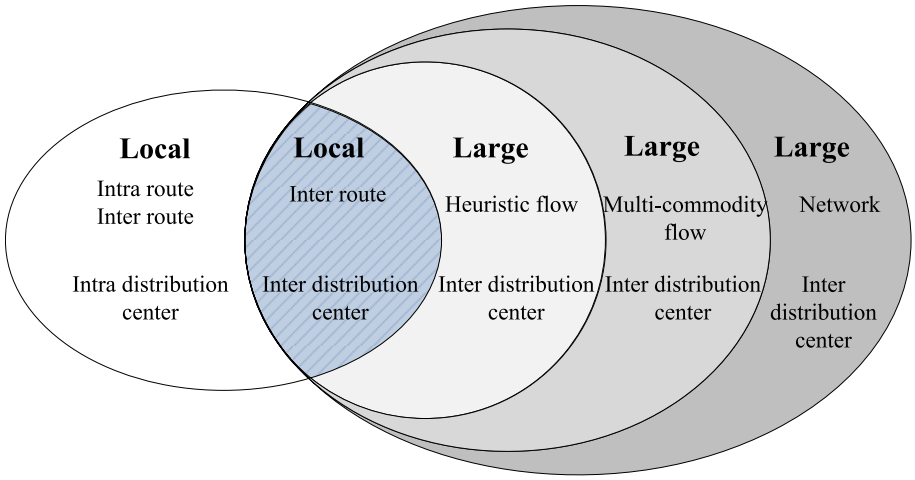


Fig. 5.2. Large moves.

Algorithm 8 Intensification procedure.

```
repeat
    repeat
        Apply local operators to improve the SPD
    until no local operator can improve the SPD
    Apply a large operator to improve the MC2DP
until no improvement is obtained with a large operator
```

5.3.5.1 Local search to improve the SPD

The local search procedure for the SPD is as it is described in Chapter 4. We recall here the main features of the local search procedure. It considers classical routing operators as insertion, swap, and 2-opt for customers; insertion, and swap for *customer-commodities*. These operators consider intra- and inter-route modifications as well as intra- and inter-distribution center modifications. The local search implements the best-improvement strategy, namely, for each operator, all feasible applications are evaluated, and the best one is implemented. The local search ends when no operator is able to find an improving solution.

5.3.5.2 Large moves to improve the MC2DP

In the local search, we improve the SPD solution without considering any modification on the SPC, which is limited for the improvement of the MC2DP solution.

The local search procedure is dedicated to improve the solution of the SPD and does not modify the SPC. This is a limitation for the improvement of the

whole solution for the MC2DP. Hence, the large operators presented in Section 5.3.3 are applied when the local search reaches a local optimum. The large operators consider an operator for the SPD which application is not feasible due to the available quantities at the distribution centers, and it tries to make it feasible by modifying the SPC.

The outline of the algorithm to apply large moves is presented in Algorithm 9. First, given the current solution, we collect all the applications of operators for the SPD that are infeasible because of the available quantities at the distribution centers. This means that all these moves concern customers or *customer-commodities* in two different routes starting from two different distribution centers. All the other moves are not involved in the application of large operators for the MC2DP. Note that the moves that would require the creation of a new route are not considered as well. Then, these local moves are sorted in increasing order with respect to the improvement, indicated as δ-cost, they would generate on the SPD. The δ-cost of a move is defined as the difference between the cost of the SPD solution after and before the implementation of the move. Thus, a negative δ-cost means the solution would be improved. Sorting the moves permits to first consider the most promising moves.

Then, we apply the large operators in the following order: heuristic flow, multi-commodity flow, and network, i.e. from the less time consuming to the most time consuming. Once the application of a large operator together with an infeasible local move is able to provide a feasible solution for the MC2DP, the large operator move is implemented on the current solution and the algorithm stops. Thus, a first-improvement strategy is applied when large operators are concerned. We propose to use this strategy instead of a best-improvement strategy since the evaluation of these large moves is not performed in constant time.

Note that infeasible local moves are ordered with respect to increasing values of the δ-cost that would cause on the SPD, and heuristic flow and multi-commodity flow operators do not modify the cost of the SPC. Thus, it is clear that the first feasible combination of a local move with the heuristic flow or multi-commodity flow operator provides the best possible improvement that their combination can produce and only local moves associated with negative δ-cost are worth checking. This reasoning can not be extended to the network operator since it may change the cost of the SPC.

Algorithm 9 Application of large operators to improve the MC2DP.

$s_{SPC} \oplus s_{SPD}$ is the current solution of the MC2DP
$\mathcal{L}$ is the set of infeasible moves from solution $s_{SPC} \oplus s_{SPD}$ for the local search operators
Sort $\mathcal{L}$ by ascending order with respect to the δ-cost of the moves
for all the moves in $\mathcal{L}$ **do**
    if the move has a negative δ-cost **then**
        if the heuristic flow operator finds a feasible solution **then**
            Obtain $(s_{SPC} \oplus s_{SPD})'$ by implementing the heuristic flow operator
            Update the s'_{SPC} by applying the *full truck* strategy
            return $(s_{SPC} \oplus s_{SPD})'$
        end if
    end if
end for
for all the moves in $\mathcal{L}$ **do**
    if the move has a negative δ-cost **then**
        if the multi-commodity flow operator finds a feasible solution **then**
            Obtain $(s_{SPC} \oplus s_{SPD})'$ by implementing the multi-commodity flow operator
            Update the s'_{SPC} by applying the *full truck* strategy
            return $(s_{SPC} \oplus s_{SPD})'$
        end if
    end if
end for
for all the moves in $\mathcal{L}$ **do**
    if the move has a δ-cost lower than a threshold δ^{max} **then**
        if the network operator finds a feasible solution **then**
            Obtain $(s_{SPC} \oplus s_{SPD})'$ by implementing the network operator
            Update the s'_{SPC} by applying the *full truck* strategy
            return $(s_{SPC} \oplus s_{SPD})'$
        end if
    end if
end for
return no improvement has been found

Since the network operator requires the resolution of a MIP, it has a high chance to be computationally expensive. Thus, only local moves associated with a δ-cost that is lower than a fixed threshold δ^{max} are tested coupled with the network operator.

If a large operator is implemented, before returning the new current solution,

the *full truck* strategy proposed in the sequential approaches (see Algorithm 10 in Chapter 4) is applied to the updated solution of the SPC. This allows more flexibility when applying local operators on the SPD, without modifying the cost of the solution.

5.4 Experimental results

In this section, we present the results obtained by the proposed integrated algorithm. The algorithm was implemented in C++ and ran on an Intel (R) Core(TM) i5-2500K, 3.30GHz, and 4GB of RAM. Cplex 12.6 is used for the resolution of the MIP presented in Section 5.3.3.1 and for the MIP and LP formulations on which the network and multi-commodity flow operators are based.

Section 5.4.1 describes the test instances on which we evaluate our algorithm. In Section 5.4.2, we study the sequential strategies that are applied to get the initial solution. Then, four configurations of the proposed integrated algorithm are tested and compared. These configuration differ in which large operators are used. In Section 5.4.3, we report the results obtained and compare them with the best known results given by the sequential approaches proposed in Chapter 4. Moreover, in Section 5.4.4, we analyze the effectiveness of each large operator.

The notations used to present the results are summarized in Table 5.1.

5.4.1 Instances

In order to evaluate the efficiency of our algorithm, we perform computational experiments on the instances proposed to evaluate the sequential approaches for the MC2DP in Chapter 4. 9 sets of instances were generated, all with 30 customers and 8 suppliers. From a base set of instances, other 8 sets were generated by modifying one of the characteristics of the base set. The base set $\mathcal{S}$ was built from the instances of the C-SDVRP (Archetti *et al.*, 2014). The creation of the instances for the MC2DP are detailed in Chapter 4 (see Section 4.5.1). In the following, we list the name and the main characteristics of each set of instances.

$\mathcal{S}$: The base set of instances. In this set, 8 suppliers, 30 customers and 2 distribution centers are considered. The locations of customers and suppliers are balanced, i.e., around each distribution center, 4 suppliers and 15 customers are located. Moreover, the quantity of a given commodity available at each supplier is the same. The

Table 5.1: Notations for computational results.

Symbol	Meaning
$nbIns$	Number of instances in each set
BKS	Best known solution cost when solving the MC2DP with sequential approaches (see Chapter 4)
$CostInit$	Cost of the initial solution cost for the MC2DP
$Cost$	Cost of the best solution of the MC2DP found by the proposed algorithm
$avg.\Delta$	Average percentage of improvement between $Cost$ and BKS ($\Delta = 100 * (Cost - BKS)/BKS$)
$min.\Delta$	Minimum percentage of improvement between $Cost$ and BKS ($\Delta = 100 * (Cost - BKS)/BKS$)
$max.\Delta$	Maximum percentage of improvement between $Cost$ and BKS ($\Delta = 100 * (Cost - BKS)/BKS$)
$avg.\Delta Init$	Average percentage of improvement between $Cost$ and $CostInit$ ($\Delta = 100 * (Cost - CostInit)/CostInit$)
$nbNBK$	Number of new best known solutions obtained
nbE	Number of solution values equal to the best known
nbW	Number of solution values worse than the best known
$\overline{nbIter}$	Average number of ALNS iterations in each set of instances
$\overline{nbHFO}$	Average number of calls to a `heuristic flow` operator
$\overline{nbMCFO}$	Average number of calls to a `multi-commodity flow` operator
$\overline{nbNO}$	Average number of calls to a `network` operator
$\overline{nbHFOimp}$	Average number of imrpovements when calling the `heuristic flow` operator
$\overline{nbMCFOimp}$	Average number of imrpovements when calling the `multi-commodity flow` operator
$\overline{nbNOimp}$	Average number of imrpovements when calling the `network` operator
$\overline{\Delta HFOimp}$	Average percentage of improvement with the `heuristic flow` operator ($\overline{\Delta HFOimp} = 100 * \overline{nbHFOimp}/\overline{nbHFO}$)
$\overline{\Delta MCFOimp}$	Average percentage of improvement with the `multi-commodity flow` operator ($\overline{\Delta MCFOimp} = 100 * \overline{nbMCFOimp}/\overline{nbMCFO}$)
$\overline{\Delta NOimp}$	Average percentage of improvement with the `network` operator ($\overline{\Delta NOimp} = 100 * \overline{nbNOimp}/\overline{nbNO}$)

other sets of instances are based on the instances in $\mathcal{S}$, with the modification of one characteristic.

$\mathcal{S}_1^S$ and $\mathcal{S}_2^S$: contain instances that are created by unbalancing the number of suppliers generated around each distribution center. In set $\mathcal{S}_1^S$, 6 suppliers are located around one of the distribution centers, while the other 2 suppliers are located around the other distribution center. In set $\mathcal{S}_2^S$, around one of the distribution center, 8 suppliers are located while no supplier is located around the other distribution center.

$\mathcal{S}_1^C$, $\mathcal{S}_2^C$, $\mathcal{S}_3^C$ and $\mathcal{S}_4^C$: contain instances that are created by unbalancing the number of customers around each distribution center.

$\mathcal{S}^O$: contain instances that are created by modifying the available quantities at suppliers. Unlike set $\mathcal{S}$ in which all suppliers have the same available quantity for a given commodity, the available quantities at suppliers are unbalanced in set $\mathcal{S}^O$.

$\mathcal{S}^D$: contain instances that are created by modifying the number of distribution centers. A third distribution center located in the middle of the segment joining the other two distribution centers is added.

All the instance sets contain 64 instances except for set $\mathcal{S}^O$ that contains 32

instances.

5.4.2 Sequential strategies for the initial solution

In this section we are interested in the computation of an initial solution for the integrated algorithm to solve the MC2DP. As mentioned in Section 5.3.2, six solutions are provided by running different sequential strategies, and the best one is kept as the initial solution. For each sequential strategy, the number of iterations of the ALNS to solve the SPD has been set to 1000.

In Table 5.2 we report for each set of instances, the number of times each sequential strategy has provided the initial solution, i.e., it has provided the best solution. The first column list the six sequential strategies. Then, there is one column for each instance set, with the number of times each strategy has provided the best solution. The last column reports the total over all the instance sets. The last row reports the total for each instance set, i.e., the number of instances in the set.

Table 5.2: Information about initial solution obtained by sequential strategies.

	Instance Set									
nb Strategy	$\mathcal{S}$	$\mathcal{S}_1^S$	$\mathcal{S}_2^S$	$\mathcal{S}_1^C$	$\mathcal{S}_2^C$	$\mathcal{S}_3^C$	$\mathcal{S}_4^C$	$\mathcal{S}^O$	$\mathcal{S}^D$	Total
nb *SPD infinite* → *SPC*	8	0	0	1	1	0	0	0	2	12
nb *SPD finite balanced* → *SPC*	17	0	1	20	18	14	20	1	0	91
nb *SPD finite supplier based* → *SPC*	8	39	23	24	17	31	25	9	37	**213**
nb *SPC not full truck* → *SPD*	5	0	14	0	2	0	0	0	5	26
nb *SPC full truck* → *SPD*	15	11	10	8	13	9	6	3	13	88
nb *SPC full truck customer based* → *SPD*	11	14	16	11	13	10	13	19	7	**114**
nbIns	64	64	64	64	64	64	64	32	64	544

From Table 5.2 it can be noticed that two sequential strategies that often provide the best initial solution are *SPD finite supplier based* → *SPC* and *SPC full truck customer based* → *SPD*. However, the strategies *SPD finite balanced* → *SPC* and *SPC full truck* → *SPD* are able to provide the best solution, especially for some sets of instances. Hence, starting with different strategies and choosing the best seems to be interesting since it permits us to have a good initial solution for all the instance sets.

5.4.3 Evaluation of the proposed algorithm

The integrated ALNS algorithm is run with a time limit of 3 minutes after the computation of an initial solution. There is no limit on the number of iterations.

Note that the computation time to generate an initial solution is around 1 minute for all the instances. We choose to use a time limit instead of a number of iterations since the large operators, especially the network operator, may be time consuming. From our point of view, imposing a time limit allows for a fair comparison.

In order to evaluate the performances of the proposed algorithm, and more specifically the impact of the large operators, we test four configurations of the algorithm on all the sets of instances. The results are compared with the best known results obtained with the sequential approaches where 5000 iterations are allowed to the ALNS to solve the SPD. The four configurations are the following:

- *HFO*: Only the `heuristic flow` operator is considered.
- *HFO+MCFO*: `Heuristic flow` and `multi-commodity flow` operators are considered.
- *ALL*: All the operators are considered, and no threshold is used to apply the network operator ($\delta^{max} = \infty$).
- *ALL+δ^{max}*: All the operators are considered, and in the intensification procedure, a threshold is used to apply the network operator (see Section 5.3.5.2). δ^{max} is set to 1% of the maximum δ-cost of the infeasible moves for the local search operators.

For each configuration of the algorithm, and each instance set, Table 5.3 reports the results of the solution obtained with the integrated algorithm to solve the MC2DP compared with the best known values provided by the sequential approaches. In Table 5.3, the first two columns describe the name of the instance set and the configuration of the integrated algorithm. Then, the meaning of the following columns is described in Table 5.1. Note that for $avg.\Delta$, $min.\Delta$, $max.\Delta$ and $avg.\Delta Init$, a negative value means that the integrated ALNS has been able to improve the results. For each set of instances, the lowest $avg.\Delta$, $min.\Delta$ and $max.\Delta$, and the highest $nbNBK$ for the four configurations are provided in bold.

The results in Table 5.3 indicate that considering heuristic flow, multi-commodity flow and network operators for the large moves is beneficial. Indeed, both configurations *ALL* and *ALL+δ^{max}* improve the average results from the best known solutions ($avg.\Delta$) for all the sets of instances, while it is not the case for the configurations *HFO* and *HFO+MCFO* that do not consider all the large operators.

Table 5.3: Results on all instance sets.

Instance Set	Configuration	$avg.\Delta$	$min.\Delta$	$max.\Delta$	$nbNBK$	nbE	nbW	$avg.\Delta Init$
$\mathcal{S}$	*HFO*	0.02	-1.02	**0.67**	2	52	10	-0.02
	HFO+MCFO	0.02	-1.05	**0.67**	3	51	10	-0.02
	ALL	-0.05	-2.33	**0.67**	5	48	11	-0.09
	ALL+δ^{max}	**-0.23**	**-2.35**	**0.67**	**19**	38	7	-0.27
$\mathcal{S}_1^S$	*HFO*	0.04	-1.59	0.86	6	38	20	-0.07
	HFO+MCFO	-0.06	-1.86	0.86	13	36	15	-0.17
	ALL	-0.02	-2.95	1.86	10	34	20	-0.13
	ALL+δ^{max}	**-0.24**	**-3.08**	**0.85**	**19**	29	16	-0.35
$\mathcal{S}_2^S$	*HFO*	-0.31	-3.62	**0.70**	**22**	37	5	-0.41
	HFO+MCFO	-0.48	**-5.12**	**0.70**	**22**	37	5	-0.58
	ALL	-0.04	-1.59	1.20	13	41	10	-0.13
	ALL+δ^{max}	**-0.65**	-4.59	**0.70**	**22**	38	4	-0.75
$\mathcal{S}_1^C$	*HFO*	-0.12	**-2.91**	1.53	16	34	14	-0.10
	HFO+MCFO	-0.21	**-2.91**	1.53	20	32	12	-0.19
	ALL	-0.19	**-2.91**	1.53	19	32	13	-0.16
	ALL+δ^{max}	**-0.49**	**-2.91**	**0.71**	**33**	28	3	-0.47
$\mathcal{S}_2^C$	*HFO*	-0.10	-1.30	2.39	18	33	13	-0.15
	HFO+MCFO	-0.14	-1.52	2.39	20	33	11	-0.20
	ALL	-0.08	-1.85	2.39	15	34	15	-0.14
	ALL+δ^{max}	**-0.56**	**-3.63**	**0.94**	**35**	23	6	-0.62
$\mathcal{S}_3^C$	*HFO*	0.08	-1.07	2.27	15	34	15	-0.10
	HFO+MCFO	0.03	-1.08	2.02	18	33	13	-0.16
	ALL	-0.02	**-3.64**	2.15	18	32	14	-0.20
	ALL+δ^{max}	**-0.32**	-3.03	**1.36**	**31**	26	7	-0.50
$\mathcal{S}_4^C$	*HFO*	-0.03	-1.88	**1.35**	13	37	14	-0.15
	HFO+MCFO	-0.06	-1.88	**1.35**	15	35	14	-0.18
	ALL	-0.11	-3.35	**1.35**	17	33	14	-0.23
	ALL+δ^{max}	**-0.51**	**-3.70**	**1.35**	**29**	27	8	-0.63
$\mathcal{S}^O$	*HFO*	-0.32	-2.63	**0.32**	11	14	7	-0.35
	HFO+MCFO	**-0.62**	-3.32	**0.32**	**13**	13	6	-0.65
	ALL	-0.30	-2.77	0.63	11	12	9	-0.34
	ALL+δ^{max}	-0.52	**-3.42**	0.37	12	12	8	-0.55
$\mathcal{S}^D$	*HFO*	-0.04	-2.21	1.51	8	46	10	-0.04
	HFO+MCFO	-0.06	-2.21	1.51	13	42	9	-0.07
	ALL	-0.12	-2.30	1.51	11	43	10	-0.13
	ALL+δ^{max}	**-0.35**	**-3.58**	**1.24**	**26**	32	6	-0.36

Moreover, it is beneficial to consider a threshold on the δ-cost of the moves for the SPD to apply the network operator. The configuration $ALL{+}\delta^{max}$ provides the best average improvement ($avg.\Delta$) for all the sets of instances (except set $\mathcal{S}^O$). Note that the configuration $HFO{+}MCFO$ provides better average improvements ($avg.\Delta$) for all the sets of instances. This indicates that the multi-commodity flow operator is very beneficial.

The use of the network operator without threshold is not always beneficial. In average it permits to improve the best known solutions for all the sets of instances, but for some sets of instances, the average improvement is worse than the configurations that do not consider the network operator. As an example, for the instance sets $\mathcal{S}_2^S$, $\mathcal{S}_2^C$ and $\mathcal{S}^O$, the value of $avg.\Delta$ is the lowest with the configuration ALL.

It is also interesting to point out that some configurations may be able to provide better solutions, not on average, but for some instances, than the configuration $ALL{+}\delta^{max}$. For example, for the instance sets $\mathcal{S}_2^S$ and $\mathcal{S}_3^C$, the best improvement ($min.\Delta$) is obtained with the configurations HFO or ALL. Moreover, for the instance set $\mathcal{S}_1^C$, all the four configurations provide the same best improvement ($min.\Delta$).

If we focus on the results provided by the configuration $ALL{+}\delta^{max}$, on the whole set of instances, we obtain a new best known solution ($nbNBK$) for 42% of the instances, a solution with the same cost than the best known (nbE) for 47% of the instances, and a solution worse than the best known (nbW) for 12% of the instances. The best improvement ($min.\Delta$) goes up to 3.70% (for the set $\mathcal{S}_4^C$). In the worst case ($max.\Delta$), we obtain a solution 1.24% worse than the best-known solution obtained with a sequential approach (for the set $\mathcal{S}_3^C$).

For the instance set $\mathcal{S}_1^C$, the values of $avg.\Delta Init$ are greater than the values of $avg.\Delta$, which means that, some initial solutions are better than the best known. This is possible since the best-known solutions have been run with 5000 iterations of the ALNS for the SPD, while the initial solutions are run with only 1000 iterations for the SPD. If the best sequential strategy is a strategy where the SPD is solved first, the ALNS with 5000 iterations will provide a better solution for the SPD. However, the SPC is then solved based on the solution of the SPD. It is then possible to obtain a worse solution for the SPC associated with the best solution of the SPD. Hence, using a sequential strategy starting with the SPD, the best solution for the MC2DP may be obtained from a solution of the SPD that is not the best. In fact, the similar conclusion have been obtained in Chapter 4 (see Section 4.5.2). That

is why the initial solution with less iterations of the ALNS may be better than the best known solution.

5.4.4 Analysis of the large operators

In this section, we analyze the performance of the large operators for each configuration of the integrated algorithm. The performance indicators are the number of calls to these operators, and the percentage of calls that resulted in an improvement of the current solution. In Table 5.4, the first two columns indicate the instance set and the configuration. The following three columns indicate, for each instance set and each configuration, the average number of calls to the heuristic flow, the multi-commodity flow and the network operators during the 3 minutes of computation of the ALNS. Then, the following three columns indicate the average percentage of calls to the heuristic flow, the multi-commodity flow and the network operators that succeeded to find an improved solution for the MC2DP. The last column reports the average number of ALNS iterations during the 3 minutes of computation.

From the results in Table 5.4, we note that the heuristic flow operator has rather low performances (usually between 2% and 6% of success). The multi-commodity flow operator has very good performances (generally around 20% success), except within the configuration *ALL*. The performances of the network operator are not very good. In the configuration $ALL{+}\delta^{max}$, the success rate is around 2%.

The configuration *ALL* seems to be unsuitable for an ALNS framework. Indeed, there are very few iterations (usually around 20) due to a lot of calls to the network operator that is very time consuming. There are usually more than 10 000 calls to the network operator, and very few calls to the other two operators since there are few iterations. Finally, the performance of the network operator is poor since the rate of success is 0.01%.

However, we can notice that using a threshold for the network operator is very beneficial. Indeed, in the configuration $ALL{+}\delta^{max}$, the number of iterations and number of calls to the network operator remain reasonable, and the rate of success for the network operator reaches 2%. In that configuration, the number of calls to each operator and the number of iterations have the same order of magnitude, which seems to be reasonable. Please note that the heuristic flow and the multi-commodity flow operators may be called less than the network operator since they only concern local moves for the SPD with a negative δ-cost.

Table 5.4: Effectiveness of the large operators.

Instance Set	Configuration	$\overline{nbHFO}$	$\overline{nbMCFO}$	$\overline{nbNO}$	$\overline{\Delta HFOimp}$	$\overline{\Delta MCFOimp}$	$\overline{\Delta NOimp}$	$\overline{nbIter}$
$\mathcal{S}$	*HFO*	2479	-	-	3.90	-	-	6379
	HFO+MCFO	1833	819	-	6.09	20.80	-	5630
	ALL	15	12	15686	2.12	5.26	0.00	30
	ALL+δ^{max}	1223	631	2104	5.49	16.52	2.17	4059
$\mathcal{S}_1^S$	*HFO*	9662	-	-	1.37	-	-	7334
	HFO+MCFO	4155	1999	-	2.82	20.20	-	6019
	ALL	45	35	14002	1.15	5.12	0.01	25
	ALL+δ^{max}	2291	954	3468	2.77	23.76	1.27	3465
$\mathcal{S}_2^S$	*HFO*	6422	-	-	0.44	-	-	23464
	HFO+MCFO	1531	419	-	2.00	38.22	-	22116
	ALL	3	1	1018	1.15	26.44	0.01	20863
	ALL+δ^{max}	578	163	376	2.49	40.30	2.20	22178
$\mathcal{S}_1^C$	*HFO*	4356	-	-	2.35	-	-	6278
	HFO+MCFO	2234	979	-	5.30	24.13	-	5624
	ALL	24	18	15201	2.50	5.65	0.01	23
	ALL+δ^{max}	1496	771	3250	4.74	17.95	1.41	3530
$\mathcal{S}_2^C$	*HFO*	3555	-	-	2.46	-	-	6104
	HFO+MCFO	1703	809	-	6.77	22.32	-	5611
	ALL	29	23	14999	2.41	5.76	0.01	25
	ALL+δ^{max}	1446	734	3241	5.77	17.59	1.59	3490
$\mathcal{S}_3^C$	*HFO*	8380	-	-	1.21	-	-	6998
	HFO+MCFO	4122	1960	-	3.90	20.53	-	5698
	ALL	26	20	14568	2.19	5.78	0.01	22
	ALL+δ^{max}	2017	1008	4389	3.49	17.46	1.14	2755
$\mathcal{S}_4^C$	*HFO*	7568	-	-	1.49	-	-	6751
	HFO+MCFO	3591	1729	-	3.67	23.05	-	5593
	ALL	29	22	14495	1.81	6.08	0.01	23
	ALL+δ^{max}	1944	1118	4275	3.43	13.83	1.10	2888
$\mathcal{S}^O$	*HFO*	92881	-	-	0.04	-	-	5360
	HFO+MCFO	45726	43693	-	0.15	0.46	-	4327
	ALL	344	340	9115	0.00	0.16	0.00	13
	ALL+δ^{max}	3385	3159	6385	0.42	1.06	0.09	517
$\mathcal{S}^D$	*HFO*	4129	-	-	2.22	-	-	5424
	HFO+MCFO	1608	880	-	5.95	13.65	-	4723
	ALL	13	10	14951	2.57	5.52	0.00	22
	ALL+δ^{max}	1202	602	2052	4.99	17.27	2.15	3326

Last, we can provide some interesting observations from the results for some specific instance sets. In the instance set $\mathcal{S}_2^S$, 8 suppliers are located around one distribution center while no supplier is located around the other distribution center. For this instance set, the number of iterations increases significantly (around 20 000), while the number of calls to the large operators is low. In fact, due to the specific characteristic of these instances, it is likely to have a current solution with all the flow of commodities passing through a single distribution center. In that case, there are no inter-distribution center moves for the SPD.

In the instance set $\mathcal{S}^O$, the available quantities at the suppliers is unbalanced: suppliers located around one distribution center have an availability of the first commodity that is greater than the other, while for suppliers located around the other distribution center it is the opposite. In that case, it is likely that the current solution has a particular structure: many customers can be delivered by both distribution centers. Hence, there are a lot of inter-distribution center moves for the SPD. Therefore the large operators are called more frequently. For these instances, the multi-commodity flow operator has poor performances (usually less than 1% of success). This may be due to the fact that it is difficult to modify the flow of commodities when the limited available quantities at the suppliers are unbalanced.

5.5 Conclusions and future works

In this chapter, we developed an integrated solution approach for the MC2DP. Based on the sequential approaches presented in Chapter 4, we improved the ALNS algorithm developed for the delivery subproblem, so that it can modify both collection and delivery decisions. We add new large operators able to modify the collection decisions in order to make feasible a move between two distribution centers for the delivery subproblem. Three large operators have been proposed: the heuristic flow is a fast heuristic that modifies the collection subproblem at zero cost; the multi-commodity flow operator uses a linear programming formulation to optimally decide the flow of commodities at zero cost; and the network operator that consists in solving a mixed integer program in order to take the decisions optimally for the collection subproblem. This last operator is the most efficient but at the same time the most expensive computationally. This integrated approach has been evaluated and compared to the best solutions obtained with the sequential approaches. The

results showed that the proposed operators are all necessary, and on average, the integrated approach is able to outperform the sequential approaches.

In order to further improve the results of the integrated approach, we provide some short term perspectives. From the results, it is clear that the network operator should not be called for all potential moves for the delivery subproblem. The use of a threshold on the cost of the moves is efficient, and we could consider providing other criteria to reduce the calls to the network operator. Such criteria may be based on the cost of the moves, but also on the quantity to move from a distribution center to another.

Another perspective is to go further in the integrated algorithm. The idea is to develop "destroy and repair" heuristics for the ALNS related to both collection and delivery subproblems. For example, we could destroy and repair the collection subproblem by removing and adding trips between suppliers and distribution centers. The difficulty in this approach is to ensure the feasibility of the whole problem after modifying both subproblems. As an example, in the collection subproblem, moving one truck from a distribution center to another one would require a deeper modification in the delivery subproblem: several customers should be delivered from another distribution center.

Another perspective to improve the integrated approach for the MC2DP is to develop a fast solving method for the network operator. Solving a mixed integer program requires a lot of computation time, and would not be suitable for large instances. Hence, we may propose to develop fast heuristic solving methods for the underlying problem. This could allow to address large size instances, like the ones proposed in the case study in Chapter 4.

5. AN INTEGRATED OPTIMIZATION APPROACH FOR A MULTI-COMMODITY TWO-ECHELON DISTRIBUTION PROBLEM

Chapter 6

Conclusions and perspectives

In this thesis, we studied vehicle routing problems considering multiple commodities, with applications in the local fresh food supply chains. The studied supply chain contains two echelons with three sets of actors: suppliers, distribution centers and customers. Suppliers are farmers that produce some fresh foods. Distribution centers are in charge of consolidation and delivery of the products to customers. Distribution centers collect products from the suppliers that perform direct trips. Products are delivered to the customers with a fleet of vehicles performing routes. Each customer requires several commodities, and the farmers produce a limited quantity of these commodities. For the minimization of the transportation cost, it is beneficial that a single customer is delivered by several vehicles. However, for the convenience of the customer, it is imposed that a single commodity is delivered at once by a single vehicle. Hence, different commodities have been explicitly considered. The complete problem is named Multi-Commodity two-echelon Distribution Problem (MC2DP). The restricted problem that addresses only the delivery from a single distribution center is named Commodity constrained Split Delivery Vehicle Routing Problem (C-SDVRP).

The main contributions of this thesis are the following.

In Chapter 2, we classified the routing problems that explicitly consider multiple commodities. The main results have been to understand why it is necessary to explicitly consider these multiple commodities, how they are considered in the models and solving methods, and what are the main applications.

In Chapter 3, we focused on the delivery subproblem with a single distribution center (C-SDVRP). We develop a dedicated heuristic algorithm based on an ALNS

framework. This is the first heuristic specifically designed to provide high-quality solutions for the medium and large size instances. The results showed that our ALNS algorithm is very effective in finding high-quality solutions on large size instances.

In Chapter 4, we addressed the whole problem (MC2DP) with collection and delivery operations and multiple distribution centers. In order to tackle this complex problem, we proposed a decomposition approach in which the collection and delivery problems are sequentially solved. For each sequential approach, three strategies have been developed in order to take into account the impact of the solution of the first subproblem on the second subproblem solution. The best algorithm usually does not provide the lowest cost either for the collection or for the delivery subproblem. The algorithms have been tested on a case study on a short and local fresh food supply chain with two types of customers: school canteens and supermarkets.

In Chapter 5, we developed an integrated approach for the MC2DP. This approach improved the proposed ALNS for the delivery subproblem by adding more global moves that also modify the collection subproblem. The computational results showed that this approach is able to outperform all the proposed sequential approaches.

This thesis gives rise to several future research directions. On the methodological point of view, the first perspective is to improve the integrated approach for the MC2DP. Our main idea is to develop an ALNS for the whole problem, i.e. for both collection and delivery operations. To do so, we propose to add destroy and repair operators for the collection subproblem, for example to remove one direct trip between a supplier and a distribution center. Hence, at each iteration of the ALNS both collection and delivery subproblems would be destroyed and repaired. After the repair, we have to ensure that the solutions of the subproblems are consistent, i.e. each distribution center collects more products than what is delivered. Another perspective to improve the integrated approach for the MC2DP is to develop a fast solving method for the collection subproblem in order to be able to tackle large size instances like the one addressed in the case study in Chapter 4.

Another research perspective of our work is to adapt the proposed ALNS algorithm for the C-SDVRP to tackle other variants of routing problems with multi-commodity constrained split deliveries. The solving methods for VRPs with multiple commodities usually duplicate the customers for each required commodities, then work with *customer-commodities* in the resolution algorithms. Like we did for the

C-SDVRP, we may propose algorithms that use a double representation of the solution by aggregating or not the *customer-commodities*. For example, we could try to adapt the proposed ALNS to solve some VRPs with muti-compartments vehicles.

Furthermore, we could adapt the proposed method to some extensions of the problem. First, in this thesis, we do not consider the capacity of the distribution centers. From a practical perspective, we could include capacity constraints at the distribution centers. Second, in this thesis we considered that the commodities are collected from suppliers and delivered to distribution centers through direct trips. We considered that the suppliers provide large supply quantities and deliver their products to distribution centers by performing one or more round trips. However, further research could consider more suppliers with lower supply quantities. Then, it would be interesting to perform routes to collect the products from the suppliers. To solve this version of the problem, we could keep the same solving framework, but the resolution of the collection subproblem to optimality would be more challenging.

Finally, from the application point of view, we could study the problem where some direct deliveries from a supplier to some customers are allowed. Indeed, in the context of short and local fresh food supply chain, it is interesting for the farmers to have direct contact with customers located near their farm. Hence, we could consider a logistics system that mixes direct deliveries with deliveries through distribution centers.

References

Abounacer, R., Rekik, M. & Renaud, J. (2014). An exact solution approach for multi-objective location-transportation problem for disaster response. *Computers & Operations Research*, **41**, 83–93. 35

Anily, S. & Bramel, J. (1999). Approximation algorithms for the capacitated traveling salesman problem with pickups and deliveries. *Naval Research Logistics (NRL)*, **46**, 654–670. 21

Archetti, C. & Speranza, M.G. (2012). Vehicle routing problems with split deliveries. *International transactions in operational research*, **19**, 3–22. 9, 43

Archetti, C., Savelsbergh, M.W. & Speranza, M.G. (2008). To split or not to split: That is the question. *Transportation Research Part E: Logistics and Transportation Review*, **44**, 114–123. 43

Archetti, C., Campbell, A.M. & Speranza, M.G. (2014). Multicommodity vs. single-commodity routing. *Transportation Science*, **50**, 461–472. vii, 23, 25, 43, 44, 45, 47, 48, 62, 63, 67, 69, 76, 87, 104, 140, 182

Archetti, C., Bianchessi, N. & Speranza, M.G. (2015). A branch-price-and-cut algorithm for the commodity constrained split delivery vehicle routing problem. *Computers & Operations Research*, **64**, 1–10. 25, 46, 64, 67, 68, 69, 76

Avella, P., Boccia, M. & Sforza, A. (2004). Solving a fuel delivery problem by heuristic and exact approaches. *European Journal of Operational Research*, **152**, 170–179. 16

Azi, N., Gendreau, M. & Potvin, J.Y. (2014). An adaptive large neighborhood search for a vehicle routing problem with multiple routes. *Computers & Operations Research*, **41**, 167–173. 46

Barbarosoğlu, G. & Arda, Y. (2004). A two-stage stochastic programming framework for transportation planning in disaster response. *Journal of the Operational Research Society*, **55**, 43–53. 35

Battarra, M., Monaci, M. & Vigo, D. (2009). An adaptive guidance approach for the heuristic solution of a minimum multiple trip vehicle routing problem. *Computers & Operations Research*, **36**, 3041–3050. 29, 30

Battarra, M., Cordeau, J.F. & Iori, M. (2014). Chapter 6: pickup-and-delivery problems for goods transportation. In *Vehicle Routing: Problems, Methods, and Applications, Second Edition*, 161–191, SIAM. 21

Beasley, J.E. (1983). Route first-cluster second methods for vehicle routing. *Omega*, **11**, 403–408. 96

Berbeglia, G., Cordeau, J.F., Gribkovskaia, I. & Laporte, G. (2007). Static pickup and delivery problems: a classification scheme and survey. *Top*, **15**, 1–31. 21

Berti, G. & Mulligan, C. (2016). Competitiveness of small farms and innovative food supply chains: The role of food hubs in creating sustainable regional and local food systems. *Sustainability*, **8**, 616. 1, 2, 79, 185, 186

Bianco, L., Caramia, M. & Giordani, S. (2009). A bilevel flow model for hazmat transportation network design. *Transportation Research Part C: Emerging Technologies*, **17**, 175–196. 33

Bianco, L., Caramia, M., Giordani, S. & Piccialli, V. (2013). Operations research models for global route planning in hazardous material transportation. In *Handbook of OR/MS Models in Hazardous Materials Transportation*, 49–101, Springer. 33

Blanquart, C., Gonçalves, A., Vandenbossche, L., Kebir, L., Petit, C. & Traversac, J.B. (2010). The logistic leverages of short food supply chains performance in terms of sustainability. In *12th World Conference on Transport Research*. 1, 80, 185

Boccia, M., Crainic, T.G., Sforza, A. & Sterle, C. (2018a). Multi-commodity location-routing: Flow intercepting formulation and branch-and-cut algorithm. *Computers & Operations Research*, **89**, 94–112. 31

Boccia, M., Crainic, T.G., Sforza, A. & Sterle, C. (2018b). Multi-commodity location-routing: Flow intercepting formulation and branch-and-cut algorithm. *Computers & Operations Research*, **89**, 94–112. 32

Bosona, T.G. & Gebresenbet, G. (2011). Cluster building and logistics network integration of local food supply chain. *Biosystems engineering*, **108**, 293–302. 81

Braekers, K., Ramaekers, K. & Van Nieuwenhuyse, I. (2016). The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering*, **99**, 300–313. 8

Brown, G.G. & Graves, G.W. (1981). Real-time dispatch of petroleum tank trucks. *Management science*, **27**, 19–32. 16

Burks, R.E. (2006). *An adaptive tabu search heuristic for the location routing pickup and delivery problem with time windows with a theater distribution application*. Ph.D. thesis, Graduate School of Engineering and Management, Air Force Institute of Technology, Ohio. 31

Caramia, M. & Guerriero, F. (2010). A milk collection problem with incompatibility constraints. *Interfaces*, **40**, 130–143. 20

Cattaruzza, D., Absi, N., Feillet, D. & Vigo, D. (2014). An iterated local search for the multi-commodity multi-trip vehicle routing problem with time windows. *Computers & Operations Research*, **51**, 257–267. 30

Cattaruzza, D., Absi, N. & Feillet, D. (2016). Vehicle routing problems with multiple trips. *4OR*, **14**, 223–259. 29

Cattrysse, D.G. & Van Wassenhove, L.N. (1992). A survey of algorithms for the generalized assignment problem. *European Journal of Operational Research*, **60**, 260–272. 95

Ceselli, A., Righini, G. & Salani, M. (2009). A column generation algorithm for a rich vehicle-routing problem. *Transportation Science*, **43**, 56–69. 25

Chajakis, E.D. & Guignard, M. (2003). Scheduling deliveries in vehicles with multiple compartments. *Journal of Global Optimization*, **26**, 43–78. 16, 20

Chen, L., Liu, Y. & Langevin, A. (2019). A multi-compartment vehicle routing problem in cold-chain distribution. *Computers & Operations Research*, **111**, 58 – 66. 20

Christiansen, M., Fagerholt, K., Rachaniotis, N.P., Tveit, I. & Øverdal, M.V. (2015). A decision support model for routing and scheduling a fleet of fuel supply vessels. In F. Corman, S. Voß & R.R. Negenborn, eds., *Computational Logistics*, 46–60, Springer International Publishing, Cham. 17

Coelho, L.C. & Laporte, G. (2013). A branch-and-cut algorithm for the multi-product multi-vehicle inventory-routing problem. *International Journal of Production Research*, **51**, 7156–7169. 26, 27

Coelho, L.C. & Laporte, G. (2015). Classification, models and exact algorithms for multi-compartment delivery problems. *European Journal of Operational Research*, **242**, 854–864. 15

Coelho, L.C., Cordeau, J.F. & Laporte, G. (2013). Thirty years of inventory routing. *Transportation Science*, **48**, 1–19. 26

Cordeau, J.F., Laganà, D., Musmanno, R. & Vocaturo, F. (2015). A decomposition-based heuristic for the multiple-product inventory-routing problem. *Computers & Operations Research*, **55**, 153–166. 27

Cornillier, F., Boctor, F.F., Laporte, G. & Renaud, J. (2008a). An exact algorithm for the petrol station replenishment problem. *Journal of the Operational Research Society*, **59**, 607–615. 15, 16

Cornillier, F., Boctor, F.F., Laporte, G. & Renaud, J. (2008b). A heuristic for the multi-period petrol station replenishment problem. *European Journal of Operational Research*, **191**, 295–305. 16

Cornillier, F., Laporte, G., Boctor, F.F. & Renaud, J. (2009). The petrol station replenishment problem with time windows. *Computers & Operations Research*, **36**, 919–935. 16

Cornillier, F., Boctor, F. & Renaud, J. (2012). Heuristics for the multi-depot petrol station replenishment problem with time windows. *European Journal of Operational Research*, **220**, 361–369. 17

Cuda, R., Guastaroba, G. & Speranza, M.G. (2015). A survey on two-echelon routing problems. *Computers & Operations Research*, **55**, 185–199. 80

Derigs, U., Gottlieb, J., Kalkoff, J., Piesche, M., Rothlauf, F. & Vogel, U. (2011). Vehicle routing with compartments: applications, modelling and heuristics. *OR spectrum*, **33**, 885–914. 13, 14, 15

Drexl, M. & Schneider, M. (2015). A survey of variants and extensions of the location-routing problem. *European Journal of Operational Research*, **241**, 283–308. 30

Dror, M. & Trudeau, P. (1989). Savings by split delivery routing. *Transportation Science*, **23**, 141–145. 9, 43

Dror, M. & Trudeau, P. (1990). Split delivery routing. *Naval Research Logistics (NRL)*, **37**, 383–402. 9, 43

El Fallahi, A., Prins, C. & Calvo, R.W. (2008). A memetic algorithm and a tabu search for the multi-compartment vehicle routing problem. *Computers & Operations Research*, **35**, 1725–1741. 14, 15, 20

Erdoğan, G., Cordeau, J.F. & Laporte, G. (2010). A branch-and-cut algorithm for solving the non-preemptive capacitated swapping problem. *Discrete Applied Mathematics*, **158**, 1599–1614. 22

Erkut, E. & Gzara, F. (2008). Solving the hazmat transport network design problem. *Computers & Operations Research*, **35**, 2234–2247. 33

Erkut, E., Tjandra, S.A. & Verter, V. (2007). Chapter 9 Hazardous Materials Transportation. In C. Barnhart & G. Laporte, eds., *Handbooks in Operations Research and Management Science*, vol. 14 of *Transportation*, 539–621, Elsevier. 33

Flores, H. & Villalobos, J.R. (2018). A modeling framework for the strategic design of local fresh-food systems. *Agricultural Systems*, **161**, 1–15. 81

François, V., Arda, Y., Crama, Y. & Laporte, G. (2016). Large neighborhood search for multi-trip vehicle routing. *European Journal of Operational Research*, **255**, 422–441. 46, 51, 60, 64

Ghasemi, P., Khalili-Damghani, K., Hafezolkotob, A. & Raissi, S. (2019). Uncertain multi-objective multi-commodity multi-period multi-vehicle location-allocation model for earthquake evacuation planning. *Applied Mathematics and Computation*, **350**, 105–132. 36

Ghiani, G., Mourão, C., Pinto, L. & Vigo, D. (2013). Chapter 15: Routing in Waste Collection Applications. In *Arc Routing: Problems, Methods, and Applications*, 351–370, SIAM. 18

Ghorbani, A. & Jokar, M.R.A. (2016). A hybrid imperialist competitive-simulated annealing algorithm for a multisource multi-product location-routing-inventory problem. *Computers & Industrial Engineering*, **101**, 116–127. 29

Gianessi, P., Alfandari, L., Létocart, L. & Wolfler Calvo, R. (2015). The multicommodity-ring location routing problem. *Transportation Science*, **50**, 541–558. 31, 32

Golden, B.L., Magnanti, T.L. & Nguyen, H.Q. (1977). Implementing vehicle routing algorithms. *Networks*, **7**, 113–148. 10

Golden, B.L., Assad, A.A. & Wasil, E.A. (2002). Routing vehicles in the real world: applications in the solid waste, beverage, food, dairy, and newspaper industries. In *The vehicle routing problem*, 245–286, SIAM. 18

Golden, B.L., Kovacs, A.A. & Wasil, E.A. (2014). Chapter 14: Vehicle Routing Applications in Disaster Relief. In *Vehicle Routing: Problems, Methods, and Applications, Second Edition*, 409–436, SIAM. 34

Gribkovskaia, I., Gullberg, B.O., Hovden, K.J. & Wallace, S.W. (2006). Optimization model for a livestock collection problem. *International Journal of Physical Distribution & Logistics Management*, **36**, 136–152. 19

Gribkovskaia, I., Halskau, Ø., Laporte, G. & Vlček, M. (2007). General solutions to the single vehicle routing problem with pickups and deliveries. *European Journal of Operational Research*, **180**, 568–584. 76

Gschwind, T., Bianchessi, N. & Irnich, S. (2019). Stabilized branch-price-and-cut for the commodity-constrained split delivery vehicle routing problem. *European Journal of Operational Research*, **278**, 91–104. 25

Gu, W., Cattaruzza, D., Ogier, M. & Semet, F. (2019). Adaptive large neighborhood search for the commodity constrained split delivery VRP. *Computers & Operations Research*, **112**, 104761. 24, 25

Guastaroba, G., Speranza, M.G. & Vigo, D. (2016). Intermediate facilities in freight transportation planning: a survey. *Transportation Science*, **50**, 763–789. 80

Gustavsson, J., Cederberg, C., Sonesson, U., van Otterdijk, R. & Meybeck, A. (2012). *Pertes et gaspillages alimentaires dans le monde: ampleur, causes et prévention : étude menée pour le Congrès international SAVE FOOD ! à Interpack 2011, Düsseldorf, Allemagne*. Organisation des Nations Unies pour l'alimentation et l'agriculture (FAO). 2, 186

Henke, T., Speranza, M. & Wäsher, G. (2015). The multi-compartment vehicle routing problem with flexible compartment sizes. *European Journal of Operational Research*, **246**, 730–743. 14, 15

Henke, T., Speranza, M.G. & Wäscher, G. (2019). A branch-and-cut algorithm for the multi-compartment vehicle routing problem with flexible compartment sizes. *Annals of Operations Research*, 1–18. 15, 18

Hernández-Pérez, H. & Salazar-González, J.J. (2004). A branch-and-cut algorithm for a traveling salesman problem with pickup and delivery. *Discrete Applied Mathematics*, **145**, 126–139. 21

Hernández-Pérez, H. & Salazar-González, J.J. (2007). The one-commodity pickup-and-delivery traveling salesman problem: Inequalities and algorithms. *Networks: An International Journal*, **50**, 258–272. 22

Hernández-Pérez, H. & Salazar-González, J.J. (2009). The multi-commodity one-to-one pickup-and-delivery traveling salesman problem. *European Journal of Operational Research*, **196**, 987–995. 22

Hernández-Pérez, H. & Salazar-González, J.J. (2014). The multi-commodity pickup-and-delivery traveling salesman problem. *Networks*, **63**, 46–59. 21, 22

Hernández-Pérez, H., Rodríguez-Martín, I. & Salazar-González, J.J. (2016). A hybrid heuristic approach for the multi-commodity pickup-and-delivery traveling salesman problem. *European Journal of Operational Research*, **251**, 44–52. 22

Hinojosa, Y., Puerto, J. & F.R., F. (2000). A multiperiod two-echelon multi-commodity capacitated plant location problem. *European Journal of Operational Research*, **123**, 271–291. 80

Holeczek, N. (2019). Hazardous materials truck transportation problems: A classification and state of the art literature review. *Transportation Research Part D: Transport and Environment*, **69**, 305–328. 33

Iakovou, E., Douligeris, C., Li, H., Ip, C. & Yudhbir, L. (1999). A maritime global route planning model for hazardous materials transportation. *Transportation science*, **33**, 34–48. 33

King, R.P., Hand, M.S. & Gómez, M.I. (2015). *Growing Local: Case Studies on Local Food Supply Chains*. U of Nebraska Press. 1, 79, 185

Lahyani, R., Coelho, L.C., Khemakhem, M., Laporte, G. & Semet, F. (2015). A multi-compartment vehicle routing problem arising in the collection of olive oil in Tunisia. *Omega*, **51**, 1–10. 20

Laporte, G. (2009). Fifty years of vehicle routing. *Transportation Science*, **43**, 408–416. 8

Li, C., Gong, L., Luo, Z. & Lim, A. (2019). A branch-and-price-and-cut algorithm for a pickup and delivery problem in retailing. *Omega*, **89**, 71 – 91. 22

Magnanti, T.L. & Wong, R.T. (1984). Network design and transportation planning: Models and algorithms. *Transportation Science*, **18**, 1–55. 86

Manerba, D., Mansini, R. & Riera-Ledesma, J. (2017). The traveling purchaser problem and its variants. *European Journal of Operational Research*, **259**, 1–18. 23

Masson, R., Lehuédé, F. & Péton, O. (2013). An adaptive large neighborhood search for the pickup and delivery problem with transfers. *Transportation Science*, **47**, 344–355. 46, 64

Miranda-De La Lama, G.C., Villarroel, M. & María, G.A. (2014). Livestock transport from the perspective of the pre-slaughter logistic chain: A review. *Meat Science*, **98**, 9–20. 19

Mirchandani, P.B. & Francis, R.L. (1990). *Discrete location theory*. 92

Mirzaei, S. & Wøhlk, S. (2019). A branch-and-price algorithm for two multi-compartment vehicle routing problems. *EURO Journal on Transportation and Logistics*, **8**, 1–33. 15

Mjirda, A., Jarboui, B., Macedo, R., Hanafi, S. & Mladenović, N. (2014). A two phase variable neighborhood search for the multi-product inventory routing problem. *Computers & Operations Research*, **52**, 291–299. 28

Moin, N.H., Salhi, S. & Aziz, N.A.B. (2011). An efficient hybrid genetic algorithm for the multi-product multi-period inventory routing problem. *International Journal of Production Economics*, **133**, 334–343. 28

Montoya-Torres, J.R., Franco, J.L., Isaza, S.N., Jiménez, H.F. & Herazo-Padilla, N. (2015). A literature review on the vehicle routing problem with multiple depots. *Computers & Industrial Engineering*, **79**, 115–129. 10, 82

Muyldermans, L. & Pang, G. (2010). On the benefits of co-collection: Experiments with a multi-compartment vehicle routing algorithm. *European Journal of Operational Research*, **206**, 93–103. 18

Nagy, G. & Salhi, S. (1998). The many-to-many location-routing problem. *Top*, **6**, 261–275. 30

Nagy, G., Wassan, N.A., Speranza, M.G. & Archetti, C. (2015). The Vehicle Routing Problem with Divisible Deliveries and Pickups. *Transportation Science*, **49**, 271–294. 74, 75, 76

Nakao, Y. & Nagamochi, H. (2007). A dp-based heuristic algorithm for the discrete split delivery vehicle routing problem. *Journal of Advanced Mechanical Design, Systems, and Manufacturing*, **1**, 217–226. 24, 25

Ogier, M., Cung, V.D. & Boissière, J. (2013). Service network design in short and local fresh food supply chain. *RAIRO-Operations Research*, **47**, 445–464. 81

Oppen, J. & Løkketangen, A. (2008). A tabu search approach for the livestock collection problem. *Computers & Operations Research*, **35**, 3213–3229. 19

Oppen, J., Løkketangen, A. & Desrosiers, J. (2010). Solving a rich vehicle routing and inventory problem using column generation. *Computers & Operations Research*, **37**, 1308–1317. 19

Ostermeier, M. & Hübner, A. (2018). Vehicle selection for a multi-compartment vehicle routing problem. *European Journal of Operational Research*, **269**, 682–694. 20

Özdamar, L., Ekinci, E. & Küçükyazici, B. (2004). Emergency logistics planning in natural disasters. *Annals of operations research*, **129**, 217–245. 35

Paredes-Belmar, G., Bronfman, A., Marianov, V. & Latorre-Núñez, G. (2017). Hazardous materials collection with multiple-product loading. *Journal of cleaner production*, **141**, 909–919. 34

Parragh, S.N., Doerner, K.F. & Hartl, R.F. (2008). A survey on pickup and delivery problems part ii: Transportation between pickup and delivery locations. *Journal für Betriebswirtschaft*, **58**, 81–117. 11

Pisinger, D. & Ropke, S. (2010). Large neighborhood search. In *Handbook of metaheuristics*, 399–419, Springer. 51, 127

Popović, D., Vidović, M. & Radivojević, G. (2012). Variable neighborhood search heuristic for the inventory routing problem in fuel delivery. *Expert Systems with Applications*, **39**, 13390–13398. 17, 27

Prins, C. (2004). A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research*, **31**, 1985–2002. 52, 96

Prins, C. (2009). A GRASP × evolutionary local search hybrid for the vehicle routing problem. *Bio-inspired algorithms for the vehicle routing problem*, 35–53. 55

Prodhon, C. & Prins, C. (2014). A survey of recent research on location-routing problems. *European Journal of Operational Research*, **238**, 1–17. 30

Rahmani, Y., Ramdane Cherif-Khettaf, W. & Oulamara, A. (2016). The two-echelon multi-products location-routing problem with pickup and delivery: formulation and heuristic approaches. *International Journal of Production Research*, **54**, 999–1019. 32

Ramkumar, N., Subramanian, P., Narendran, T.T. & Ganesh, K. (2012). Mixed integer linear programming model for multi-commodity multi-depot inventory routing problem. *OPSEARCH*, **49**, 413–429. 28, 29

Raviv, T., Tzur, M. & Forma, I.A. (2013). Static repositioning in a bike-sharing system: models and solution approaches. *EURO Journal on Transportation and Logistics*, **2**, 187–229. 22

Reed, M., Yiannakou, A. & Evering, R. (2014). An ant colony algorithm for the multi-compartment vehicle routing problem. *Applied Soft Computing*, **15**, 169–176. 18

Rieck, J., Ehrenberg, C. & Zimmermann, J. (2014). Many-to-many location-routing with inter-hub transport and multi-commodity pickup-and-delivery. *European Journal of Operational Research*, **236**, 863–878. 31

Riera-Ledesma, J. & Salazar-González, J.J. (2012). Solving school bus routing using the multiple vehicle traveling purchaser problem: A branch-and-cut approach. *Computers & Operations Research*, **39**, 391 – 404. 23

Ropke, S. & Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, **40**, 455–472. 46, 48, 56, 58, 60, 64, 76, 97, 133

Ross, G.T. & Soland, R.M. (1975). A branch and bound algorithm for the generalized assignment problem. *Mathematical Programming*, **8**, 91–103. 95

Rucabado-Palomar, T. & Cuéllar-Padilla, M. (2018). Short food supply chains for local food: a difficult path. *Renewable Agriculture and Food Systems*, 1–10. 1, 79, 185

Sadjady, H. & Davoudpour, H. (2012). Two-echelon, multi-commodity supply chain network design with mode selection, lead-times and inventory costs. *Computer & Operations Research*, **39**, 1345–1354. 80

Salani, M. & Vacca, I. (2011). Branch and price for the vehicle routing problem with discrete split deliveries and time windows. *European Journal of Operational Research*, **213**, 470 – 477. 25

Shaabani, H. & Kamalabadi, I.N. (2016). An efficient population-based simulated annealing algorithm for the multi-product multi-retailer perishable inventory routing problem. *Computers & Industrial Engineering*, **99**, 189–201. 28

Shaw, P. (1997). A new local search algorithm providing high quality solutions to vehicle routing problems. *APES Group, Dept of Computer Science, University of Strathclyde, Glasgow, Scotland, UK*. 46

Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. In *International Conference on Principles and Practice of Constraint Programming*, 417–431, Springer. 46

Solomon, M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, **35**, 254–265. 62, 104

Speranza, M.G. & Ukovich, W. (1994). Minimizing transportation and inventory costs for several products on a single link. *Operations Research*, **42**, 879–894. 26, 27

Speranza, M.G. & Ukovich, W. (1996). An algorithm for optimal shipments with given frequencies. *Naval Research Logistics*, **43**, 655–671. 26, 27

Surjandari, I., Rachman, A., Dianawati, F. & Wibowo, R.P. (2011). Petrol delivery assignment with multi-product, multi-depot, split deliveries and time windows. *International Journal of Modeling and Optimization*, **1**, 375. 17

Sze, J.F., Salhi, S. & Wassan, N. (2016). A hybridisation of adaptive variable neighbourhood search and large neighbourhood search: Application to the vehicle routing problem. *Expert Systems with Applications*, **65**, 383–397. 46

Sze, J.F., Salhi, S. & Wassan, N. (2017). The cumulative capacitated vehicle routing problem with min-sum and min-max objectives: An effective hybridisation of adaptive variable neighbourhood search and large neighbourhood search. *Transportation Research Part B: Methodological*, **101**, 162–184. 46

Toth, P. & Vigo, D. (2014). *Vehicle routing: problems, methods, and applications*. SIAM. 8, 42

Tzeng, G.H., Cheng, H.J. & Huang, T.D. (2007). Multi-objective optimal planning for designing relief delivery systems. *Transportation Research Part E: Logistics and Transportation Review*, **43**, 673–686. 35

Verter, V. & Kara, B.Y. (2008). A path-based approach for hazmat transport network design. *Management Science*, **54**, 29–40. 33

Vidović, M., Popović, D. & Ratković, B. (2014). Mixed integer and heuristics model for the inventory routing problem in fuel delivery. *International Journal of Production Economics*, **147**, 593–604. 17

Yi, W. & Kumar, A. (2007). Ant colony optimization for disaster relief operations. *Transportation Research Part E: Logistics and Transportation Review*, **43**, 660–672. 35

Yi, W. & Özdamar, L. (2007). A dynamic logistics coordination model for evacuation and support in disaster response activities. *European Journal of Operational Research*, **179**, 1177–1193. 35

Zhang, Z., Cheang, B., Li, C. & Lim, A. (2019). Multi-commodity demand fulfillment via simultaneous pickup and delivery for a fast fashion retailer. *Computers & Operations Research*, **103**, 81–96. 23

Appendix

A Detailed results on the benchmark instances for the C-SDVRP

Table A1, Table A2 and Table A3 report the detailed results for the small, medium and large instances respectively. We report values in bold whenever we improve (or optimize or equal to) the respective instances.

Appendix

Table A1: Detailed computational results for the small sized instances ($n = 15$).

instances										ALNS results			
	M	p	n_{cc}	CC_1	CC_2	CC_3	Δ	α	OPT	$Cost$	Δ_O	$t(s)$	nbR
C101	2	0.6	22	10	12		[1,100]	1.1	283.3404	283.3404	**0.00**	6.01	6
		0.6	22	10	12		[40,60]	1.1	480.4342	480.4342	**0.00**	5.07	11
		1	30	15	15		[1,100]	1.1	422.4965	422.4965	**0.00**	10.64	9
		1	30	15	15		[40,60]	1.1	685.1662	685.1662	**0.00**	5.19	15
		0.6	22	10	12		[1,100]	1.5	241.0386	241.0386	**0.00**	8.67	5
		0.6	22	10	12		[40,60]	1.5	348.2731	348.2731	**0.00**	5.63	7
		1	30	15	15		[1,100]	1.5	340.5474	340.5474	**0.00**	14.70	7
		1	30	15	15		[40,60]	1.5	490.2973	490.2973	**0.00**	8.27	10
		0.6	22	10	12		[1,100]	2	200.9092	200.9092	**0.00**	8.66	4
		0.6	22	10	12		[40,60]	2	239.6829	239.6829	**0.00**	5.80	5
		1	30	15	15		[1,100]	2	239.9424	239.9424	**0.00**	13.57	5
		1	30	15	15		[40,60]	2	357.5929	357.5929	**0.00**	9.14	7
		0.6	22	10	12		[1,100]	2.5	170.0453	170.0453	**0.00**	10.31	3
		0.6	22	10	12		[40,60]	2.5	207.8259	207.8259	**0.00**	7.74	4
		1	30	15	15		[1,100]	2.5	205.7830	205.7830	**0.00**	12.29	4
		1	30	15	15		[40,60]	2.5	302.1813	302.1813	**0.00**	11.26	6
R101	2	0.6	22	10	12		[1,100]	1.1	408.8971	408.8971	**0.00**	7.43	7
		0.6	22	10	12		[40,60]	1.1	565.3383	565.3383	**0.00**	4.12	11
		1	30	15	15		[1,100]	1.1	537.2029	537.2029	**0.00**	9.75	9
		1	30	15	15		[40,60]	1.1	679.0232	679.0232	**0.00**	5.26	15
		0.6	22	10	12		[1,100]	1.5	353.0119	353.0119	**0.00**	6.77	5
		0.6	22	10	12		[40,60]	1.5	455.9724	455.9724	**0.00**	6.88	8
		1	30	15	15		[1,100]	1.5	443.0829	443.0829	**0.00**	12.31	7
		1	30	15	15		[40,60]	1.5	558.9565	558.9565	**0.00**	8.03	10
		0.6	22	10	12		[1,100]	2	301.4316	301.4316	**0.00**	8.50	4
		0.6	22	10	12		[40,60]	2	379.2848	379.2848	**0.00**	7.69	6
		1	30	15	15		[1,100]	2	370.5561	370.5561	**0.00**	14.73	5
		1	30	15	15		[40,60]	2	444.1868	444.1868	**0.00**	9.27	8
		0.6	22	10	12		[1,100]	2.5	280.7646	280.7646	**0.00**	7.50	3
		0.6	22	10	12		[40,60]	2.5	342.6854	342.6854	**0.00**	7.27	5
		1	30	15	15		[1,100]	2.5	323.5761	323.5761	**0.00**	13.68	4
		1	30	15	15		[40,60]	2.5	401.6087	401.6087	**0.00**	13.03	6
C101	3	0.6	28	12	7	9	[1,100]	1.1	333.4709	333.4709	**0.00**	11.29	7
		0.6	28	12	7	9	[40,60]	1.1	440.2241	440.2241	**0.00**	11.11	9
		1	45	15	15	15	[1,100]	1.1	428.5164	435.7543	1.69	25.05	8
		1	45	15	15	15	[40,60]	1.1	638.0919	638.0919	**0.00**	20.69	13
		0.6	28	12	7	9	[1,100]	1.5	262.8069	262.8069	**0.00**	10.47	5
		0.6	28	12	7	9	[40,60]	1.5	306.6363	306.6363	**0.00**	9.86	6
		1	45	15	15	15	[1,100]	1.5	315.9600	315.96	**0.00**	25.38	6
		1	45	15	15	15	[40,60]	1.5	457.9430	457.943	**0.00**	16.15	9
		0.6	28	12	7	9	[1,100]	2	204.9380	204.938	**0.00**	10.83	4
		0.6	28	12	7	9	[40,60]	2	263.2896	263.2896	**0.00**	9.34	5
		1	45	15	15	15	[1,100]	2	265.0623	265.0623	**0.00**	23.37	5
		1	45	15	15	15	[40,60]	2	347.3580	347.358	**0.00**	22.65	7
		0.6	28	12	7	9	[1,100]	2.5	168.2958	168.2958	**0.00**	10.09	3
		0.6	28	12	7	9	[40,60]	2.5	202.9044	202.9044	**0.00**	10.25	4
		1	45	15	15	15	[1,100]	2.5	206.6970	206.697	**0.00**	20.36	4
		1	45	15	15	15	[40,60]	2.5	310.7978	310.7978	**0.00**	24.45	6
R101	3	0.6	28	12	7	9	[1,100]	1.1	401.7502	401.7502	**0.00**	10.06	7
		0.6	28	12	7	9	[40,60]	1.1	497.1385	497.1385	**0.00**	8.78	10
		1	45	15	15	15	[1,100]	1.1	491.0411	491.0411	**0.00**	22.81	9
		1	45	15	15	15	[40,60]	1.1	679.0232	679.0232	**0.00**	17.69	15
		0.6	28	12	7	9	[1,100]	1.5	347.3693	347.3693	**0.00**	9.94	5
		0.6	28	12	7	9	[40,60]	1.5	410.813	410.813	**0.00**	8.19	7
		1	45	15	15	15	[1,100]	1.5	409.2905	409.2905	**0.00**	22.31	6
		1	45	15	15	15	[40,60]	1.5	541.0336	541.0336	**0.00**	15.16	9
		0.6	28	12	7	9	[1,100]	2	303.1439	303.1439	**0.00**	10.47	4
		0.6	28	12	7	9	[40,60]	2	343.7159	343.7159	**0.00**	11.04	5
		1	45	15	15	15	[1,100]	2	345.8351	345.8351	**0.00**	22.07	5
		1	45	15	15	15	[40,60]	2	444.1868	444.1868	**0.00**	17.91	8
		0.6	28	12	7	9	[1,100]	2.5	278.2234	278.2234	**0.00**	8.02	3
		0.6	28	12	7	9	[40,60]	2.5	312.3100	312.31	**0.00**	10.69	4
		1	45	15	15	15	[1,100]	2.5	320.3490	320.349	**0.00**	23.38	4
		1	45	15	15	15	[40,60]	2.5	393.8357	393.8357	**0.00**	20.34	6

Table A2: Detailed computational results for the mid sized instances ($n \in \{20; 40; 60; 80\}$).

instances									ALNS results					
	n	p	n_{cc}	CC_1	CC_2	CC_3	id	OPT/BKS	$Cost$	$\Delta_{O/B}$	$t(s)$	$nbMPO$	$nbMPOimp$	nbR
C101	20	0.6	37	15	11	11	1	573.8617*	573.8617	**0.00**	31.67	2	0	6
			39	13	10	16	2	592.0651*	592.0651	**0.00**	29.17	0	0	7
			38	13	13	12	3	595.5289*	595.5289	**0.00**	36.29	4	0	7
			40	9	14	17	4	617.8838*	617.8838	**0.00**	40.64	3	0	8
			33	9	10	14	5	628.2804*	628.2804	**0.00**	29.14	4	0	8
		1	60	20	20	20	1	750.6251*	750.6251	**0.00**	82.73	7	0	9
			60	20	20	20	2	714.6453*	714.6453	**0.00**	78.86	9	1	9
			60	20	20	20	3	626.1553*	626.1553	**0.00**	67.16	6	0	9
			60	20	20	20	4	747.7008*	747.7008	**0.00**	84.32	9	0	10
			60	20	20	20	5	768.5189*	768.5189	**0.00**	72.62	4	0	10
R101	20	0.6	37	15	11	11	1	457.8598*	457.8598	**0.00**	38.69	7	0	6
			39	13	10	16	2	667.0131*	667.0131	**0.00**	38.14	2	0	7
			38	13	13	12	3	455.0534*	455.0534	**0.00**	34.87	3	1	7
			40	9	14	17	4	589.9082*	589.9082	**0.00**	37.92	6	0	8
			33	9	10	14	5	663.2159*	663.2159	**0.00**	24.73	2	0	8
		1	60	20	20	20	1	599.8380*	599.8380	**0.00**	70.50	7	0	9
			60	20	20	20	2	863.8829*	864.1552	0.03	94.33	15	0	9
			60	20	20	20	3	617.9120	617.9662	0.01	78.97	10	0	9
			60	20	20	20	4	712.0175*	712.0175	**0.00**	83.18	7	1	10
			60	20	20	20	5	794.4068*	794.4068	**0.00**	81.62	10	0	10
C101	40	0.6	72	19	24	29	1	844.5669	841.0159	**-0.42**	70.98	4	0	9
			77	29	23	25	2	1005.8069	1002.4435	**-0.33**	80.82	17	0	12
			78	24	26	28	3	879.2568*	879.2568	**0.00**	83.96	8	0	11
			81	28	28	25	4	921.0554	922.4288	0.15	74.44	5	0	12
			73	25	19	29	5	868.7426*	868.7426	**0.00**	80.50	6	1	11
		1	120	40	40	40	1	1330.0617	1304.7180	**-1.91**	155.89	16	2	18
			120	40	40	40	2	1357.7864	1357.7890	0.00	163.58	18	3	19
			120	40	40	40	3	1309.3540	1299.4327	**-0.76**	149.76	12	0	16
			120	40	40	40	4	1238.8599	1238.3923	**-0.04**	158.73	17	1	17
			120	40	40	40	5	1287.4121	1273.2181	**-1.10**	178.96	19	4	16
R101	40	0.6	72	19	24	29	1	761.7828	764.0135	0.29	94.07	17	0	9
			77	29	23	25	2	896.0147	896.8111	0.09	57.28	8	0	12
			78	24	26	28	3	851.0276*	851.0276	**0.00**	98.89	20	0	11
			81	28	28	25	4	973.9919	976.5911	0.27	78.62	6	0	12
			73	25	19	29	5	854.3462*	855.1124	0.09	74.66	11	0	11
		1	120	40	40	40	1	1246.4958	1242.8772	**-0.29**	157.76	21	2	18
			120	40	40	40	2	1244.9154	1238.4378	**-0.52**	148.03	19	2	19
			120	40	40	40	3	1056.1320*	1056.1320	**0.00**	131.46	15	2	16
			120	40	40	40	4	1255.4495	1245.6884	**-0.78**	160.65	24	1	17
			120	40	40	40	5	1101.1214	1101.1214	**0.00**	143.75	14	2	16

Table A2 (continued).

instances									ALNS results					
	n	p	n_{cc}	CC_1	CC_2	CC_3	id	OPT/BKS	$Cost$	$\Delta_{O/B}$	$t(s)$	$nbMPO$	$nbMPOimp$	nbR
C101	60	0.6	107	34	36	37	1	1241.1029	1228.7010	**-1.00**	182.54	20	0	15
			112	40	33	39	2	1331.7718	1331.7718	**0.00**	204.25	33	1	17
			110	39	34	37	3	1184.7776	1180.6147	**-0.35**	189.09	14	0	14
			113	41	34	38	4	1303.1604	1284.9398	**-1.40**	195.14	19	0	16
			108	32	30	46	5	1303.7277	1303.3246	**-0.03**	194.56	18	0	16
		1	180	60	60	60	1	2003.0431	1996.6060	**-0.32**	332.53	26	0	27
			180	60	60	60	2	1667.6461	1671.8163	0.25	402.08	19	0	24
			180	60	60	60	3	1816.1236	1795.5325	**-1.13**	445.02	43	3	23
			180	60	60	60	4	1906.5617	1909.0054	0.13	361.95	22	2	26
			180	60	60	60	5	1650.5799	1637.1273	**-0.82**	477.33	44	3	22
R101	60	0.6	107	34	36	37	1	1293.2293	1286.0364	**-0.56**	181.69	19	1	15
			112	40	33	39	2	1326.5733	1317.7203	**-0.67**	136.28	13	1	17
			110	39	34	37	3	1028.5158	1028.5158	**0.00**	180.90	18	1	14
			113	41	34	38	4	1235.5766	1225.3226	**-0.83**	218.23	29	1	17
			108	32	30	46	5	1149.5550	1149.5673	0.00	168.89	11	0	16
		1	180	60	60	60	1	2086.6870	2068.5355	**-0.87**	346.07	24	3	27
			180	60	60	60	2	1662.5753	1660.3220	**-0.14**	328.54	33	0	23
			180	60	60	60	3	1581.5715	1562.5867	**-1.20**	400.14	31	1	23
			180	60	60	60	4	1732.8742	1723.4548	**-0.54**	328.65	17	1	26
			180	60	60	60	5	1507.7615	1503.2634	**-0.30**	362.81	18	3	22
C101	80	0.6	142	44	51	47	1	1648.0955	1647.8619	**-0.01**	329.97	22	2	20
			157	50	55	52	2	1616.9601	1603.4580	**-0.84**	355.95	24	1	21
			148	50	47	51	3	1750.6889	1742.9882	**-0.44**	366.80	27	2	22
			158	47	51	60	4	1468.4500	1446.7739	**-1.48**	413.62	33	2	19
			147	45	42	60	5	1702.9460	1684.7620	**-1.07**	400.47	39	1	21
		1	240	80	80	80	1	2277.9557	2255.8933	**-0.97**	668.31	23	6	30
			240	80	80	80	2	2133.9879	2118.7056	**-0.72**	684.15	31	4	29
			240	80	80	80	3	2623.9684	2588.2988	**-1.36**	720.92	35	5	35
			240	80	80	80	4	2389.1831	2377.7662	**-0.48**	758.18	31	10	32
			240	80	80	80	5	2410.9786	2405.0859	**-0.24**	689.43	40	7	33
R101	80	0.6	142	44	51	47	1	1467.6251	1449.3824	**-1.24**	319.10	26	1	20
			157	50	55	52	2	1482.3639	1481.5891	**-0.05**	349.74	28	0	21
			148	50	47	51	3	1618.6780	1606.2737	**-0.77**	278.83	21	1	22
			158	47	51	60	4	1432.0982	1416.8707	**-1.06**	331.53	22	1	19
			147	45	42	60	5	1482.7288	1466.4647	**-1.10**	319.93	22	2	20
		1	240	80	80	80	1	2137.3158	2107.6678	**-1.39**	697.74	29	6	31
			240	80	80	80	2	1955.9811	1938.9429	**-0.87**	710.27	40	2	29
			240	80	80	80	3	2302.7431	2291.3231	**-0.50**	618.40	30	0	35
			240	80	80	80	4	2123.9847	2113.1821	**-0.51**	685.71	30	3	32
			240	80	80	80	5	2155.4925	2138.8196	**-0.77**	520.92	19	1	33

Table A3: Detailed computational results for the large sized instances ($n = 100$).

instances										ALNS results					
	M	p	n_{cc}	CC_1	CC_2	CC_3	Δ	α	BKS	$Cost$	Δ_B	$t(s)$	$nbMPO$	$nbMPOimp$	nbR
C101	2	0.6	134	56	78		[1,100]	1.1	2035.3013	2016.5051	**-0.92**	260.68	18	0	30
			140	60	80				2180.0203	2170.9690	**-0.42**	270.33	22	1	35
			135	62	73				2082.1981	2076.9958	**-0.25**	236.59	21	1	32
			140	68	72				2148.4188	2133.8412	**-0.68**	268.38	29	0	32
			133	55	78				2041.9553	2016.1057	**-1.27**	270.64	31	2	31
C101	2	0.6	134	56	78		[1,100]	1.5	1573.6610	1571.8390	**-0.12**	325.56	22	9	23
			140	60	80				1713.4453	1690.1594	**-1.36**	322.81	34	2	25
			135	62	73				1594.5043	1579.4423	**-0.94**	326.38	42	3	23
			140	68	72				1620.3658	1593.0295	**-1.69**	295.26	26	1	23
			133	55	78				1544.2592	1542.7069	**-0.10**	327.96	32	0	23
C101	2	0.6	134	56	78		[1,100]	2	1484.4280	1484.8897	0.03	443.90	35	0	17
			140	60	80				1601.5050	1594.3828	**-0.44**	390.17	25	3	19
			135	62	73				1571.8295	1551.0808	**-1.32**	358.75	25	1	17
			140	68	72				1539.4321	1532.8325	**-0.43**	409.17	32	1	18
			133	55	78				1545.9487	1537.2621	**-0.56**	345.92	25	1	17
C101	2	0.6	134	56	78		[1,100]	2.5	1293.4366	1290.4392	**-0.23**	509.62	34	3	14
			140	60	80				1386.2707	1378.6220	**-0.55**	407.99	25	1	15
			135	62	73				1323.7604	1318.8051	**-0.37**	429.45	20	0	14
			140	68	72				1337.2558	1319.5941	**-1.32**	475.72	36	0	14
			133	55	78				1316.6055	1307.1230	**-0.72**	408.30	26	3	14
C101	2	0.6	134	56	78		[40,60]	1.1	3717.5992	3686.3063	**-0.84**	212.18	18	1	60
			140	60	80				3943.6197	3923.7703	**-0.50**	211.63	13	0	65
			135	62	73				3755.8823	3720.0138	**-0.95**	247.43	17	0	60
			140	68	72				3840.6868	3815.5816	**-0.65**	271.71	37	0	63
			133	55	78				3673.3203	3645.8466	**-0.75**	230.77	34	0	60
C101	2	0.6	134	56	78		[40,60]	1.5	2685.9916	2670.5453	**-0.58**	259.21	22	0	42
			140	60	80				2824.0916	2780.6093	**-1.54**	269.90	31	0	44
			135	62	73				2633.8403	2606.9999	**-1.02**	278.21	35	0	41
			140	68	72				2738.3272	2703.7315	**-1.26**	306.65	35	0	43
			133	55	78				2628.9645	2606.6952	**-0.85**	249.35	23	0	42
C101	2	0.6	134	56	78		[40,60]	2	2371.4171	2364.7218	**-0.28**	331.61	32	0	30
			140	60	80				2509.9521	2489.1959	**-0.83**	358.79	26	0	32
			135	62	73				2408.7012	2401.3033	**-0.31**	343.90	24	0	30
			140	68	72				2443.4540	2437.5545	**-0.24**	353.23	30	0	31
			133	55	78				2464.7661	2440.5193	**-0.98**	339.58	30	0	31
C101	2	0.6	134	56	78		[40,60]	2.5	1968.1149	1963.9711	**-0.21**	368.04	12	0	24
			140	60	80				2031.8053	2027.4671	**-0.21**	367.94	22	0	25
			135	62	73				1954.6604	1950.7756	**-0.20**	406.74	31	0	23
			140	68	72				2002.5383	1987.3101	**-0.76**	343.06	30	1	24
			133	55	78				1973.9611	1978.6316	0.24	367.40	20	0	24

Table A3 (continued).

instances										ALNS results					
	M	p	n_{cc}	CC_1	CC_2	CC_3	Δ	α	BKS	$Cost$	Δ_B	$t(s)$	$nbMPO$	$nbMPOimp$	nbR
C101	2	1	200	100	100		[1,100]	1.1	3146.6597	3096.0220	**-1.61**	493.02	34	3	48
			200	100	100				3081.7941	3047.0941	**-1.13**	459.72	24	2	50
			200	100	100				3376.5593	3341.0044	**-1.05**	530.95	39	6	53
			200	100	100				3336.4515	3293.8945	**-1.28**	508.36	35	8	52
			200	100	100				3063.1397	3025.7796	**-1.22**	459.09	28	1	49
C101	2	1	200	100	100		[1,100]	1.5	2356.8893	2334.4763	**-0.95**	526.05	30	2	35
			200	100	100				2326.5291	2301.5070	**-1.08**	566.56	41	2	36
			200	100	100				2526.2982	2518.5285	**-0.31**	554.75	34	6	39
			200	100	100				2500.0207	2479.4547	**-0.82**	534.61	34	0	38
			200	100	100				2303.9564	2297.1414	**-0.30**	527.30	25	3	36
C101	2	1	200	100	100		[1,100]	2	2129.0342	2124.8932	**-0.19**	603.94	25	2	27
			200	100	100				2169.8922	2170.6119	0.03	629.32	35	4	27
			200	100	100				2256.4305	2245.4843	**-0.49**	617.32	31	2	29
			200	100	100				2269.9257	2262.6622	**-0.32**	554.71	19	1	29
			200	100	100				2160.1093	2156.8393	**-0.15**	523.47	19	0	27
C101	2	1	200	100	100		[1,100]	2.5	1760.6323	1753.7203	**-0.39**	693.48	36	3	21
			200	100	100				1836.8031	1811.6854	**-1.37**	756.15	33	6	22
			200	100	100				1882.7674	1871.8435	**-0.58**	637.95	29	4	23
			200	100	100				1910.0314	1895.2543	**-0.77**	716.49	32	3	23
			200	100	100				1810.3104	1808.1843	**-0.12**	738.02	48	5	21
C101	2	1	200	100	100		[40,60]	1.1	5603.0836	5536.0546	**-1.20**	433.77	19	0	93
			200	100	100				5547.3519	5504.4821	**-0.77**	454.40	26	1	95
			200	100	100				5749.2419	5679.6959	**-1.21**	424.29	17	0	96
			200	100	100				5711.3460	5645.4455	**-1.15**	463.91	18	0	95
			200	100	100				5535.6135	5479.9888	**-1.00**	469.36	23	0	92
C101	2	1	200	100	100		[40,60]	1.5	3913.4546	3849.0963	**-1.64**	507.87	29	1	62
			200	100	100				3909.5761	3867.0017	**-1.09**	482.07	28	2	63
			200	100	100				3976.2127	3959.2347	**-0.43**	502.91	27	1	64
			200	100	100				3942.3417	3898.8312	**-1.10**	583.69	47	0	63
			200	100	100				3867.9698	3839.8436	**-0.73**	598.34	44	1	62
C101	2	1	200	100	100		[40,60]	2	3343.1458	3328.8877	**-0.43**	613.39	30	0	45
			200	100	100				3446.6946	3391.5541	**-1.60**	609.40	27	2	45
			200	100	100				3402.9488	3402.1518	**-0.02**	593.52	25	1	47
			200	100	100				3459.2005	3446.1003	**-0.38**	604.41	29	1	47
			200	100	100				3400.1668	3384.4928	**-0.46**	628.07	30	1	45
C101	2	1	200	100	100		[40,60]	2.5	2734.4477	2717.2918	**-0.63**	680.71	33	2	35
			200	100	100				2788.0824	2776.1339	**-0.43**	596.97	22	0	36
			200	100	100				2780.4184	2761.2009	**-0.69**	616.17	23	3	36
			200	100	100				2815.0422	2786.8485	**-1.00**	589.10	25	1	36
			200	100	100				2767.8571	2751.4713	**-0.59**	680.74	33	0	35

Table A3 (continued).

instances									BKS	ALNS results					
	M	p	n_{cc}	CC_1	CC_2	CC_3	Δ	α		$Cost$	Δ_B	$t(s)$	$nbMPO$	$nbMPOimp$	nbR
C101	3	0.6	179	65	50	64	[1,100]	1.1	2233.2625	2199.3967	**-1.52**	447.71	38	2	34
			194	67	63	64			2406.5116	2357.4534	**-2.04**	509.65	32	3	36
			186	62	58	66			2499.3959	2487.8220	**-0.46**	505.21	45	1	37
			193	69	57	67			2266.5132	2283.0043	0.73	448.84	27	0	36
			190	61	56	73			2529.7072	2509.7369	**-0.79**	471.85	39	4	37
C101	3	0.6	179	65	50	64	[1,100]	1.5	1724.7339	1698.9351	**-1.50**	449.14	32	7	25
			194	67	63	64			1812.1988	1807.9754	**-0.23**	579.95	28	3	27
			186	62	58	66			1908.2163	1893.4346	**-0.77**	521.23	38	2	27
			193	69	57	67			1763.0602	1745.0116	**-1.02**	568.39	47	1	26
			190	61	56	73			1938.0044	1930.7389	**-0.37**	576.69	26	4	28
C101	3	0.6	179	65	50	64	[1,100]	2	1651.6968	1626.3891	**-1.53**	633.94	38	1	19
			194	67	63	64			1654.5853	1652.1369	**-0.15**	677.73	30	2	20
			186	62	58	66			1712.4417	1684.7036	**-1.62**	648.54	40	2	20
			193	69	57	67			1702.0681	1677.4419	**-1.45**	584.07	22	0	20
			190	61	56	73			1719.8367	1714.1925	**-0.33**	600.65	25	1	21
C101	3	0.6	179	65	50	64	[1,100]	2.5	1413.2271	1411.6091	**-0.11**	681.11	34	0	15
			194	67	63	64			1442.7602	1424.9209	**-1.24**	810.87	45	0	16
			186	62	58	66			1452.9146	1428.4270	**-1.69**	712.86	42	2	16
			193	69	57	67			1458.4966	1442.4131	**-1.10**	780.48	36	0	16
			190	61	56	73			1446.4444	1444.4045	**-0.14**	756.13	31	1	17
C101	3	0.6	179	65	50	64	[40,60]	1.1	3318.3873	3257.0030	**-1.85**	401.09	32	0	53
			194	67	63	64			3509.4788	3489.2080	**-0.58**	435.08	26	1	56
			186	62	58	66			3603.3698	3582.1768	**-0.59**	399.24	25	0	55
			193	69	57	67			3518.1872	3484.3878	**-0.96**	446.43	32	0	56
			190	61	56	73			3649.5231	3615.4109	**-0.93**	496.11	48	0	57
C101	3	0.6	179	65	50	64	[40,60]	1.5	2424.2712	2407.4264	**-0.69**	419.67	21	0	37
			194	67	63	64			2525.9423	2523.9967	**-0.08**	558.51	28	0	40
			186	62	58	66			2610.6052	2600.9273	**-0.37**	452.04	26	2	39
			193	69	57	67			2535.7810	2525.7262	**-0.40**	468.16	32	0	39
			190	61	56	73			2673.1778	2640.8231	**-1.21**	479.02	25	0	40
C101	3	0.6	179	65	50	64	[40,60]	2	2270.3526	2265.3978	**-0.22**	482.40	25	1	28
			194	67	63	64			2346.2918	2326.5017	**-0.84**	643.35	26	1	30
			186	62	58	66			2316.4770	2313.2239	**-0.14**	540.93	21	0	29
			193	69	57	67			2367.4327	2352.4767	**-0.63**	574.32	27	4	30
			190	61	56	73			2339.7488	2321.6905	**-0.77**	560.01	38	3	30
C101	3	0.6	179	65	50	64	[40,60]	2.5	1875.4922	1857.8147	**-0.94**	501.48	23	1	22
			194	67	63	64			1971.8409	1948.0161	**-1.21**	643.45	28	2	24
			186	62	58	66			1927.4449	1917.8823	**-0.50**	606.26	40	1	23
			193	69	57	67			1974.0424	1972.7209	**-0.07**	618.31	30	0	24
			190	61	56	73			1929.9052	1918.9720	**-0.57**	607.40	16	1	24

Table A3 (continued).

instances									BKS	ALNS results					
	M	p	n_{cc}	CC_1	CC_2	CC_3	Δ	α		$Cost$	Δ_B	$t(s)$	$nbMPO$	$nbMPOimp$	nbR
C101	3	1	300	100	100	100	[1,100]	1.1	3458.3101	3378.7279	**-2.30**	952.16	50	5	53
			300	100	100	100			3314.9537	3245.1446	**-2.11**	914.82	43	6	51
			300	100	100	100			3294.7994	3265.0504	**-0.90**	865.51	33	8	52
			300	100	100	100			3430.0465	3370.6534	**-1.73**	825.87	34	0	55
			300	100	100	100			3214.8478	3162.0426	**-1.64**	900.59	39	7	51
C101	3	1	300	100	100	100	[1,100]	1.5	2610.2355	2561.2706	**-1.88**	972.71	30	6	39
			300	100	100	100			2514.8470	2459.3592	**-2.21**	1024.42	35	3	38
			300	100	100	100			2532.1347	2476.9681	**-2.18**	1083.84	38	6	38
			300	100	100	100			2641.7530	2565.6834	**-2.88**	923.07	37	2	40
			300	100	100	100			2455.7585	2404.8723	**-2.07**	968.00	28	5	37
C101	3	1	300	100	100	100	[1,100]	2	2300.4077	2280.2733	**-0.88**	1211.44	42	5	29
			300	100	100	100			2192.9203	2193.6158	0.03	1153.59	43	3	28
			300	100	100	100			2260.6841	2253.2807	**-0.33**	1142.33	28	3	28
			300	100	100	100			2410.0967	2390.4263	**-0.82**	1099.96	28	5	30
			300	100	100	100			2247.5337	2231.8800	**-0.70**	1240.23	26	4	28
C101	3	1	300	100	100	100	[1,100]	2.5	1929.7279	1914.9262	**-0.77**	1348.23	17	4	23
			300	100	100	100			1871.0897	1856.8834	**-0.76**	1606.09	20	0	23
			300	100	100	100			1867.4470	1873.4338	0.32	1519.55	26	3	23
			300	100	100	100			2013.6222	1985.1207	**-1.42**	1396.51	41	0	24
			300	100	100	100			1910.9060	1875.5780	**-1.85**	1625.37	56	8	22
C101	3	1	300	100	100	100	[40,60]	1.1	5239.6555	5196.4247	**-0.83**	906.42	34	1	86
			300	100	100	100			5143.4560	5059.3108	**-1.64**	1016.59	51	4	84
			300	100	100	100			5200.9560	5103.2213	**-1.88**	1054.92	36	3	83
			300	100	100	100			5291.7632	5243.8349	**-0.91**	987.19	50	1	89
			300	100	100	100			5174.4761	5114.0431	**-1.17**	978.28	41	0	85
C101	3	1	300	100	100	100	[40,60]	1.5	3789.0183	3755.7431	**-0.88**	829.41	32	1	60
			300	100	100	100			3789.1122	3733.9771	**-1.46**	944.05	48	3	60
			300	100	100	100			3752.1660	3732.6484	**-0.52**	971.90	53	2	60
			300	100	100	100			3839.3184	3770.5935	**-1.79**	897.98	35	6	61
			300	100	100	100			3825.3398	3740.0157	**-2.23**	1020.43	52	4	60
C101	3	1	300	100	100	100	[40,60]	2	3343.5307	3331.3655	**-0.36**	1145.10	38	1	45
			300	100	100	100			3322.9033	3307.6401	**-0.46**	1081.25	25	1	44
			300	100	100	100			3318.3125	3297.9859	**-0.61**	1044.56	27	2	44
			300	100	100	100			3424.6490	3382.5992	**-1.23**	1254.62	39	0	45
			300	100	100	100			3379.4211	3330.9490	**-1.43**	1240.97	50	4	45
C101	3	1	300	100	100	100	[40,60]	2.5	2807.4471	2785.0606	**-0.80**	1190.80	39	3	36
			300	100	100	100			2746.6361	2735.8088	**-0.39**	1259.03	43	4	35
			300	100	100	100			2754.3082	2760.9908	0.24	1255.07	24	7	35
			300	100	100	100			2803.2445	2789.5742	**-0.49**	1251.01	56	3	36
			300	100	100	100			2767.2392	2746.4348	**-0.75**	1145.49	24	0	36

Table A3 (continued).

instances										ALNS results					
	M	p	n_{cc}	CC_1	CC_2	CC_3	Δ	α	BKS	$Cost$	Δ_B	$t(s)$	$nbMPO$	$nbMPOimp$	nbR
R101	2	0.6	134	56	78		[1,100]	1.1	1916.2385	1887.4950	**-1.50**	279.09	27	1	31
			140	60	80				2153.6485	2152.5710	**-0.05**	300.47	31	1	35
			135	62	73				1927.2419	1920.3454	**-0.36**	265.37	21	0	33
			140	68	72				1971.0569	1956.0302	**-0.76**	302.81	31	3	32
			133	55	78				1833.7028	1828.4456	**-0.29**	306.71	25	0	32
R101	2	0.6	134	56	78		[1,100]	1.5	1496.8719	1482.2547	**-0.98**	319.35	29	0	22
			140	60	80				1685.8891	1670.7013	**-0.90**	307.37	39	1	25
			135	62	73				1530.9401	1517.8384	**-0.86**	296.24	34	0	23
			140	68	72				1549.2746	1541.2523	**-0.52**	307.34	17	0	24
			133	55	78				1476.0508	1464.6001	**-0.78**	312.00	23	0	23
R101	2	0.6	134	56	78		[1,100]	2	1226.8612	1213.6448	**-1.08**	355.57	37	0	17
			140	60	80				1363.4234	1351.5177	**-0.87**	378.88	35	0	19
			135	62	73				1254.4877	1249.8766	**-0.37**	370.13	31	0	18
			140	68	72				1268.1761	1260.8517	**-0.58**	418.11	33	2	17
			133	55	78				1203.2001	1200.0221	**-0.26**	325.71	31	2	17
R101	2	0.6	134	56	78		[1,100]	2.5	1067.2709	1059.0853	**-0.77**	385.07	28	2	13
			140	60	80				1178.0644	1169.7717	**-0.70**	453.09	43	0	15
			135	62	73				1087.0068	1084.9342	**-0.19**	414.91	33	0	14
			140	68	72				1102.6288	1104.8424	0.20	409.88	28	4	14
			133	55	78				1050.7626	1061.7723	1.05	430.69	30	0	14
R101	2	0.6	134	56	78		[40,60]	1.1	3358.5111	3333.0578	**-0.76**	263.31	25	0	60
			140	60	80				3575.0576	3553.3423	**-0.61**	250.22	19	0	65
			135	62	73				3344.4207	3327.4991	**-0.51**	223.60	20	0	61
			140	68	72				3406.2463	3386.6904	**-0.57**	275.68	25	0	63
			133	55	78				3269.4765	3244.8496	**-0.75**	236.62	30	0	60
R101	2	0.6	134	56	78		[40,60]	1.5	2438.2268	2420.4671	**-0.73**	261.82	22	0	41
			140	60	80				2610.9667	2598.2538	**-0.49**	293.50	24	0	44
			135	62	73				2427.5931	2414.9036	**-0.52**	274.40	29	0	41
			140	68	72				2503.2952	2481.8594	**-0.86**	274.12	17	1	43
			133	55	78				2440.5711	2414.1367	**-1.08**	313.44	36	0	42
R101	2	0.6	134	56	78		[40,60]	2	1895.5429	1895.7308	0.01	326.88	26	0	30
			140	60	80				1995.3201	1992.3865	**-0.15**	330.00	22	0	32
			135	62	73				1896.6321	1881.4183	**-0.80**	326.29	22	0	30
			140	68	72				1915.4443	1906.1707	**-0.48**	353.72	34	0	31
			133	55	78				1852.4127	1861.0882	0.47	324.98	20	0	31
R101	2	0.6	134	56	78		[40,60]	2.5	1587.2595	1577.0403	**-0.64**	344.39	31	0	24
			140	60	80				1668.3025	1659.9154	**-0.50**	330.66	21	1	25
			135	62	73				1578.8911	1570.4920	**-0.53**	343.04	21	0	24
			140	68	72				1608.9971	1591.3215	**-1.10**	348.60	24	2	24
			133	55	78				1561.0713	1558.7095	**-0.15**	305.21	12	1	24

Table A3 (continued).

instances									ALNS results					
	M	p	n_{cc}	CC_1	CC_2	CC_3	Δ	α	BKS	$Cost$	Δ_B	$t(s)$	$nbMPO$	$nbMPOimp$	nbR
R101	2	1	200	100	100		[1,100]	1.1	2771.6431	2756.7887	**-0.54**	531.18	38	6	49
			200	100	100				2834.4928	2827.3512	**-0.25**	426.29	19	2	49
			200	100	100				3016.8985	2985.2441	**-1.05**	507.52	35	3	53
			200	100	100				3048.9983	3031.0145	**-0.59**	533.16	35	3	53
			200	100	100				2800.6961	2784.0475	**-0.59**	496.69	29	4	49
R101	2	1	200	100	100		[1,100]	1.5	2128.3373	2122.9233	**-0.25**	530.35	29	2	35
			200	100	100				2194.4059	2183.9167	**-0.48**	578.51	42	3	36
			200	100	100				2299.5101	2295.0886	**-0.19**	501.07	25	3	39
			200	100	100				2307.0220	2297.4593	**-0.41**	582.93	39	3	38
			200	100	100				2149.6086	2137.8987	**-0.54**	507.42	24	1	36
R101	2	1	200	100	100		[1,100]	2	1715.9710	1698.3738	**-1.03**	551.73	25	1	27
			200	100	100				1730.2647	1722.1094	**-0.47**	589.96	33	3	27
			200	100	100				1813.5162	1816.3983	0.16	547.78	31	4	29
			200	100	100				1827.3575	1832.7354	0.29	582.61	32	7	29
			200	100	100				1717.4555	1704.9316	**-0.73**	512.40	16	1	27
R101	2	1	200	100	100		[1,100]	2.5	1461.4911	1452.8434	**-0.59**	682.70	32	2	21
			200	100	100				1466.1825	1460.9983	**-0.35**	705.62	43	9	22
			200	100	100				1533.0524	1534.9745	0.13	633.60	33	4	23
			200	100	100				1545.9473	1541.2999	**-0.30**	702.58	34	4	23
			200	100	100				1431.3942	1432.8548	0.10	631.85	38	0	21
R101	2	1	200	100	100		[40,60]	1.1	4878.3554	4815.8420	**-1.28**	477.78	28	1	93
			200	100	100				4885.1949	4854.3103	**-0.63**	533.46	46	0	95
			200	100	100				4954.1386	4928.0506	**-0.53**	396.24	18	0	97
			200	100	100				4906.3430	4886.1829	**-0.41**	357.43	15	0	95
			200	100	100				4812.6715	4778.5328	**-0.71**	390.36	16	0	93
R101	2	1	200	100	100		[40,60]	1.5	3515.9354	3489.7450	**-0.74**	602.15	39	3	62
			200	100	100				3584.7753	3544.2844	**-1.13**	554.11	32	0	64
			200	100	100				3592.7577	3551.5178	**-1.15**	609.51	45	0	64
			200	100	100				3583.5791	3546.3662	**-1.04**	512.74	25	3	63
			200	100	100				3495.4743	3472.3544	**-0.66**	586.66	41	3	62
R101	2	1	200	100	100		[40,60]	2	2640.2882	2622.6570	**-0.67**	636.79	37	2	45
			200	100	100				2669.1453	2657.6846	**-0.43**	485.80	15	1	46
			200	100	100				2690.6244	2665.3118	**-0.94**	578.84	29	1	46
			200	100	100				2704.5457	2695.0898	**-0.35**	620.18	28	0	47
			200	100	100				2655.6363	2641.4628	**-0.53**	550.42	24	1	46
R101	2	1	200	100	100		[40,60]	2.5	2153.4354	2141.4426	**-0.56**	644.84	25	0	35
			200	100	100				2213.9592	2189.4736	**-1.11**	571.24	23	0	36
			200	100	100				2222.5353	2202.7464	**-0.89**	675.78	38	1	36
			200	100	100				2204.7443	2203.0546	**-0.08**	650.99	34	3	36
			200	100	100				2159.9779	2149.3260	**-0.49**	538.39	16	0	35

Table A3 (continued).

instances										ALNS results					
	M	p	n_{cc}	CC_1	CC_2	CC_3	Δ	α	BKS	$Cost$	Δ_B	$t(s)$	$nbMPO$	$nbMPOimp$	nbR
R101	3	0.6	179	65	50	64	[1,100]	1.1	2091.9773	2082.4706	**-0.45**	448.50	37	2	34
			194	67	63	64			2248.9343	2221.3245	**-1.23**	534.88	43	2	36
			186	62	58	66			2137.6592	2134.4081	**-0.15**	447.11	27	2	37
			193	69	57	67			2131.7891	2131.5677	**-0.01**	458.80	24	1	36
			190	61	56	73			2205.0653	2183.8628	**-0.96**	478.30	31	0	38
R101	3	0.6	179	65	50	64	[1,100]	1.5	1638.1808	1628.8003	**-0.57**	507.92	39	8	25
			194	67	63	64			1735.8727	1726.5884	**-0.53**	564.75	38	5	26
			186	62	58	66			1688.8930	1678.2108	**-0.63**	563.18	48	7	27
			193	69	57	67			1695.4234	1683.4501	**-0.71**	502.74	27	3	26
			190	61	56	73			1708.9017	1698.7518	**-0.59**	508.70	24	0	27
R101	3	0.6	179	65	50	64	[1,100]	2	1344.1715	1333.3018	**-0.81**	579.85	38	4	19
			194	67	63	64			1417.4800	1409.0886	**-0.59**	638.09	33	4	20
			186	62	58	66			1382.2839	1366.0804	**-1.17**	554.83	24	2	20
			193	69	57	67			1385.4181	1380.4631	**-0.36**	628.49	23	3	20
			190	61	56	73			1377.7779	1372.9908	**-0.35**	612.43	32	3	21
R101	3	0.6	179	65	50	64	[1,100]	2.5	1146.9121	1144.9686	**-0.17**	703.87	39	4	15
			194	67	63	64			1214.4350	1211.4598	**-0.24**	739.74	37	7	16
			186	62	58	66			1188.0089	1183.2168	**-0.40**	707.54	47	2	16
			193	69	57	67			1190.1513	1188.4567	**-0.14**	750.52	30	0	16
			190	61	56	73			1203.4458	1199.0058	**-0.37**	676.18	27	1	17
R101	3	0.6	179	65	50	64	[40,60]	1.1	3048.1305	3027.5465	**-0.68**	455.96	37	2	54
			194	67	63	64			3278.5334	3244.0645	**-1.05**	444.34	25	2	57
			186	62	58	66			3035.4818	3030.9153	**-0.15**	451.00	25	0	55
			193	69	57	67			3126.1265	3115.1466	**-0.35**	432.35	24	1	56
			190	61	56	73			3144.6563	3112.9713	**-1.01**	469.27	37	1	57
R101	3	0.6	179	65	50	64	[40,60]	1.5	2234.7360	2224.7829	**-0.45**	447.71	29	2	37
			194	67	63	64			2376.7317	2368.8940	**-0.33**	531.41	31	1	39
			186	62	58	66			2255.2386	2254.6679	**-0.03**	528.31	38	3	39
			193	69	57	67			2315.3242	2313.7430	**-0.07**	459.05	18	1	39
			190	61	56	73			2371.8751	2356.9995	**-0.63**	463.99	19	0	40
R101	3	0.6	179	65	50	64	[40,60]	2	1806.0300	1787.6283	**-1.02**	468.94	27	1	28
			194	67	63	64			1915.8784	1898.5402	**-0.90**	573.18	25	1	30
			186	62	58	66			1780.7996	1776.2130	**-0.26**	560.20	34	0	29
			193	69	57	67			1844.3127	1832.6458	**-0.63**	615.95	33	2	30
			190	61	56	73			1857.0664	1850.7579	**-0.34**	539.93	42	1	30
R101	3	0.6	179	65	50	64	[40,60]	2.5	1507.1235	1489.3193	**-1.18**	475.82	37	2	22
			194	67	63	64			1603.4522	1587.9639	**-0.97**	639.35	39	0	24
			186	62	58	66			1505.6898	1505.9298	0.02	489.36	26	3	23
			193	69	57	67			1559.1172	1555.8633	**-0.21**	651.61	35	2	24
			190	61	56	73			1563.5929	1554.1869	**-0.60**	616.64	47	4	24

Table A3 (continued).

instances									BKS	ALNS results					
	M	p	n_{cc}	CC_1	CC_2	CC_3	Δ	α		$Cost$	Δ_B	$t(s)$	$nbMPO$	$nbMPOimp$	nbR
R101	3	1	300	100	100	100	[1,100]	1.1	3071.7439	3041.7396	**-0.98**	893.51	45	4	53
			300	100	100	100			2946.0070	2934.6564	**-0.39**	837.63	29	6	52
			300	100	100	100			2985.6872	2964.3072	**-0.72**	934.43	41	7	52
			300	100	100	100			3105.7428	3079.8490	**-0.83**	813.04	31	5	55
			300	100	100	100			2852.7976	2858.3302	0.19	944.08	32	4	51
R101	3	1	300	100	100	100	[1,100]	1.5	2345.1176	2325.9298	**-0.82**	930.92	31	7	39
			300	100	100	100			2265.2202	2250.6250	**-0.64**	945.72	35	4	38
			300	100	100	100			2296.8218	2300.1563	0.15	1068.96	37	5	38
			300	100	100	100			2371.0709	2365.1241	**-0.25**	1020.65	37	3	40
			300	100	100	100			2190.7829	2185.2957	**-0.25**	956.33	20	5	37
R101	3	1	300	100	100	100	[1,100]	2	1859.7774	1847.8667	**-0.64**	1070.10	31	3	29
			300	100	100	100			1816.4798	1800.5916	**-0.87**	1023.86	34	1	28
			300	100	100	100			1825.7908	1813.6036	**-0.67**	1117.82	40	5	28
			300	100	100	100			1876.5282	1871.2735	**-0.28**	1052.49	26	3	30
			300	100	100	100			1765.6787	1763.3915	**-0.13**	1277.70	27	7	28
R101	3	1	300	100	100	100	[1,100]	2.5	1578.8174	1561.1608	**-1.12**	1208.35	36	4	23
			300	100	100	100			1539.1188	1522.9296	**-1.05**	1498.51	37	8	23
			300	100	100	100			1550.6387	1527.6379	**-1.48**	1474.97	33	3	23
			300	100	100	100			1597.1673	1584.9913	**-0.76**	1285.72	28	1	24
			300	100	100	100			1510.7153	1494.2903	**-1.09**	1395.11	27	6	22
R101	3	1	300	100	100	100	[40,60]	1.1	4607.7839	4605.8439	**-0.04**	887.44	25	1	87
			300	100	100	100			4540.1249	4510.2998	**-0.66**	820.83	23	1	85
			300	100	100	100			4560.4271	4509.0150	**-1.13**	848.02	25	0	85
			300	100	100	100			4678.0437	4643.9582	**-0.73**	821.14	23	1	89
			300	100	100	100			4541.4038	4518.3114	**-0.51**	900.92	29	1	85
R101	3	1	300	100	100	100	[40,60]	1.5	3411.5788	3379.7642	**-0.93**	1001.82	47	2	60
			300	100	100	100			3373.6096	3362.6067	**-0.33**	1095.28	56	3	60
			300	100	100	100			3409.7802	3385.7239	**-0.71**	1056.57	54	3	60
			300	100	100	100			3432.4292	3408.6962	**-0.69**	1021.90	43	5	61
			300	100	100	100			3433.4163	3374.5175	**-1.72**	1022.52	49	3	60
R101	3	1	300	100	100	100	[40,60]	2	2639.2852	2626.6402	**-0.48**	1149.13	35	3	45
			300	100	100	100			2609.9934	2604.937	**-0.19**	1033.69	20	1	44
			300	100	100	100			2600.4690	2582.7011	**-0.68**	1123.78	40	4	44
			300	100	100	100			2656.4737	2636.4516	**-0.75**	1344.94	41	4	46
			300	100	100	100			2620.8533	2627.2844	0.25	1138.19	25	2	45
R101	3	1	300	100	100	100	[40,60]	2.5	2195.1808	2196.4906	0.06	1226.16	41	2	36
			300	100	100	100			2186.6508	2170.1547	**-0.75**	1264.51	34	6	35
			300	100	100	100			2192.0684	2179.0868	**-0.59**	1255.90	46	5	35
			300	100	100	100			2241.4465	2220.6163	**-0.93**	1227.96	32	0	36
			300	100	100	100			2179.4536	2167.7968	**-0.53**	1157.51	34	5	35

B Algorithm and results

B.1 Algorithm for the full truck strategy

Algorithm 10 Full truck strategy.

for each supplier $s \in \mathcal{V}_S$ **do**
  for each commodity $m \in \mathcal{M}$ **do**
    Compute the remaining quantity $R_{sm} = O_{sm} - \sum_{\forall d \in \mathcal{V}_D} q_{sd}^m$
  end for
  Let nbR_s be the number of commodities of supplier s such that $R_{sm} > 0$
end for
for each distribution center $d \in \mathcal{V}_D$ **do**
  for each supplier $s \in \mathcal{V}_S$ **do**
    Compute the remaining capacity in the trucks $R_{sd} = x_{sd} \times Q_S - \sum_{m \in \mathcal{M}} q_{sd}^m$
    while $nbR_s > 0$ and $R_{sd} > 0$ **do**
      Compute the average remaining capacity per commodity as $avgR_{sd} = R_{sd}/nbR_s$
      for each commodity $m \in \mathcal{M}$ **do**
        Determine the maximum quantity of commodity m that supplier s can insert in the trucks sent to DC d as $Q_{sd}^m = min\{avgR_{sd}, R_{sm}\}$
        Add Q_{sd}^m to q_{sd}^m
        Remove Q_{sd}^m units from R_{sd}
        Remove Q_{sd}^m units from R_{sm}
      end for
      Update nbR_s
    end while
  end for
end for

B.2 Detailed results for setting the value of δ

We discuss here how we set the value of δ that determines the location of duplicated customers and distribution center. We solve the delivery subproblem (SPD) for each instance $\mathfrak{I}_{MC2DP}$ in the base set $\mathbb{S}$. The collection subproblem is not considered, and the SPD is solved as discussed in Section 4.4.2. The ALNS is run with a limit of 5000 iterations. To evaluate the solutions, we compare the cost obtained for the SPD ($SPDcost$) with twice the optimal value OPT for the corresponding instance of the C-SDVRP, as reported in Archetti *et al.* (2014). By construction of the instances, $2OPT$ is a valid upper bound for the SPD. These tests are performed with values of δ: 100, 50, 30, 20, 10, and 5.

The results are presented in Table B4. We present results for each group of instances, where a group is defined by a triplet (I, M, p). Results for a group are averaged on the values of α and Δ. Each instance contains 30 customers and 2 distribution centers. Table B4 reports the average number of *customer-commodities* in each group of instances ($avg.n_{cc}$), the number of instances in the group ($nbIns$), the average deviation ($avg.\Delta$) of the cost obtained by solving the SPD and the upper bound $2OPT$, the average CPU time in seconds ($avg.t(s)$). Moreover, the last three columns respectively report the number of solutions such that the cost obtained by solving the SPD is higher than, equal to or lower than $2OPT$.

We observe that when δ is large (i.e., equal to 100), the solution value equals $2OPT$ most of the time. In this case, by construction of the instances, this value is most likely to be the optimal value. On the other side, when δ is smaller than 100, the value $2OPT$ is a valid upper bound that can be improved by running the algorithm for the SPD. When δ equals 5 or 10, for all instances we obtain a value lower than $2OPT$ since the distribution centers are then very close. We are interested in generating a set of instances where the average deviation from $2OPT$ is negative, but with some instances such that the cost of the SPD equals $2OPT$. Hence, we set $\delta = 30$.

Table B4: Results for SPD with different values of δ in the instances.

Case 1: δ=100

$\mathcal{I}$	M	p	$avg.n_{cc}$	$nbIns$	$avg.\Delta$	$avg.t(s)$	$SPDcost > 2OPT$	$SPDcost = 2OPT$	$SPDcost < 2OPT$
C101	2	0.6	44	8	0.00	33.21	0	8	0
C101	2	1	60	8	0.01	45.55	1	7	0
C101	3	0.6	56	8	0.00	52.80	0	8	0
C101	3	1	90	8	0.20	82.53	4	4	0
R101	2	0.6	44	8	0.00	32.53	0	8	0
R101	2	1	60	8	0.00	52.62	0	8	0
R101	3	0.6	56	8	0.00	49.43	0	8	0
R101	3	1	90	8	0.00	95.92	0	8	0
		total		64	0.03	55.57	5	59	0

Case 2: δ=50

C101	2	0.6	44	8	0.00	32.70	0	8	0
C101	2	1	60	8	0.00	47.42	0	8	0
C101	3	0.6	56	8	0.00	48.98	0	8	0
C101	3	1	90	8	0.32	81.94	4	4	0
R101	2	0.6	44	8	-1.02	28.02	0	3	5
R101	2	1	60	8	-0.40	48.23	0	5	3
R101	3	0.6	56	8	-0.89	48.07	0	2	6
R101	3	1	90	8	-0.31	98.67	0	4	4
		total		64	-0.29	54.25	4	42	18

Case 3: δ=30

C101	2	0.6	44	8	-0.04	30.87	0	7	1
C101	2	1	60	8	-0.15	46.21	0	7	1
C101	3	0.6	56	8	-0.13	50.06	0	6	2
C101	3	1	90	8	0.23	84.65	3	4	1
R101	2	0.6	44	8	-6.67	30.99	0	0	8
R101	2	1	60	8	-7.67	52.84	0	0	8
R101	3	0.6	56	8	-6.43	48.26	0	0	8
R101	3	1	90	8	-7.59	90.94	0	0	8
		total		64	-3.56	54.35	3	24	37

Case 4: δ=20

C101	2	0.6	44	8	-4.32	30.41	0	0	8
C101	2	1	60	8	-3.44	41.52	0	0	8
C101	3	0.6	56	8	-3.80	43.25	0	0	8
C101	3	1	90	8	-2.76	80.34	0	0	8
R101	2	0.6	44	8	-14.94	28.73	0	0	8
R101	2	1	60	8	-14.98	40.78	0	0	8
R101	3	0.6	56	8	-14.90	43.97	0	0	8
R101	3	1	90	8	-14.10	88.89	0	0	8
		total		64	-9.15	49.74	0	0	64

Case 5: δ=10

C101	2	0.6	44	8	-20.81	33.21	0	0	8
C101	2	1	60	8	-17.86	44.37	0	0	8
C101	3	0.6	56	8	-18.68	55.91	0	0	8
C101	3	1	90	8	-16.96	96.62	0	0	8
R101	2	0.6	44	8	-21.56	30.85	0	0	8
R101	2	1	60	8	-18.42	42.43	0	0	8
R101	3	0.6	56	8	-21.55	44.18	0	0	8
R101	3	1	90	8	-18.02	77.60	0	0	8
		total		64	-19.23	53.15	0	0	64

Case 6: δ=5

C101	2	0.6	44	8	-15.40	30.55	0	0	8
C101	2	1	60	8	-14.01	44.46	0	0	8
C101	3	0.6	56	8	-14.35	53.59	0	0	8
C101	3	1	90	8	-13.00	96.31	0	0	8
R101	2	0.6	44	8	-20.46	31.24	0	0	8
R101	2	1	60	8	-15.97	41.87	0	0	8
R101	3	0.6	56	8	-20.54	47.92	0	0	8
R101	3	1	90	8	-16.92	74.95	0	0	8
		total		64	-16.33	52.61	0	0	64

Résumé Etendu

La production et la livraison de produits alimentaires frais ont considérablement évolué en Europe depuis les années 1950, notamment grâce à la modernisation des outils et des processus permettant de répondre à la demande des clients et de réduire les coûts de production. Les entreprises multinationales ont joué un rôle majeur d'intermédiaire entre les producteurs et les consommateurs (Rucabado-Palomar & Cuéllar-Padilla, 2018). De nos jours, l'un des problèmes majeurs que rencontrent les producteurs est leurs faibles revenus : au cours des dernières décennies, ils ont été encouragés à produire davantage, alors que leurs prix de vente unitaire étaient en baisse. Cependant, dans de nombreuses régions, il existe (1) des exploitations de taille moyenne où sont cultivés des produits variés de grande qualité (fraîcheur, peu de pesticides) et (2) des clients qui recherchent des produits de qualité et de une traçabilité (King *et al.*, 2015). Par conséquent, l'idée est apparue de connecter localement les producteurs et les clients (Berti & Mulligan, 2016), par le biais d'une chaîne logistique en circuits courts (et/ou) de proximité. Le principal objectif de ce type de chaîne logistique est d'améliorer la captation de valeur au bénéfice de la production.

Les circuits courts sont officiellement définis par le ministère français de l'Agriculture comme un mode de commercialisation des produits agricoles des producteurs aux consommateurs, soit en vente directe, soit en vente indirecte avec un seul intermédiaire entre les producteurs et les consommateurs. Les circuits de proximité peuvent impliquer plusieurs intermédiaires, mais tous les acteurs doivent être situés sur une zone limitée (par exemple, en tenant compte des restrictions géographiques ou politiques). La distance maximale entre les acteurs correspond généralement à environ 80 km (Blanquart *et al.*, 2010). Les chaînes logistiques classiques impliquent de nombreuses opérations de manipulation des produits et des temps de stockage et de transport importants. Il est alors crucial, dans ce type de chaîne logistique,

d'éviter le gaspillage alimentaire et les pertes économiques dues à la périssabilité des produits frais. En effet, selon un rapport de l'Organisation des Nations Unies pour l'alimentation et l'agriculture (Gustavsson *et al.*, 2012), environ un tiers des aliments destinés à la consommation humaine est perdu ou gaspillé dans le monde.s Lorsque l'on considère des circuits courts et de proximité, les acteurs sont situés dans une zone restreinte et il y a peu d'intermédiaires. Ceci fournit naturellement une meilleure garantie sur la traçabilité et la fraîcheur des produits. En effet, le délai entre la collecte et la livraison est généralement au maximum de 24 heures et les activités de manutention sont limitées car le nombre d'intermédiaires est limité.

Les circuits courts et de proximité impliquent peu d'intermédiaires. Les producteurs doivent donc prendre en charge une grande partie de la commercialisation et de la distribution de leurs produits, ce qui n'est pas leur activité principale. Cela est faisable lorsque les producteurs vendent directement leurs produits aux clients, car les volumes sont généralement faibles. Pour les ventes indirectes (via des cantines, restaurants ou supermarchés), les volumes sont plus importants. La chaîne logistique doit donc être conçue de manière appropriée pour organiser les flux de produits et minimiser les coûts de transport afin d'être compétitif par rapport aux chaînes logistiques traditionnelles.

Les circuits courts et de proximité s'organisent généralement autour d'un ensemble de plateformes de distribution. Comme le souligne Berti & Mulligan (2016), les plateformes logistiques sont les infrastructures les plus utilisées pour répondre à la demande croissante de produits locaux. Les producteurs approvisionnent généralement ces plateformes logistiques en effectuant des livraisons directes car les volumes sont importants. Les plateformes logistiques sont ensuite chargées de la consolidation et de la livraison des produits aux clients. En outre, on suppose généralement qu'un seul preneur de décision gère toutes les plateformes logistiques et coordonne la planification du transport pour les opérations de collecte et de livraison. Ce preneur de décision peut être une association de producteur ou une administration locale. Les plateformes logistiques sont considérées comme le seul intermédiaire de la chaîne logistique.

L'objectif de cette thèse est de concevoir et de mettre en œuvre des méthodes de résolution efficaces pour les problèmes de routage qui se posent dans la livraison de produits agricoles frais en circuits courts et locaux. L'une des principales caractéristiques de ces problèmes de routage est de prendre en compte explicitement

plusieurs produits car: (1) tous les producteurs ne commercialisent pas les mêmes produits, et (2) les produits peuvent être livrés aux clients par différents véhicules.

Dans cette thèse, notre objectif est d'étudier un problème complexe de distribution dans une chaîne logistique à deux échelons, dans laquelle trois groupes d'acteurs sont impliqués: les producteurs, les plateformes logistiques et les clients. Plusieurs produits sont collectés auprès des producteurs et livrés aux clients via des plateformes logistiques qui permettent de consolider les produits. Chaque producteur a une quantité disponible donnée pour chaque produit (éventuellement 0) et chaque client a une demande pour chaque produit (éventuellement 0). Les produits sont collectés auprès des producteurs et acheminés aux plateformes logistiques via des livraisons directes, puis distribués des plateformes logistiques aux clients par une flotte de véhicules effectuant des tournées de livraison. Les livraisons directes des producteurs aux clients ne sont pas considérées. Nous supposons que les produits sont compatibles, c'est-à-dire que chaque véhicule peut transporter n'importe quel sous-ensemble de produits tant que sa capacité est respectée. Plusieurs visites chez un client sont autorisées afin de réduire les coûts de transport. Cependant, pour le confort des clients, chaque produit demandé doit être livré en une seule fois.

Dans cette thèse, nous commençons par faire un état de l'art des problèmes de tournées de véhicules qui prennent explicitement en compte plusieurs produits à livrer. Classiquement, les problèmes de tournées de véhicules concernent implicitement plusieurs produits, par exemple en agrégeant les produits demandés en fonction du volume ou du poids pour former une demande unique ou en décomposant le problème pour chaque produit (par exemple s'ils sont livrés à partir de dépôts différents ou si les véhicules sont dédiés à un seul produit). Dans cette état de l'art, pour chaque problème, nous déterminons ce qui motive la prise en compte explicite de plusieurs produits, la manière dont les produits sont considérés dans les formulations et les méthodes de résolution, et quelles sont les applications principales.

Ensuite, nous abordons la livraison des clients à partir d'une seule plateforme logistique, en considérant plusieurs produits. Plus précisément, le problème étudié est appelé le *Commodity constrained Split Delivery Vehicle Routing Problem* (C-SDVRP). Ce problème apparait lorsque les clients ont besoin de plusieurs produits et acceptent qu'ils soient livrés séparément. Tous les produits peuvent être mélangés dans un même véhicule à condition que la capacité du véhicule soit satisfaite. Plusieurs visites chez un client sont autorisées, mais chaque produit doit être

livré en une seule fois. Nous proposons une heuristique basée sur un algorithme de type ALNS (*Adaptive Large Neighborhood Search*) pour résoudre le C-SDVRP, dans le but de traiter efficacement les instances de moyenne et grande taille. Nous prenons en compte les caractéristiques spécifiques du C-SDVRP et adaptons plusieurs mouvements de recherche locale afin d'améliorer une solution. De plus, un opérateur basé sur la programmation mathématique qui réaffecte les produits aux différentes routes est utilisé pour améliorer une nouvelle meilleure solution globale. Des tests ont été menés sur des instances de référence tirés de la littérature. Les résultats démontrent l'efficacité de l'algorithme, qui peut fournir un grand nombre de nouvelles solutions de meilleure qualité, dans des temps de calcul courts.

Pour résoudre le problème complet, à savoir le *Multi-Commodity two-echelon Distribution Problem* (MC2DP), nous proposons des approches de résolution séquentielles. Nous proposons une décomposition du MC2DP en deux sous-problèmes, à savoir un problème de collecte dans lequel les producteurs fournissent leurs produits aux plateformes logistiques qui effectuent des trajets allers-retours, et un problème de livraison dans lequel les plateformes logistiques livrent les clients en utilisant une flotte de véhicules homogènes qui effectuent des tournées. Le problème de collecte est modélisé sous la forme d'un programme linéaire à variables mixtes (MIP). Le problème de livraison est le cas multi-dépôts du C-SDVRP. Par conséquent, nous étendons l'algorithme ALNS développé pour le C-SDVRP au cas avec plusieurs dépôts pour les opérations de livraison. Ensuite, nous abordons les approches de résolution séquentielles pour résoudre le MC2DP dan son ensemble. Dans le MC2DP, les décisions de collecte (quelle quantité de chaque produit est livrée à quelle plateforme logistique) ont un impact sur les opérations de livraison. Ainsi, la collecte et la livraison doivent être déterminées conjointement. Nous présentons deux approches de résolution séquentielles basées sur la résolution, dans un ordre différent, des sous-problèmes de collecte et de livraison. Dans les deux cas, la solution du premier sous-problème détermine la quantité de chaque produit dans chaque plateforme logistique. Le deuxième sous-problème prend cette information en entrée. Nous proposons également différentes stratégies pour guider la solution du premier sous-problème afin de prendre en compte l'impact de sa solution sur le second sous-problème. Les approches séquentielles proposées sont évaluées et comparées à la fois sur des instances générées de manière aléatoire et sur une étude de cas liée aux circuits courts et de proximité pour la livraison de produits agricoles frais. Les

résultats montrent l'impact des caractéristiques du problème sur les stratégies de résolution séquentielle.

Les approches séquentielles proposées présentent l'avantage d'être faciles à concevoir et à comprendre par des décideurs qui ne sont experts. Cependant, une approche de résolution séquentielle peut fournir des solutions largement sous-optimales dans certains cas. Ainsi, nous continuons à développer une approche de résolution intégrée plus sophistiquée pour le MC2DP afin d'améliorer les résultats obtenus par les approches séquentielles. Sur la base des approches séquentielles, nous améliorons l'algorithme ALNS. Nous ajoutons de nouveaux opérateurs capables de modifier les opérations de collecte et de livraison en déplaçant certaines quantités de produit d'une plateforme logistique à une autre. Certains de ces opérateurs reposent sur une programmation mathématique afin de modifier la solution actuelle du sous-problème de collecte. Cette approche intégrée est évaluée et comparée aux meilleures solutions obtenues avec les approches séquentielles. Les résultats montrent que les opérateurs proposés sont tous nécessaires et qu'en moyenne, l'approche intégrée est capable de fournir de meilleurs résultats que toutes les approches séquentielles.

Cette thèse donne lieu à plusieurs axes de recherche futurs. Du point de vue méthodologique, la première perspective consiste à améliorer l'approche intégrée pour résoudre le MC2DP. L'idée principale est de développer un algorithme ALNS pour le problème dans son ensemble, c'est-à-dire pour les opérations de collecte et de livraison. Pour ce faire, nous proposons d'ajouter des opérateurs de destruction et de réparation au sous-problème de collecte, par exemple pour supprimer un trajet direct entre un producteur et une plateforme logistique. Par conséquent, à chaque itération de l'algorithme ALNS, les sous-problèmes de collecte et de livraison seraient détruits et réparés. Après la réparation, nous devons nous assurer que les solutions des sous-problèmes sont cohérentes, c'est-à-dire que chaque plateforme logistique collecte plus de produits que ce qu'elle livre. Une autre perspective pour améliorer l'approche intégrée pour résoudre le MC2DP consiste à développer une méthode de résolution rapide du sous-problème de collecte afin de pouvoir traiter des instances de grande taille telles que celles abordées dans l'étude de cas pour les approches séquentielles.

Une autre perspective de recherche de nos travaux consiste à adapter l'algorithme ALNS proposé pour le C-SDVRP afin de traiter d'autres variantes de problèmes de tournées de véhicules avec plusieurs produits et la possibilité de faire des livraisons

multiples. Les méthodes de résolution des tournées de véhicules avec plusieurs produits dupliquent généralement les clients pour chaque produit demandé, puis fonctionnent avec ces clients dupliqués dans les algorithmes de résolution. Comme nous l'avons fait pour C-SDVRP, nous pouvons proposer des algorithmes utilisant une double représentation de la solution en agrégeant ou non les clients dupliqués. Par exemple, nous pourrions essayer d'adapter l'algorithme ALNS proposé pour résoudre certains problèmes de tournées de véhicules avec des véhicules possédant plusieurs compartiments.

De plus, nous pourrions adapter la méthode proposée à certaines extensions du problème. Premièrement, dans cette thèse, nous ne considérons pas la capacité des plateformes logistiques. D'un point de vue pratique, nous pourrions inclure des contraintes de capacité sur les plateformes logistiques. Deuxièmement, dans cette thèse, nous avons considéré que les produits sont collectés auprès des producteurs et acheminés aux plateformes logistiques par des trajets directs. Nous avons considéré que les producteurs fournissent des quantités importantes et livrent leurs produits aux centres de distribution en effectuant un ou plusieurs allers-retours. Cependant, des recherches futures pourraient envisager un plus grand nombre de producteurs avec des quantités d'approvisionnement moins importantes. Il serait alors intéressant de réaliser des tournées de véhicules afin de collecter les produits auprès des producteurs. Pour résoudre cette version du problème, nous pourrions conserver le même cadre de résolution, mais la résolution du sous-problème de collecte à l'optimalité serait plus difficile.

Enfin, du point de vue de l'application, nous pourrions étudier le problème où il est autorisé que certains producteurs livrent directement quelques clients sans passer par les plateformes logistiques. En effet, dans le contexte de la livraison de produits agricoles frais en circuits courts et de proximité en produits frais, il est intéressant que les producteurs aient un contact direct avec des clients situés à proximité de leur ferme. Par conséquent, nous pourrions envisager un système logistique associant livraisons directes et livraisons via des plateformes logistiques.

Titre: Problèmes de tournées de véhicules avec plusieurs produits et applications à la livraison de produits frais en circuits courts et locaux

Résumé: Dans cette thèse, nous étudions les problèmes de tournées de véhicules pour la livraison de plusieurs produits, avec des applications dans les chaînes logistiques en circuit court et local. La chaîne logistique étudiée comprend deux échelons, et est composée de trois groupes d'acteurs: les fournisseurs, les plateformes de distribution et les clients. Les fournisseurs sont des agriculteurs qui produisent des produits alimentaires frais. Les plateformes de distribution sont chargées de la consolidation et de la livraison des produits aux clients. Les plateformes de distribution collectent les produits auprès des fournisseurs en faisant un ou plusieurs aller-retours. Les produits sont livrés aux clients avec une flotte de véhicules effectuant des tournées. Chaque client demande plusieurs produits et chaque agriculteur produit une quantité limitée de ces produits. Pour minimiser les coûts de transport, il est avantageux qu'un même client soit livré par plusieurs véhicules. Mais pour le confort du client, il est imposé qu'un produit soit livré en une seule fois par un seul véhicule. En conséquence, les différents produits sont explicitement pris en compte dans les modèles et méthodes de résolution. Le problème complet est nommé *Multi-Commodity two-echelon Distribution Problem* (MC2DP). Le problème restreint qui concerne uniquement la livraison à partir d'une seule plateforme de distribution est nommé *Commodity constrained Split Delivery Vehicle Routing Problem* (C-SDVRP). Nous proposons d'abord une heuristique basée sur un ALNS (*Adaptive Large Neighborhood Search*) pour résoudre le C-SDVRP. Nous abordons ensuite le problème complet (MC2DP) avec des opérations de collecte et de livraison et plusieurs plateformes de distribution. Afin de résoudre ce problème complexe, nous proposons de décomposer le problème: la collecte et la livraison sont résolues de manière séquentielle. De plus, nous développons une approche intégrée pour le MC2DP afin d'améliorer les solutions obtenues par l'approche de décomposition.

Mots-clés: problèmes de tournées de véhicules, livraison de plusieurs produits, recherche à voisinage large adaptatif, recherche locale, livraison de produits frais en circuits courts et locaux.

Title: Multiple commodities routing problems with applications in the local fresh food supply chain

Abstract: In this thesis, we study vehicle routing problems considering multiple commodities, with applications in the local fresh food supply chains. The studied supply chain contains two echelons with three sets of actors: suppliers, distribution centers and customers. Suppliers are farmers that produce some fresh foods. Distribution centers are in charge of consolidation and delivery of the products to customers. Distribution centers collect products from the suppliers that perform direct trips. Products are delivered to the customers with a fleet of vehicles performing routes. Each customer requires several commodities, and the farmers produce a limited quantity of these commodities. For the minimization of the transportation cost, it is beneficial that a single customer is delivered by several vehicles. However, for the convenience of the customer, it is imposed that a single commodity is delivered at once by a single vehicle. Hence, different commodities have been explicitly considered. The complete problem is named Multi-Commodity two-echelon Distribution Problem (MC2DP). The restricted problem that addresses only the delivery from a single distribution center is named Commodity constrained Split Delivery Vehicle Routing Problem (C-SDVRP). We first propose a heuristic based on the Adaptive Large Neighborhood Search (ALNS) for the C-SDVRP. Then, we address the whole problem (MC2DP) with collection and delivery operations and multiple distribution centers. In order to tackle this complex problem, we propose to decompose the problem: collection and delivery are sequentially solved. Furthermore, we develop an integrated approach for the MC2DP to improve the solutions obtained by the sequential approach.

Keywords: vehicle routing problem, multiple commodities, adaptive large neighborhood search, local search, local fresh food supply chain.