Université de Montréal

Tactical Vehicle Routing Planning with Application to Milk Collection and Distribution

par
Iman Dayarian
Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Thèse présentée à la Faculté des arts et des sciences
en vue de l'obtention du grade de
Philosophiæ Doctor (Ph.D.)
en informatique option recherche opérationnelle

Decembre 2013

Université de Montréal

Faculté des arts et des sciences

Cette thèse intitulée:

Tactical Vehicle Routing Planning with Application to Milk Collection and Distribution

présenté par:

Iman Dayarian

a été évalué par un jury composé des personnes suivantes:

Jacques Ferland
(président-rapporteur)

Michel Gendreau
(directeur de recherche)

Teodor Gabriel Crainic, Walter Rei
(co-directeur)

Louis-Martin Rousseau
(membre du jury)

Karen Smilowitz
(examinateur externe)

Thèse acceptée le:
16/12/2013

Sommaire

De nombreux problèmes pratiques qui se posent dans dans le domaine de la logistique, peuvent être modélisés comme des problèmes de tournées de véhicules. De façon générale, cette famille de problèmes implique la conception de routes, débutant et se terminant à un dépôt, qui sont utilisées pour distribuer des biens à un nombre de clients géographiquement dispersé dans un contexte où les coûts associés aux routes sont minimisés. Selon le type de problème, un ou plusieurs dépôts peuvent-être présents. Les problèmes de tournées de véhicules sont parmi les problèmes combinatoires les plus difficiles à résoudre.

Dans cette thèse, nous étudions un problème d'optimisation combinatoire, appartenant aux classes des problèmes de tournées de véhicules, qui est liée au contexte des réseaux de transport. Nous introduisons un nouveau problème qui est principalement inspiré des activités de collecte de lait des fermes de production, et de la redistribution du produit collecté aux usines de transformation, pour la province de Québec. Deux variantes de ce problème sont considerées. La première, vise la conception d'un plan tactique de routage pour le problème de la collecte-redistribution de lait sur un horizon donné, en supposant que le niveau de la production au cours de l'horizon est fixé. La deuxième variante, vise à fournir un plan plus précis en tenant compte de la variation potentielle de niveau de production pouvant survenir au cours de l'horizon considéré.

Dans la première partie de cette thèse, nous décrivons un algorithme exact pour la première variante du problème qui se caractérise par la présence de fenêtres de temps, plusieurs dépôts, et une flotte hétérogène de véhicules, et dont l'objectif est de minimiser le coût de routage. À cette fin, le problème est modélisé comme un problème multi-attributs de tournées de véhicules. L'algorithme exact est basé sur la génération de colonnes impliquant un algorithme de plus court chemin élémentaire avec contraintes de ressources.

Dans la deuxième partie, nous concevons un algorithme exact pour résoudre la deuxième variante du problème. À cette fin, le problème est modélisé comme un problème de tournées de véhicules multi-périodes prenant en compte explicitement les variations potentielles du niveau de production sur un horizon donné. De nouvelles stratégies sont proposées pour résoudre le problème de plus court chemin élémentaire avec contraintes de ressources, impliquant dans ce cas une structure particulière étant donné la caractéristique multi-périodes du problème général.

Pour résoudre des instances de taille réaliste dans des temps de calcul raisonnables, une approche de résolution de nature heuristique est requise. La troisième partie propose un algorithme de recherche adaptative à grands voisinages où de nombreuses nouvelles stratégies d'exploration et d'exploitation sont proposées pour améliorer la performances de l'algorithme proposé en termes de la qualité de la solution obtenue et du temps de calcul nécessaire.

Mots-clés. Problème de tournées de véhicules, problème de tournées de véhicules multi-périodes, génération de colonnes, chemin la plus court élémentaire avec contraintes de ressources, recherche adaptative à grands voisinages, algorithmes exacts, algorithmes heuristiques.

Summary

Many practical problems arising in real-world applications in the field of logistics can be modeled as vehicle routing problems (VRP). In broad terms, VRPs deal with designing optimal routes for delivering goods or services to a number of geographically scattered customers in a context in which, routing costs are minimized. Depending on the type of problem, one or several depots may be present. Routing problems are among the most difficult combinatorial optimization problems.

In this dissertation we study a special combinatorial optimization problem, belonging to the class of the vehicle routing problem that is strongly linked to the context of the transportation networks. We introduce a new problem setting, which is mainly inspired by the activities of collecting milk from production farms and distributing the collected product to processing plants in Quebec. Two different variants of this problem setting are considered. The first variant seeks a tactical routing plan for the milk collection-distribution problem over a given planning horizon assuming that the production level over the considered horizon is fixed. The second variant aims to provide a more accurate plan by taking into account potential variations in terms of production level, which may occur during the course of a horizon. This thesis is cast into three main parts, as follows:

In the first part, we describe an exact algorithm for the first variant of the problem, which is characterized by the presence of time windows, multiple depots, and a heterogeneous fleet of vehicles, where the objective is to minimize the routing cost.

To this end, the problem is modeled as a multi-attribute vehicle routing problem. The exact algorithm proposed is based on the column generation approach, coupled with an elementary shortest path algorithm with resource constraints.

In the second part, we design an exact framework to address the second variant of the problem. To this end, the problem is modeled as a multi-period vehicle routing problem, which explicitly takes into account potential production level variations over a horizon. New strategies are proposed to tackle the particular structure of the multi-period elementary shortest path algorithm with resource constraints.

To solve realistic instances of the second variant of the problem in reasonable computation times, a heuristic approach is required. In the third part of this thesis, we propose an adaptive large neighbourhood search, where various new exploration and

exploitation strategies are proposed to improve the performance of the algorithm in terms of solution quality and computational efficiency.

Keywords. Vehicle routing problem, multi-period vehicle routing problem, column generation, elementary shortest path with resource constraints, adaptive large neighborhood search, exact algorithm, heuristic algorithm.

To my parents and my wife.

به یاد روزهای خوب

Contents

Liste des figures

List of Tables

Acknowledgement

Carrying out the requisite work and then writing this thesis was, undoubtedly, the most arduous task I have undertaken. However, one of the joys of having completed the thesis is looking back at everyone who has helped me over the past six years.

I would like to begin by thanking my three supervisors: Professors Michel Gendreau, Teodor Gabriel Crainic, and Walter Rei. It is an often-used cliché, but in my case it is no overstatement to say that without the consistent guidance, tutelage, support, unparalleled knowledge, and encouragement of my three supervisors, this thesis would never have existed.

Thank you also to Serge Bisaillon for his patience and intelligence, which helped me getting through the implementation of my algorithms.

I would also like to thank Dr. Alireza Rahimi-Vahed for his valuable accompany during the last six years, which allowed me to learn many things. I also thank Dr. Mehdi Towhidi for his help in implementation tasks. Thank you also to all the graduate students at CIRRELT, primarily for giving me someone to moan at, when work was not progressing according to plan.

To my family, particularly my parents and sister, thank you for your love, support, and unwavering belief in me. Without you, I would not be the person I am today.

Above all, I would like to thank my wife, Leila, for her love and constant support, for all the late nights and early mornings, and for keeping me sane over the past few months. But most of all, thank you for being my best friend. I owe you a lot.

Chapter 1

Introduction

Combinatorial optimization is an important field in the area of computer science and operations research. Many real-life activities, as well as industrial applications, can be modeled as combinatorial optimization problems. The vehicle routing problem (VRP), initially introduced by Dantzig and Ramser [42], is an important key to efficient logistics system management; in fact, it is one of the most important combinatorial optimization in the field. Logistics incorporates activities which attempt to govern physical flows of raw material, semi-finished and finished products, as well as to assess the informational flow within an organization in order to provide resources relevant to needs, economic conditions and the necessary quality of service within organizations. In other words, logistics represents all activities that allow the right amounts of products to be provided at a lower cost when and where a demand exists.

The classic VRP concerns itself with fashioning a set of collection or delivery routes performed by a homogeneous fleet of vehicles, departing from one depot in order to satisfy customer demands so that some objective functions are optimized [151]. Often, the objective function of the VRP consists of minimizing the vehicles' total traveled distance, but it is also common to minimize other objectives such as the sum of fixed and variable costs or the number of used vehicles.

This dissertation deals with tactical planning for a class of routing problems inspired by some real-world logistics activities. Planning is the process of organizing the activities required to achieve a desired goal with respect to the available means. Planning a distribution system involves devising routes to satisfy different clients' requests, so as to optimize an objective function. An effective tactical plan requires a comprehensive knowledge of restrictions and requirements for different periods of the considered planning horizon. This is based on a global vision of development practices during the period in question. This view is often obtained through information from historical data or predictive models.

Vehicle routing problems have been the subject of considerable research, as shown by the existence of extensive scientific literature on the subject. The VRP generalizes

the well-known traveling salesman problem (TSP), but is much more difficult to solve in practice [102]. Over the past few decades, researchers in the field have introduced numerous extensions to the classic VRP, often to address practical problem settings encountered in real-world transportation activities. As Toth and Vigo [151] claimed, the use of appropriate solution methodologies to efficiently solve vehicle routing problems in distribution processes may result in savings, ranging from 5% to 20% in transportation costs. An appropriate solution methodology relies on simultaneously representing the reality through the mathematical models while proposing concrete solution approaches, both in terms of solution quality and computational effort.

Similar to many other combinatorial optimizations, further to general increases in the computational power and also significant advances, both in exact and heuristic methods, the tools and knowledge to find good solutions to practical variants of the VRP in a reasonable time has tremendously increased. However, these studies have still been criticized for being too focused on non-realistic models based on simplified assumptions. Therefore, many recent studies have attempted to simultaneously incorporate several complicating attributes, defining the feasibility structure and optimality criteria of the considered problems, which in turn translate to an increase in the complexity in terms of modeling and solution methodology. In the combinatorial optimization literature, such problems are referred to as "multi-attribute problems." The multi-attribute vehicle routing problems (MAVRPs) [79] are often concerned with a subset of the following attributes or characteristics: capacity and travel time constraints, time window restrictions, heterogeneous vehicle fleets with different travel costs, order-vehicle compatibility constraints, orders with multiple pickups and deliveries, different start and end locations for vehicles, and route restrictions for vehicles.

While many theoretical and methodological advances have been made in the classic VRPs (such as the capacitated vehicle routing problem (CVRP) or the vehicle routing problem with time windows (VRPTW)), more recently, a growing tendency to address real-life routing problems, often taking the form of MAVRPs, has been observed among researchers.

This dissertation consists of three papers, concerning the development of mathematical models as well as methodological solution approaches addressing different variants of a real-life MAVRP, inspired by the milk collection and distribution activities in Quebec.

1.1 Problem statement

This dissertation addresses two variants of a combinatorial optimization problem encountered in the context of milk collection in Quebec. For the sake of simplicity, throughout this dissertation, this problem is referred to as the dairy transportation problem (DTP). We begin this section by providing some facts regarding the dairy industry in Quebec. About 49% of Canada's dairy farms are located in Quebec, while about 50% of Canada's dairy industries are located in Quebec. Furthermore, the dairy

industry ranked third in the Canadian agricultural sector, in terms of value in 2010 [1].

The Fédération des producteurs de lait du Québec (FPLQ) is responsible for negotiating on behalf of the dairy farmers annual transportation contracts with the carriers. A negotiation takes place through the FPLQ to discuss and determine the cost and contracts before formalizing the agreement [2, 101]. Each contract with a carrier involves multiples routes, each of which consists of an origin (the vehicle depot), a collection sequence from producers' farms, and a destination (the processing plant).

The approach currently used by to construct the milk routes is mainly based on historical data. Based on the above description, there are three types of stakeholders, which are described below:

- The producers, which periodically produce a limited quantity of one or more products.
- The plants, which periodically receive the products. They transform these raw materials into consumable goods.
- The carriers, which collect the products from the producers and deliver them to the plants. Each carrier has one or more depots, where the vehicles are located.

The individual vehicles usually have different capacities, fixed costs, and variable costs. The fixed costs are the expenses that are not related to the distance traveled and should be paid when the vehicle is used while the variable costs depend on the distance traveled. Each plant has an associated demand window indicating the minimum and maximum quantities that can be delivered.

A route is a path that starts and ends at a depot and visits producers and plants; it may contain one or more pick-up and delivery phases. A route is feasible if the pick-ups do not exceed the vehicle capacity and the associated time windows are respected. The cost of a route is the sum of the costs of the arcs on the path plus the sum of the vehicle's fixed and variable costs. We assume throughout this paper that the triangle inequality holds for the costs and travel times. Also, the service times are considered to be independent of the quantities collected or delivered.

A vehicle can perform one or more circuits per day. We define three route types as follows:

Simple route: Each vehicle visits several producers and collects their products. It then delivers its entire load to one plant and returns to its depot.

Multi-drop route: A vehicle delivers its load to more than one plant before returning to its depot.

Interlaced multi-drop routes: Vehicles perform several circuits per day. A vehicle may visit other producers after completing its first visit to a plant. One or more plants are visited.

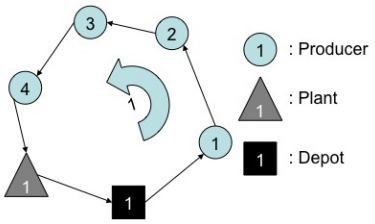

Figure 1.1: General configuration of simple route

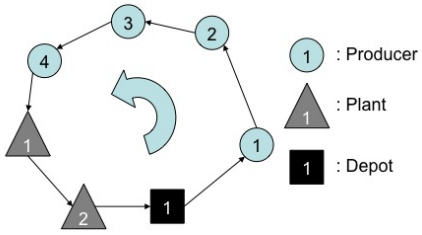

Figure 1.2: General configuration of multi-drop route

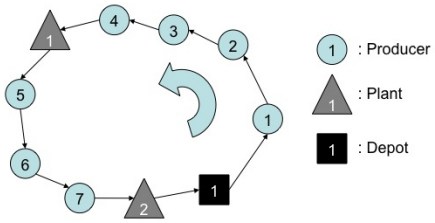

Figure 1.3: General configuration of interlaced multi-drop route

In this dissertation, we assume that all the routes are under the simple route pattern.

1.1.1 Single-period variant of the DTP

The first problem considered in this dissertation consists of a MAVRP variant of the problem described above. The attributes and characteristics of this problem setting are inspired by the DTP. In the considered problem, a daily plan is designed for a heterogeneous fleet of vehicles that depart from different depots and must visit a set of producers for collection operations. The collected products are delivered to a set of processing plants before the vehicles return to their original depots. In this multi-attribute vehicle routing problem, two kinds of constraints are considered: 1) each vehicle has a limited capacity of collecting products, and 2) there is time window associated with each producer, during which the collection operation must begin. The daily production of each producer is assumed to be known a priori, and fixed over the horizon. Moreover, each plant has a predetermined demand, which must be fulfilled by receiving a sufficient quantity of product per day. The cost of each vehicle route is computed through a system of fees depending on the distance traveled. Two variants of this problem are considered. In the first variant, which corresponds to the current state

of activities performed in Quebec, producers are preassigned to depots. The main goal of this variant is to design a set of least cost routes so that each producer is served via a route departing from its assigned depot. Moreover, one has to make sure that the daily plants' demands are covered by the delivered quantity of product. The second variant of the problem, allowing the revision of the producer-depot assignments, corresponds to a setting where all the preassignments are removed.

1.1.2 Multi-period variant of the DTP

The second problem considered in this dissertation is a new variant of the DTP. The main goal of this problem is to design a unique routing plan, which will be used as the basis of cost negotiations between different stockholders in a contract. The designed plan is considered to be fixed over a horizon despite the fact that, often in reality, producers' production levels may vary on daily or seasonal bases. The daily variations, caused by exceptional situations such as daily meteorological variations, cattle nutrition or cattle diseases, are quite minor. On the other hand, seasonal variations, caused by seasonal meteorological changes as well as animal birth cycles, are more significant. It is worth noting that, in this particular context, by a season, we do not necessarily refer to the conventional definition of a season (quarter of a year), but rather, the sub-periods of the year determined by variations in cattle's production levels, which could be shorter or longer than the traditional calendar use of the term.

Currently, these variations are not accounted for while designing the routes. More precisely, the producers' production over a horizon is represented by their mean values, often obtained from historical data. Accordingly, executing such a plan during the periods of the horizon with higher-than-average production levels, a planned route may fail at a given producer as a result of insufficient residual capacity to serve each producer's demand. In this dissertation, such a situation is referred to as a *Failure*. Following a failure, the vehicle returns to its assigned plant to empty its tank, before resuming the route by visiting the remainder of its planned producers and delivering to the plant once again. We call this extra travel to a plant a *Recourse Action*. The main goal of this problem is to design a single plan over a given horizon, so that the total routing costs (the sum of planned routes and recourse costs) plus the vehicles fixed costs are minimized.

In this problem, since all the producers in a given geographical region experience the same seasonal circumstances, their seasonal variations are almost perfectly correlated. Accordingly, one can recognize periods of the horizon, corresponding to production seasons, in which all the producers' production levels are relatively fixed (ignoring potential daily variations). In fact, while designing the routes, potential daily variations, which represent the intra-period production fluctuations, are often absorbed by leaving a safety capacity of 5% to10% to be empty on each vehicle.

1.2 Thesis contributions

The main contributions of this dissertation are as follows:

- We introduce two variants (single- and multi-period) of the MAVRP inspired by milk collection and redistribution in Quebec. It differs from well-studied variants such as VRPTW and MDVRPTW because there is an extra level of difficulty associated with the assignment of routes to plants. We also investigate the characteristics of these problems.
- We first propose a set partitioning formulations for the single-period variant of the problem.
- We develop a branch-and-price algorithm. It includes a number of structural exploration and exploitation features that improve the computational efficiency of the solution strategy.
- We perform an extensive analysis using a large set of randomly generated instances, to illustrate the efficiency of the algorithm and investigate the characteristics of the problem.
- Next, we propose a mathematical programming model for the multi-period variant of the problem considering the seasonal behavior of the supply.
- We propose a state-of-the-art branch-and-price algorithm. It includes a series of bounds as well as structural modifications in the multi-period problem, allowing us to take advantage of technical advances in single-period VRPs.
- We perform an extensive analysis using a large set of randomly generated instances, to illustrate the performance and the limits of our algorithm.
- In the third part of the dissertation, we design a new metaheuristic based on the adaptive large neighborhood search (ALNS) for for the multi-period problem setting.
- We design several new operators based on the special structure of the problem. We also adapt some existing operators in the literature.
- We propose a new adaptive layer for the ALNS in which destruction and construction heuristics are coupled to form the operators, rather than being treated independently.
- We propose a new diversity management system for the ALNS, which is based on extracting information from a list of diverse solutions and restarting the search from a diverse solution when it seems to be trapped in a local optimum.
- To evaluate the quality of the solution, we compute a series of lower and upper bounds on the value of the multi-period solution. We compare the solutions obtained through the ALNS with these bounds.

- We perform a series of extensive numerical tests for a large set of randomly generated instances, to illustrate the performance of the algorithm in terms of computational time and solution quality.

1.3 Thesis organization

This dissertation addresses two combinatorial problems, encountered in a collection-distribution context. Chapter 2 covers the state of the art for the works related to the context of this dissertation. A brief classification of different exact and metaheuristic solution methods for the main extensions of the VRP is presented.

Chapter 3, the first paper of this dissertation, addresses the problem described in Section 1.1.1, as well as the details of the problem setting and the proposed solution method based on the branch-and-price algorithm. The performance of our algorithm is discussed in the same chapter.

The second paper of this dissertation is presented in Chapter 4. This paper addresses the second problem, described in Section 1.1.2. In this chapter, we present a set covering formulation for the multi-period variant of the problem. The solution method is again based on the branch-and-price algorithm. This chapter finishes with the presentation of the numerical results and an evaluation of the performance of our algorithm.

Chapter 5, forming the third paper of this dissertation, takes over the same problem as in Chapter 4. To be able to address more realistic size configurations of this problem, we propose a solution method, which is based on a metaheuristic framework. Different features of the adaptive large neighbourhood search, proposed in this paper, are discussed in detail. Extensive numerical tests are performed to evaluate the performance of our algorithm. The outcomes of these tests are presented at the end of this chapter.

Finally, Chapter 6 provides conclusions, including the evaluation of our work as well as our research perspectives.

Chapter 2

Literature Review

The vehicle routing problem (VRP), first introduced by Dantzig and Ramser [42], is one of the most challenging problems in the field of combinatorial optimization. The classical VRP is defined as the problem of designing least-cost delivery routes from a depot to a set of geographically scattered customers. However, due to the diversity of practical operating rules and constraints encountered in real-life applications, many variants and extensions of the classical VRP have been introduced and addressed in the literature. For books and survey articles on the VRP, readers are referred to Toth and Vigo [151], Cordeau et al. [37], Golden et al. [74] and Laporte [102].

The classical VRP, usually referred to as the capacitated VRP (CVRP), can formally be defined as follows: Let $G = (\mathcal{V}, \mathcal{A})$ be a directed graph, where $\mathcal{V} = \{0, \ldots, n\}$ is the vertex set and $\mathcal{A} \subseteq (\mathcal{V} \times \mathcal{V})$ is the arc set. Vertex 0 represents the depot, whereas the remaining vertices $\mathcal{N} = \{1, \ldots, n\}$ correspond to customers. A fleet of m identical vehicles of capacity Q is based at the depot. The fleet size is given *a priori* or is a decision variable. Each customer i has a non-negative demand q_i. A cost matrix c_{ij} is defined on arc set $\mathcal{A}$, where travel costs, distances and travel times are often considered to be proportional. The VRP consists of designing m vehicle routes, such that each route starts and ends at the depot, each customer is visited exactly once by a single vehicle, and the total demand of a route does not exceed Q.

Numerous extensions and variants of the CVRP are introduced in the literature. Among them, the following problems have similar attributes or characteristic to the problems of this dissertation:

The Vehicle Routing Problem with Time Windows (VRPTW): This problem is an extension of the VRP, in which the service for each customer must start within a time interval, referred to as a time window. If the vehicle arrives before the beginning of a customer time window, it has to wait. The VRPTW has been proved by Savelsbergh [137] as a NP-hard problem.

The Multi-Depot Vehicle Routing Problem (MDVRP): This problem is about

designing the most efficient schedule for the vehicles; while assuming that multiple depots, from which the vehicles depart and arrive, exist.

2.1 Set covering-based algorithms for the VRP

Exact methods for the VRP are based on two large formulation families: 1) flow-based formulations, and 2) set covering-based formulations. The flow-based formulation is often solved using the branch-and-cut approach, while the set covering-based formulations are solved using the branch-and-price approach.

In this section, we first start by presenting a general set covering formulation for the CVRP. We devote special attention to the branch-and-price approach, since this is the core algorithm used in Chapters 3 and 4 of this thesis.

2.1.1 Set covering formulation for the VRP

To present the set covering formulation of the classical VRP, let us suppose that $\mathcal{R}$ is the set of all feasible routes for a given problem. Moreover, the binary parameter a_{ir} is equal to 1, if customer i is visited by route $r \in \mathcal{R}$ and zero, otherwise. Let us also suppose that c_r and c_k stand for the variable and fixed vehicle costs associated with route r. The set covering formulation of the VRP is as follows:

$$\min \quad \sum_{r \in \mathcal{R}} (c_r + c_k) y_r \tag{2.1}$$

subject to

$$\sum_{r \in \mathcal{R}} a_{ir} y_r \geq 1 \quad (i \in \mathcal{N}); \tag{2.2}$$

$$y_r \in \{1, 0\} \quad (r \in \mathcal{R}). \tag{2.3}$$

In the above formulation, Constraints (2.2) insure that each producer is visited exactly once by exactly one route. Note that the set covering formulation is derived from a set partitioning formulation, in which Constraints (2.2) are under equality form. However, it is worth mentioning that in the case of problems, where the distance matrix satisfies the triangle inequality, the set partitioning and set covering formulations are equivalent. The advantage of working on the set covering formulation is related to its simpler implementation.

To solve the set covering formulation, first, a linear relaxation is often considered. The linear relaxation is obtained by replacing Constraints (2.3) with the following constraints:

$$0 \leq y_r \leq 1 \quad (r \in \mathcal{R}). \tag{2.4}$$

To solve the linear programming relaxation of the problem without enumerating all routes, one can use the column generation technique. In this procedure, a subset of all possible routes is enumerated, and the linear relaxation, which is restricted to this partial route set, is solved. Using the values of the optimal dual variables with respect to the partial route set, we solve a simpler optimization problem, where we identify whether there is a route that should be included in the formulation. Then, the linear relaxation of this expanded problem is resolved. This procedure continues until no new improving routes are found.

2.1.2 Branch-and-price for the VRP

For more than two decades, branch-and-price has been successful in solving a wide variety of the VRP (see Desrosiers et al. [51] and Desaulniers et al. [47]). Branch-and-price algorithm is a branch-and-bound in which the lower bounds are computed using the column generation technique (for a recent and complete survey of column generation, see Lübbecke and Desrosiers [109]). The column generation is a generalization of the [43]. Often, starting from the arc-flow formulation, the problem is formulated as a set partitioning problem on which the column generation is applied. The set partitioning formulation of the CVRP was originally proposed by Balinski and Quandt [15] and associates a binary variable with each feasible route. Such a formulation splits the problem into two major structures: a master problem and a subproblem. In the VRP application, each variable of the set partitioning formulation represents a feasible route. Moreover, most successful decomposition approaches for different variants of the VRP cast the subproblem as an elementary shortest path problem with resource constraints (ESPPRC).

The column generation is an iterative process, which is based on the interaction between the master problem and the subproblem. The master problem is, in fact, the linear relaxation of the set partitioning formulation of the problem, where the constraints regarding the feasibility of routes are evaluated in the subproblem. The subproblem involves a modified objective function based on the dual variables of the master problem. More precisely, the subproblem is responsible for finding feasible routes with negative reduced cost with respect to the current dual variables of the master problem, which are then fed to the master problem. At each iteration of the column generation, a restricted linear master problem (RLMP) is solved rather than the master problem. The set of the columns in the RLMP is limited to only those that have already been generated. Once the RLMP is solved, the objective function of the subproblem is updated with respect to the current dual variables of the RLMP. The interaction between the master problem and the subproblem continues until no negative-reduced-cost column is found through solving the subproblem.

The column generation does not guarantee the integrality of the solution. Therefore, a branch-and-price scheme, which is based on the idea of the branch-and-bound algorithm, is employed. In this approach, at the end of the column generation procedure, if the obtained solution is an integer, it is a valid solution to the original master problem. If the solution of the relaxed problem does not satisfy the integrality conditions, branching occurs to cut off the current fractional point. A valid branching scheme should cut off the current fractional solution, produce a balanced search tree and keep the structure of the problem unchanged. At the end of the search process, the best integer solution is the optimal solution for the original integer problem.

The master problem

A relaxation of Constraints 2.1 and 2.3 converts the set partitioning model into a set covering yielding the linear master problem. Consequently, the RLMP, which is restricted to a smaller subset of columns $\mathcal{R}' \subseteq \mathcal{R}$, takes the following form:

$$\min \quad \sum_{r \in \mathcal{R}'} (c_r + c_k) y_r \tag{2.5}$$

subject to

$$\sum_{r \in \mathcal{R}'} a_{ir} y_r \geq 1 \quad (i \in \mathcal{N}); \tag{2.6}$$

$$0 \leq y_r \leq 1 \quad (r \in \mathcal{R}'). \tag{2.7}$$

where Constraints 2.6 guarantee at least one visit to each producer.

The subproblem

The subproblem should find one or more columns with negative reduced costs with respect to a given dual solution of the RLMP. The subproblem takes the form of an ESPPRC, where the resource constraint concerns with the vehicle capacities. Consider the following dual variables of the RLMP (2.5)-(2.7):

- λ_i : nonnegative dual variable of (2.6) for producer $i \in \mathcal{N}$;

Let x_{ij} be a binary decision variable, which equals 1 if customer j follows customer i on the shortest path and 0, otherwise. Moreover, associated with each customer i there is a demand q_i. Using these notations, the subproblem takes the following form:

$$\min \quad \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} (c_{ij} - \lambda_i) x_{ij} + c_k \tag{2.8}$$

subject to

$$\sum_{i \in \mathcal{N}} x_{ih} - \sum_{j \in \mathcal{N}} x_{hj} = 0; \quad (h \in \mathcal{N}) \tag{2.9}$$

$$\sum_{j \in \mathcal{N}} x_{0j} = 1; \tag{2.10}$$

$$\sum_{i \in \mathcal{N}} q_i \sum_{j \in \mathcal{N}} x_{ij} \leq Q; \tag{2.11}$$

$$x_{ij} \in \{1, 0\} \quad (i, j \in \mathcal{N}). \tag{2.12}$$

In this model, Constraints 2.9-2.10 are flow constraints which result in a path from depot to itself. Constraints 2.11 are the capacity constraint, whereas Constraints 2.12 ensures the solution integrality.

Elementary shortest path problem with resource constraints

Most of advances in the field of the branch-and-price for the classical variants of the VRP such as the CVRP and the VRPTW, have been built upon proposing efficient methodologies to optimally solve the subproblem. As mentioned in Section 2.1.2, in most VRP applications solved by column generation, the subproblem corresponds to a shortest path problem with resource constraints (SPPRC), an ESPPRC, or one of their variants. In these problems, one may have resource constraints related to time (e.g., time windows, route duration), vehicle capacity and many other resources.

The standard approach to solve the SPPRC or the ESPPRC, in practice, is based on dynamic programming and has a pseudo-polynomial complexity. This approach, being an extension of the well-known Ford-Bellman algorithm, is to associate with each possible partial path a label indicating the consumption of resources. In the classical dynamic programming-based algorithms, starting from an initial label associated with the depot, the algorithm extends the labels along arcs using extension functions. To avoid creating a huge number of labels, dominated labels are eliminated by some dominance rules.

The elementary condition adds an extra layer of complexity to the SPPRC, being a NP-hard problem [53]. One of the main papers dedicated to the ESPPRC, especially as a subproblem for a column generation methodology, is the work of Feillet et al. [55]. The authors proposed an exact algorithm, based on the label-correcting algorithm of M. [110], in which new resources are introduced to enforce the elementary path constraint. Their algorithm has strongly improved the efficiency of the algorithm by introducing the idea of *unreachable nodes* supporting the elementary paths.

In a classical labeling procedure for an ESPPRC, based on Feillet et al. [55], where resource constraint is the vehicle capacity, each label $\sigma = (C, L, S, \phi)$ has the following component:

1. C for the reduced cost,
2. L for the vehicle load,
3. S for the number of unreachable nodes,
4. set ϕ containing unreachable nodes from the current label.

A label $\sigma_1 = (C_1, L_1, S_1, \phi_1)$ dominates a label $\sigma_2 = (C_2, L_2, S_2, \phi_2)$, if the following criteria are met:

(a) $L_1 \leq L_2$,

(b) $C_1 \leq C_2$,

(c) $S_1 \leq S_2$,

(d) $\phi_1 \subseteq \phi_2$.

Since the paper of Feillet et al. [55], many authors addressed the ESPPRC and many improvements have been achieved in accelerating the solution of the ESPPRC. The most promising methodologies for the ESPPRC, relaxing partially the elementary conditions, are based on either the decremental state-space relaxation (DSSR), on the ng-route relaxation, or on ng-route decremental state-space relaxation (ngR-DSSR). These two strategies are briefly introduced in the following sections.

State-space relaxation

In the DSSR strategy, initially proposed by Boland et al. [23], the elementary conditions are completely or partially relaxed, turning the ESPPRC into a SPPRC. Adding new resources, up to obtaining the optimal elementary solution, iteratively tightens the relaxation. At the end of each iteration of solving the SPPRC, using the label correcting procedure, all columns with no cycle are added into the master problem. Moreover, the set of critical nodes are recognized and through a state-space augmentation strategy, cycle formation on one or more of them is prohibited for the next iterations. A critical node refers to a node on which a cycle has occurred in the relaxed problem, SPPRC. The process repeatedly continues by adding partial elementary conditions to the SPPRC as long as new negative reduced cost columns, even containing cycles, price out. Several state-space augmentation strategies have been proposed by Boland et al. [23].

The main advantage of this strategy is a reduction in the number of resources stored on each label to be compared and updated following each label extension, while also guaranteeing the elementarity of the obtained columns. Let us suppose that for a given label σ, resource $\widehat{S}$ keeps the number of unreachable critical nodes, and a set $\widehat{\phi} \subseteq \Phi$ contains the critical nodes unreachable from the current label. The set Φ stores all the nodes recognized as critical. Accordingly, for a label σ_1 to dominate a label σ_2 the following conditions are required:

(a_{dssr}) $L_1 \leq L_2$,

(b_{dssr}) $C_1 \leq C_2$,

(c_{dssr}) $\widehat{S_1} \leq \widehat{S_2}$,

(d_{dssr}) $\widehat{\phi_1} \subseteq \widehat{\phi_2}$.

ng-route relaxation

In the ng-Route Relaxation strategy, proposed by Baldacci et al. [13], the elementary conditions are partially relaxed. Consequently, the optimality of the lower bound is sacrificed for the sake of time efficiency. Following this relaxation, near-elementary routes are often generated with a fraction of the computational effort.

Based on this strategy, a new state-space relaxation is used to compute lower bounds to routing problems, such as the CVRP and the VRPTW. It consists of partitioning the set of all possible paths ending at a generic vertex according to prefixed neighbourhoods of graph vertices and also a mapping function. The latter associates a subset of the visited vertices with each path that depends on the order in which such vertices are visited. The subset of vertices associated with each ng-path is used to impose partial elementarity. This relaxation was proved to be particularly effective in computing lower bounds in the CVRP, the VRPTW and the traveling salesman problem with time windows (TSPTW).

The ng-Route Relaxation performs as follows. Suppose that $\mathcal{V}_r$ represents the set of producers visited by partial path r. Moreover, for each customer $i \in \mathcal{N}$, let $\mathcal{N}_i \subseteq \mathcal{N}$, the so-called neighborhood of producer i, represent a set of producers, selected according to a neighborhood criterion for customer i. For label σ, associated with a given partial path $r = (0, i_1, \ldots, i_n)$, we define a set $\Pi(r) \subseteq \mathcal{V}_r$, containing all prohibited immediate extensions from customer i_n. The set $\Pi(r)$ is

$$\Pi(r) = \{i_j \in \mathcal{V}_r | i_j \in \bigcap_{k=j+1}^{n} \mathcal{N}_{i_k}, j = 1, \ldots, n-1\} \cup \{i_n\}. \tag{2.13}$$

Consequently, each label σ has new members S_{ng}, representing the size of the Π set, where Π represents the set of inaccessible customers according to the ng-rules. Given two labels, $\sigma_1 = (C_1, T_1, L_1, S_{ng_1}, \Pi_1)$ and $\sigma_2 = (C_2, T_2, L_2, S_{ng_2}, \Pi_2)$, label σ_1 dominates label σ_2 if and only if the following conditions are satisfied:

(a_{ng}) $L_1 \leq L_2$,

(b_{ng}) $C_1 \leq C_2$,

(c_{ng}) $S_{ng_1} \leq S_{ng_2}$,

(d_{ng}) $\Pi_1 \subseteq \Pi_2$.

ng-Route Decremental State-Space Relaxation (ngR-DSSR)

An efficient ng-route pricing was proposed by Martinelli [111], in which a DSSR technique is embedded into the ng-route relaxation. It consists of an ng-route relaxation procedure in which resources associated with the vertices' neighbours are initially deactivated. These neighbourhoods are iteratively augmented based on a DSSR scheme to insure the ng-feasibility of all obtained columns.

2.2 Metaheuristics for the VRP

In this section, we review the most popular categories of metaheuristics used in the literature of the VRP. Complete surveys on the application of metaheuristics to different extensions of the VRP can be found in [68, 156]. There exist the following two major categories of metaheuristics [147, 63].

1. Neighborhood-based methods,
2. Population-based methods.

Using the concepts from these two categories, two classes of methodologies are also designed: 1) hybrid metaheuristics, and 2) parallel an cooperative metaheuristics. Hybrid metaheuristics, combine concepts from various solution methodologies or metaheuristic classes to take advantage of their respective strengths. The combining may take the form of a juxtaposition of methods (e.g., two algorithms called on consecutively) or an indissociable inclusion of elements from one method into a fully-functional different metaheuristic. Hybrids may exclusively combine metaheuristic concepts, or also involve algorithmic ideas and modules from mathematical programming, constraint programming, tree-search procedures, and so on. On the other hand, parallel metaheuristics are concerned with the efficient exploitation of simultaneous work (often on several processors) to solve a given problem instance [39].

2.2.1 Neighborhood-based metaheuristics

While solving optimization problems, neighborhood-based metaheuristics improve a single solution. They could be viewed as search trajectories through the neighbourhoods of the search space of the problem at hand [38]. The trajectories are performed by iterative procedures that move from one solution to another one in the search space. A search trajectory is defined as follows: from a solution x_t at iteration t, the search moves to a solution x_{t+1} in the neighbourhood of x_t until a stopping criterion is met.

Numerous neighborhood-based metaheuristics have been proposed over the past decades. Some of the main single solution-based metaheuristics are simulated annealing

(SA) [93, 154], tabu search [73, 70, 71], variable neighbourhood search (VNS) [113, 78], large neighbourhood search (LNS) [142] and finally adaptive large neighbourhood search (ALNS) [134, 135, 121].

In simulated annealing, the heuristic considers some neighbouring state x_{t+1} of the current state x_t at each iteration, and probabilistically decides between moving the system to state x_{t+1} or staying in state x_t. These probabilities are based on a temperature parameter, which is lowered from one iteration to the next. The probability of accepting a non-improving solution diminishes with the value of the temperature, as the algorithm unfolds. If the temperature is reduced sufficiently slowly, then the system can reach an equilibrium (steady state) at each iteration. Typically this step is repeated until the system reaches a state that is good enough for the application, or until a given computation budget has been exhausted. The most well-known simulated annealing algorithms in the context of the VRP are developed by Golden et al. [76] and Osman [117].

In tabu search, at each iteration, starting from a solution x_t, the algorithm moves to the best solution x_{t+1} in the neighbourhood of x_t, even if this leads to a deterioration of the objective function value. To avoid cycling, attributes of recently visited solutions are declared tabu for a certain number of iterations. This process is repeated until a stopping criterion is satisfied. Some of the most well-known tabu search algorithms for the VRP are proposed by Gendreau et al. [64, 65], Rego and Roucairol [129], Xu and Kelly [159], Bachem et al. [8] and Toth and Vigo [152].

The variable neighbourhood search explores distant neighborhoods of the current solution x_t, and potentially moves to a new solution x_{t+1}. Based on the fact that a local optimum is defined for a given neighbourhood, the VNS systematically changes the nature or the parameters of the neighborhoods. The space exploration during the search is performed in two phases; firstly, descent to find a local optimum and finally, a perturbation phase to get out of the corresponding valley. The order of neighbourhood evaluations and the solution acceptance criteria can be either deterministic or probabilistic. For the CVRP, additional solution perturbation mechanisms are sometimes employed [99, 58, 28].

In large neighbourhood search, an initial solution is gradually improved by alternately destroying and constructing the solution. The LSN algorithm is based on the observation that searching a large neighborhood may result in finding high quality local optima, and consequently, promising final solutions. The LNS algorithm is also related to the ruin-and-recreate approach of Schrimpf [138].

In the same spirit, the adaptive large neighbourhood search extends the large neighborhood search heuristic of Shaw [142] by allowing the use of multiple destruction and construction operators within the same search process. This method belongs to the class of very large-scale neighborhood search algorithms [122]. At each iteration t, the heuristic destroys a part of the current solution x_t using a destruction operator and then reconstructs it, using a construction operator, so that a new solution x_{t+1} is generated. The frequency of applying each operator is adapted throughout the search relatively to

its past performance. The new solution x_{t+1} replaces the current solution through an acceptance criterion, which is often based on simulated annealing. Recently, ALNS has provided good solutions for a wide variety of vehicle routing problems. For example see [134, 121, 122, 7, 118].

2.2.2 Population-based metaheuristics

Population-based metaheuristics could be viewed as iterative improvement in a population of solutions. Most population-based metaheuristics share the following common concepts: First, the population is initialized. Then, a new population of solutions is generated. Finally, this new population is integrated into the current one using some selection procedures. The search process is stopped when a given stopping criterion is satisfied.

Some examples of population-based metaheuristics are: genetic algorithms (GA) and evolutionary algorithms (EA) [84], scatter search (SS) and its generalization called path relinking (PR) [69, 131] as well as particle swarm optimization (PSO) [92, 143].

Evolutionary algorithms are inspired from a natural metaphor, while genetic algorithm is one of the best-known solution methodologies. Different main types of evolutionary algorithms have independently evolved during the past five decades. For example, genetic algorithms, mainly developed by Holland [82, 83], and evolution strategies, developed by Rechenberg [127, 128]. Basically, the evolutionary algorithms mimic the way that species evolve and adapt to their environment, according to the Darwinian principle of natural selection.

Under this paradigm, a population of solutions, known as chromosomes, evolves from one generation to the next through the application of operators that are similar to those found in nature, like survival of the fittest, genetic crossover and mutation. Traditional GA and EA have a tendency to progress too slowly. However, they have been enhanced with various education operators such as local search. Many successes in genetic algorithms for the CVRP have been achieved since the introduction of giant-tour solution representation without trip delimiters by citetPrins2004. Some of the most well-known GAs for the CVRP are proposed by Baker and Ayechew [10], Prins [125], Nagata et al. [115], Vidal et al. [155], while a complete survey on the EAs is provided by Potvin [123].

Scatter search is an evolutionary metaheuristic whose main idea is based on the fact that useful information about the solution is typically contained in a diverse collection of elite solutions. Therefore, multiple solution combinations enhance the opportunity to exploit information contained in the union of elite solutions. The combination strategies incorporate both diversification and intensification. The scatter search explores solution spaces by evolving a set of reference points, operating on a small set of solutions while making only limited use of randomization. On the other hand, path-relinking is an intensification strategy used to explore trajectories connecting elite solutions obtained by heuristic methods such as scatter search and tabu search. In path-relinking,

unlike other evolutionary methods, such as GAs, where randomness is a key factor in the creation of offspring from parent solutions, path-relinking systematically generates new solutions by exploring paths that connect elite solutions. To generate the desired paths, an initial solution and a guiding solution are selected from a so-called reference list of elite solutions to represent the starting and the ending points of the path. Attributes of guiding solution are progressively inserted in the initial solution, so that the intermediate solutions contain fewer characteristics from the initial solution and more from the guiding solution. The trajectory created from connecting these two solutions potentially contains new improving solutions. Ho and Gendreau [81] proposed a path-relinking-based algorithm for the VRP.

Particle swarm optimization is another optimization method which iteratively tries to improve a candidate solution with respect to a given quality measure. This method is based on a population of candidate solutions, called particles, and moving these particles around in the search-space according to the experiments under study. Each particle has its vector position, while each particle's movement is influenced by its local best known position and is also guided toward the best known positions in the search-space, which are updated as better positions are found by other particles. Thus, at each iteration, particles move to a new vector position in the research space and the process is repeated until a stop condition is met, usually after a certain number of iterations.

2.3 Selected extensions of the VRP

In this section, we review the literature of two VRP extensions, which include similar features to the problems considered in this dissertation. For each class of problem, we first present the exact algorithms followed by classifying metaheuristic methods.

2.3.1 VRP with time windows

The vehicle routing problem with time window (VRPTW) is an important generalization of the classical VRP, in which the service at each customer i must begin within a time window $[a_i, b_i]$. A vehicle may arrive before a_i and wait until the client becomes available, while arriving after b_i is prohibited. The VRPTW has several applications in the management of distribution such as food and beverage delivery, newspaper delivery and collection of industrial and commercial waste [75]. Therefore, the VRPTW is one of the most intensively studied NP-hard combinatorial optimization problems in the last three decades. In fact, the VRPTW is a generalization of the VRP, when $a_i = 0$ and $b_i = \infty$ for all customer i.

Exact methods are still not able to address most large-size applications of the VRPTW. Moreover, the performance of existing exact methods strongly varies according to the time-window characteristics. Therefore, heuristic and meta-heuristic approaches have been the methodology of choice (see Bräysy and Gendreau [26, 27]

and Gendreau and Tarantilis [63].). The algorithms proposed have been mostly evaluated and compared on standard benchmark instances introduced by Solomon [146] and Gehring and Homberger [61] relative to their computational efficiently and the quality of the solutions obtained. In this section, we first start by focusing on exact methods and then the metaheuristic approaches for the VRPTW are presented.

The first optimization algorithm for the VRPTW has been proposed by Kolen et al. [96], who used a dynamic programming procedure coupled with a state-space relaxation [31] to find lower bounds in a branch-and-bound algorithm. Among the exact methods, three approaches have been applied to address the VRPTW: the Lagrangian relaxation methods, as well as the column generation and branch-and-cut approaches. Each of these methods is briefly described below.

Fisher [56] and Fisher et al. [57] proposed a Lagrangian relaxation based on m-tree by relaxing the constraints on the flow conservation, capacity, and time window and then they proposed formulations to manage each of the violated capacity and time window constraints. Kohl and Madsen [94] and Kallehauge et al. [91] proposed other exact solution methods based on Lagrangian relaxation for VRPTW.

Balinski and Quandt [15] were among the first to propose a set partitioning formulation for the CVRP. Later, Desrosiers et al. [50] and Kohl et al. [95] also applied the column generation approach for the VRPTW. More recently, Danna and Pape [41] developed a cooperative scheme between branch-and-price and local search applied to the VRPTW, to speed up the branch-and-price finding good integer solutions. During the process of branch-and-price, a local search is applied from the best known integer solution. This often results in improving the upper bounds that will later be used to prune new nodes in the tree. In addition, new columns obtained in the course of the local search can be injected into the restricted master problem. Thus, the algorithm of branch-and-price, taking advantage of the local search is provided at each stage by high quality upper bounds. On the other hand, the local search takes advantage of branch-and-price, by working on a variety of different initial solutions and therefore, an effective diversification form.

A branch-and-cut algorithm for the VRPTW was developed by Bard et al. [16]. The algorithm incorporates five inequalities types: subtour elimination constraints, comb inequalities, incompatible path inequalities, time and load constraints and incompatible pair inequalities. At each node of the search tree, an upper bound is computed by means of a "greedy randomized adaptive search procedure (GRASP)," proposed by Kontoravdis and Bard [97].

Desaulniers et al. [48] proposed an efficient branch-and-price-and-cut approach, in which the three following conclusions are presented: First, the application of the tabu search method for heuristically solving the subproblem and rapidly generating negative reduced cost columns showed to be effective. Second, the generalization of the k-path inequalities did not prove to be very useful in practice. Third, the partial elementary shortest path problem with resource constraints subproblem provided a compromise between lower bound quality and subproblem complexity.

As mentioned before, due to the difficulty of the VRPTW and its practical relevance, there is a strong need to develop fast algorithms that are capable of producing good quality solutions in a shorter time. Here, we focus on the meta-heuristics proposed in the literature for the VRPTW. A complete survey on the application of the meta-heuristics to address the VRPTW is presented in Gendreau and Tarantilis [63].

The first application of tabu search, as a meta-heuristic, to the VRPTW can be attributed to Semet and Taillard [141] and Potvin et al. [124], who combined an insertion heuristic proposed by Solomon [146] with an improving idea based on node and chain exchange precedure. Badeau et al. [9] developed and implemented a parallel tabu search for the VRPTW. In the proposed algorithm, first, some initial solutions are generated using a stochastic insertion heuristic. Each of the initial solutions is then improved by applying a tabu search. Finally, for a predetermined number of iterations, a solution is constructed from routes obtained from each iteration. A process called decomposition/reconstruction method enhances this solution. A hybrid method to solve the VRPTW is presented by Rousseau et al. [136]. Their Method, based on the work of Pesant and Gendreau [119], describes how a constraint framework can be used to efficiently explore large neighborhoods using a branch-and-bound procedure. The proposed algorithm by Rousseau et al. [136] uses three operators, which define three different neighborhoods. During neighborhood exploration through branch-and-bound, propagation and pruning are used to reduce the search space. Bräysy and Dullaert [24] implemented a fast evolutionary metaheuristic for the VRPTW. The proposed method consists of several steps: an initial set of solutions is first generated, then, the number of routes in each initial solution is reduced through a new ejection chains procedure. Finally, the total distance is minimized using an evolutionary metaheuristic which consists of moving the chains of consecutive customers and a parallel insertion heuristic.

The most competitive results are currently offered by the hybrid genetic algorithm of Nagata et al. [115] and Vidal et al. [157]. Nagata et al. [115] proposed a method, which combines powerful route minimization procedures, with a very effective edge assembly crossover, and extremely efficient local search procedures. On the other hand, Vidal et al. [157] proposed an efficient hybrid genetic search with advanced diversity control for a large class of time-constrained vehicle routing problems, including multi-depot vehicle routing problem with time windows (MDVRPTW), periodic vehicle routing problem with time windows (PVRPTW), and split delivery vehicle routing problem with time windows (SDVRPTW). The proposed algorithm, based on the hybrid genetic search coupled with local search of Vidal et al. [155], outperforms all current state-of-the-art approaches on classical literature benchmark instances for any combination of periodic, multi-depot, site-dependent, and duration-constrained vehicle routing problem with time windows.

The Adaptive Large Neighbourhood Search of Pisinger and Ropke [121] and the Unified Tabu Search (UTS) of Cordeau et al. [34, 35, 36] are also worth mentioning. These methods stand out in terms of simplicity and wider applicability, as both have been extended to address various VRP variants.

Pisinger and Ropke [121] addressed a large variety of VRP extensions, including

VRPTW. Different problem settings are first transformed to a rich pickup and delivery problem with time windows (RPDPTW). In order to transform a VRPTW instance to a RPDPTW instance, the authors map every customer in the VRPTW to a request in the RPDPTW. Such a request consists of a pickup at the depot and a delivery at the customer site. The amount of goods that should be carried by the requests is equal to the demand of the corresponding customer. The time window of the pickup is set to $[a_d, a_d]$, where a_d is the start of the time window of the depot in the VRPTW and its service time is set to zero. The time window and service time of the delivery correspond to the original VRPTW. The obtained RPDPTW is solved using the adaptive large neighborhood search framework introduced by Ropke and Pisinger [134] and Ropke and Pisinger [135]. They proposed a time oriented removal operator, in which they try to remove requests that are served at roughly the same time hoping that these requests are easy to interchange.

2.3.2 Multi-depot VRP and periodic VRP

The multi-depot vehicle routing problem is another generalization of the classical VRP, in which the vehicles are housed in several depots. In other words, routes can originate from several available depots. The goal is to design a set of lowest-cost routes, so that all the customer demands are met. The MDVRP, being an extension of the VRP, is a NP-hard problem as proved by Bertossi et al. [20].

Three main variants of the MDVRP are possible. In the first type, called problems with fixed fleet, each vehicle ends its route to the same depot, where it has started. In addition, the number of all vehicles housed in each depot is fixed. This is the case of situations in which several carriers perform the transportation task, while each company has its own depot with fixed sets of vehicle assigned to each depot. The second possibility is that the number of vehicles assigned to each depot varies; that is, one can manoeuvre the vehicle-depot assignments. This is the case of a company that can affect the vehicles at its depots depending on the distribution of demands on the network. The third situation is when each vehicle can end its route at a depot which is not necessarily the one from which the route began. The most studied variant of the MDVRP in the literature is the first case.

A problem closely related to the MDVRP is the periodic VRP (PVRP). The PVRP was initially introduced in the seminal paper by Beltrami and Bodin [17]. In this problem, vehicle routes must be constructed over multiple units of time (e.g., day), forming a time horizon. Each customer is characterized by a service frequency, representing the number of visits to be performed during the time horizon, as well as a list containing the possible visit-period combinations, called patterns. The main goal of the PVRP is to select a visit pattern for each customer, based on which a set of least-cost routes over the time horizon are constructed. The MDVRP can be formulated as a PVRP by realizing that different depots can be modeled as multiple periods in the context of a PVRP. Therefore, any algorithm that solves the PVRP can also solve the MDVRP.

The literature on exact approaches for the MDVRP is sparse. Laporte et al. [104]

developed a branch-and-bound algorithm that is capable of solving problems of small size (up to 50 clients and 8 depots) with symmetrical cost matrix. Later, Laporte et al. [105] studied the asymmetrical classes of the MDVRP. In the first step, the algorithm transforms the problem into an assignment problem. Optimal solutions are then found by a branch-and-bound algorithm, in which the subproblems are assignment problems. This method is capable of solving problems with up to 80 customers and 3 depots.

A recent exact method reporting results on the MDVRP is one proposed by Baldacci and Mingozzi [11]. This method is based on the additive bounding procedure, initially proposed by Christofides et al. [30], applied to several different relaxations of the problem. The MDVRP is formulated as a set partitioning and solved using a column generation strengthened with the so-called strong capacity constraints and clique inequalities. Baldacci et al. [12] proposed an exact algorithm for the PVRP that generalizes their former method for the MDVRP.

Mingozzi [112] proposed an exact method to to solve the MDPVRP. In the proposed method, the problem is first modeled using an extension of the set partitioning formulation of the CVRP. The proposed exact method to solve the problem uses variable pricing in order to reduce the set of variables to more practical proportions. The pricing model is based on the bounding procedure for finding near optimal solutions of the dual problem of the LP relaxation of the set partitioning model. The bounding procedure is an additive procedure that determines a lower bound on the MDPVRP as the sum of the dual solution costs obtained by a sequence of five different heuristics for solving the dual problem, where each heuristic explores a different structure of the MDPVRP. Three of these heuristics are based on relaxations, whereas the two others combine subgradient optimization with column generation.

Two meta-heuristic algorithms based on a large neighborhood search and tabu search were presented by Pepin et al. [118]. Using a set of randomly generated instances, the authors compared these two methods with three other existing approaches: A heuristic application of CPLEX MIP solver, a Lagrangian heuristic and a heuristic column generation. They showed the dominance of heuristic column generation, in terms of solution quality. Based on their results, if the goal is to find a compromise between quality and computational time, the large neighborhood search is the best alternative.

Renaud et al. [130], Cordeau et al. [34, 35] also proposed algorithms based on tabu search for MDVRP. Renaud et al. [130] proposed a tabu search heuristic, in which an initial solution is built by first assigning each customer to its nearest depot. A petal algorithm is then used to solve the VRP associated with each depot. The algorithm is completed by an improvement phase using either a subset of the 4-opt exchanges to improve individual routes, swapping customers between routes from the same or different depots, or exchanging customers between three routes. Unified tabu search by Cordeau et al. [34] can be used to solve the MDVRP, the PVRP and also periodic travelling sales man problem (PTSP). In their algorithm, assigning each customer to its nearest depot generates an initial solution and a VRP solution is generated for each depot by means of a sweep algorithm. Improvements are performed by transferring a

customer between two routes incident to the same depot, or by relocating a customer in a route associated with another depot. Insertions and removals of customers are performed using the GENI heuristic [64] operator. The authors introduced an innovative guidance scheme, which collects statistics on customer assignments to periods and vehicle routes in order to penalize recurring assignments within the solutions obtained and, thus, gradually diversify the search. For a long period of time, this method stood as the state of the art solution approach for both the PVRP and the MDVRP. It has been first outperformed by the variable neighborhood search (VNS) of Hemmelmayr et al. [80]. This latter is based on various well-known VRP neighborhoods,such as string relocate, swap, and 3-opt. It is also worth mentioning the VNS algorithm of Pirkwieser and Raidl [120] with multilevel refinement strategy, which is specifically designed for large-size instances.

Pisinger and Ropke [121] addressed a large variety of the VRP extensions, including the MDVRP. The MDVRP is first transformed to a RPDPTW, which is then solved using the ALNS framework. The MDVRP transformation to a RPDPTW is performed as follows: Create a dummy base location, where all routes start and end and where all ordinary requests are picked up. Moreover, create a dummy request for each vehicle k in the problem. The pickup and delivery locations of these requests are located at the depot of the corresponding vehicle. A dummy request has zero demand, it does not have any service time and it can be served at any time. The set N_k of each vehicle k contains all ordinary requests and the dummy request corresponding to the vehicle. The precedence of a pickup and a delivery corresponding to an ordinary request are set to zero and two, respectively, while the precedence of the pickup and delivery of the dummy requests are set to one and three, respectively. The distance and travel time between an ordinary request pickup and any other location is set to zero. All other distances and travel times are set as defined by the original MDVRP.

Dondo and Cerdá [52] studied the case of a vehicle routing problem with multiple depots in the presence of time windows. They suggested an approach based on a large-scale neighbourhood search algorithm that gradually improves an initial solution through a three-phase cluster-based hybrid approach. Lau et al. [106] addressed an MDVRP using a genetic algorithm with fuzzy logic adjusting the crossover rate and mutation rate.

Cordeau and Maischberger [33] proposed a parallel tabu search-based algorithm to address several variants of the VRP, including the MDVRP. Their algorithm is based on embedding tabu search within the framework of iterated local search.

Most solution methods proposed address the periodic and multi-depot VRP settings, with neighborhood-based methods yielding, the best results on standard benchmark instances. Two evolutionary approaches, which both take advantage of geometric aspects within the the MDVRP were proposed. Thangiah and Salhi [149] represented solutions as circles in the 2D space, whereas Ombuki-Berman and Hanshar [116] introduced a mutation operator that targets the depot assignment to "borderline" customers, which are close to several depots. Recently, evolutionary methods have proven to be efficient on the standard VRP [125, 114, 155]. Prins [125] introduced an im-

portant methodological element, namely the solution representation for the VRP as a TSP tour without delimiters along with a polynomial time algorithm to partition the sequence of customers into separate routes. Vidal et al. [155] proposed a hybrid genetic algorithm to address MDVRP, PVRP and MDPVRP. The most interesting feature of the proposed algorithm is based on combining the exploration capabilities of genetic algorithms with efficient local search-based improvement procedures and diversity management mechanisms. The authors proposed a new population diversity management mechanism, which allows broader access to reproduction, while preserving the memory of what characterizes good solutions represented by the elite individuals of the population. The method evolves feasible and infeasible solutions in two separate subpopulations. Genetic operators are iteratively applied to select two parents from the subpopulations, combine them into an offspring, which undergoes a local search-based education, is repaired if infeasible, and is finally inserted into the suitable subpopulation. The method terminates when a predefined number of successive iterations have been performed without improvement.

Crainic et al. [40] proposed a parallel cooperative search method, called integrative cooperative search (ICS), to solve multi-attribute combinatorial optimization problems. The ICS framework relies on an attribute decomposition approach and its structure is similar to a self-adaptive evolutionary meta-heuristic evolving several independent populations, where one of these populations corresponds to the solutions of the main problem. The other populations consist of the solutions addressing specific dimensions of the problem, obtained by attribute fixing. The authors used the MDPVRP with time windows to illustrate the applicability of the developed methodology. Lahrichi et al. [100] enhanced the idea of ICS. The proposed ICS framework involves problem decomposition by decision sets, integration of elite partial solutions yielded by the sub-problems, and adaptive guiding mechanism to solve highly complex combinatorial optimization problems. The authors used the MDPVRP to present the applicability of the developed methodology.

Chapter 3

A Column Generation Approach for a Multi-Attribute Vehicle Routing Problem

This paper has been submitted to the European Journal of Operational Research.

Abstract

In this paper, we consider a deterministic multi-attribute vehicle routing problem derived from a real-life milk collection system. This problem is characterized by the presence of a heterogeneous fleet of vehicles, multiple depots, and several resource constraints. A branch-and-price methodology is proposed to tackle the problem. In this methodology, different branching strategies, adapted to the special structure of the problem, are implemented and compared. The computational results show that the branch-and-price algorithm performs well in terms of solution quality and computational efficiency.

Keywords: Multi-attribute vehicle routing problem, Heterogeneous fleet, Multiple depots, Branch-and-price.

3.1 Introduction

The vehicle routing problem (VRP) lies at the center of logistics and distribution management and is one of the most studied problems in the field of operations research. Numerous variants have been studied since the problem was first introduced by Dantzig and Ramser [42]. The simplest problem in this domain is the capacitated vehicle routing problem (CVRP). In the CVRP, all the customers correspond to deliveries. The customers' demands are deterministic, known in advance, and may not be split. The vehicles are identical and based at a single central depot. Each vehicle can perform only one route, and the quantity supplied cannot exceed the vehicle capacity. The objective most commonly used is to minimize the total cost (i.e., a weighted function of the number of routes and their length or travel time) of serving all the customers [151].

In recent decades, there has been a tremendous improvement in algorithms that find good solutions to practical variants of the VRP in a reasonable time. This is due not only to the general increase in computing power, but also to significant advances in both exact and heuristic methods. However, VRP research has often been criticized for being too focused on nonrealistic models, and simplifying assumptions reduce the practical applications.

Many real-world combinatorial optimization problems, including logistics applications and transportation problems, have several complicating attributes. Theses attributes lead to the characteristics, constraints, and objectives that define the problem. When there are many attributes the problem becomes complex and challenging. In the combinatorial optimization literature, such problems are called "multi-attribute problems." Recently, the research community has focused on simultaneously considering multiple attributes, to provide more representative models of real-world situations. In particular, VRP researchers have recently concentrated on multi-attribute vehicle routing problems (MAVRP) [79]. They have explored several variations of the MAVRP, each representing a specialized extension of the classical VRP and reflecting a real-world application. However, not all variants have received the same attention. Furthermore, most of the contributions have developed heuristics and metaheuristics, and there are few efficient exact algorithms for the variants of the MAVRP.

We introduce a new MAVRP variant that incorporates some common real-world features. It is inspired by collection-redistribution activities in the raw-milk industry of Quebec. This problem consists of route planning for a heterogeneous fleet of vehicles departing from different depots. The vehicles must visit a set of producers in specific time windows, and the collected product is then delivered to processing plants. Finally, the vehicles return to their home depots. The most similar model in the literature is the multi-depot heterogeneous vehicle routing problem with time windows (MDHVRPTW).

The main goal of this paper is to investigate the challenges of complex problems with features such as collection-redistribution activities. We formulate a multi-attribute VRP with certain special features that takes the form of an MDHVRPTW with deliv-

eries to plants. The main contributions of this paper are summarized as follows:

- We introduce a variant of the MAVRP. It differs from well-studied variants such as VRPTW and MDVRPTW because there is an extra level of difficulty associated with the assignment of routes to plants.
- We propose a set partitioning formulation for this problem.
- We develop a branch-and-price algorithm. It includes a number of structural exploration and exploitation features that improve the computational efficiency of the solution strategy.
- We perform an extensive analysis using a large set of randomly generated instances, to illustrate the efficiency of the algorithm and investigate the characteristics of the problem.

The remainder of the paper is organized as follows. In Section 3.2, we describe in detail the problem class and its different variants. In Section 3.3, we give a brief literature review to better position the present study. In Section 3.4, we choose a special case of the problem class and present the set partitioning model. In Section 3.5, we present the proposed solution methodology, and experimental results are given in Section 3.6. Finally, Section 3.7 provides concluding remarks.

3.2 Problem class

In this section, we introduce a new MAVRP variant inspired by the dairy problem in Quebec [101]. This problem represents many real-world transportation activities. Basically, it consists of constructing collection routes that are then assigned to plants that receive the collected products. It is usually encountered in the collection and redistribution of perishable products. There are three types of stakeholders, as described below:

- The producers, which periodically produce a limited quantity of one or more products.
- The plants, which periodically receive the products. They transform these raw materials into consumable goods.
- The carriers, which collect the products from the producers and deliver them to the plants. Each carrier has one or more depots where the vehicles are located. The vehicles usually have different capacities, fixed costs, and variable costs. The fixed costs are the expenses that are not related to the distance traveled and have to be paid when the vehicle is used; the variable costs depend on the distance traveled.

In most applications, each producer has an associated time window indicating the earliest and latest collection times. Each plant has an associated demand window indicating the minimum and maximum quantities that can be delivered.

A route is a path that starts and ends at a depot and visits producers and plants; it may contain one or more pick-up and delivery phases. A route is feasible if the pick-ups do not exceed the vehicle capacity and the associated time windows are respected. The cost of a route is the sum of the costs of the arcs on the path plus the sum of the vehicle's fixed and variable costs. We assume throughout this paper that the triangle inequality holds for the costs and travel times. Also, the service times are considered to be independent of the quantities collected or delivered.

There may be some preassignments based on contractual restrictions, strategic/tactical planning decisions, or equipment compatibility. We introduce three: (1) producer-depot preassignments, which assign a producer to a specific depot; (2) producer-plant preassignments, which specify which plant receives the products of a given producer; (3) producer-depot-plant preassignments, which assign a producer to a depot and a plant. The most general variant of the problem has no preassignments.

A vehicle can perform one or more circuits per day. We define three route types as follows:

Simple route: Each vehicle visits several producers and collects their products. It then delivers its entire load to one plant and returns to its depot.

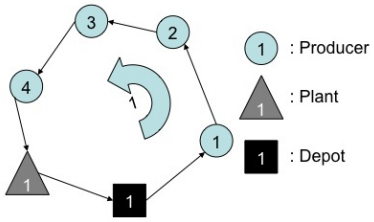

Figure 3.1: General configuration of simple route

Multi-drop route: A vehicle delivers its load to more than one plant before returning to its depot.

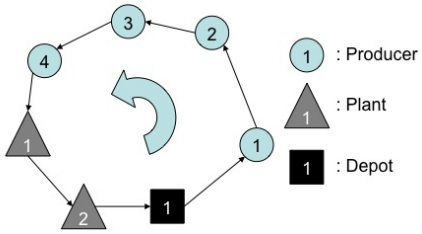

Figure 3.2: General configuration of multi-drop route

Interlaced multi-drop routes: Vehicles perform several circuits per day. A vehicle may visit other producers after completing its first visit to a plant. One or more plants are visited.

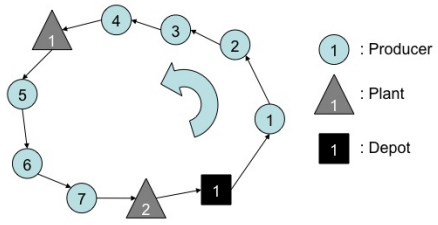

Figure 3.3: General configuration of interlaced multi-drop route

We consider simple routes, and Section 3.4 gives the details of this subclass of the problem.

3.3 Literature review

In this section, we review research into different variants of the MAVRP. We focus on exact algorithms rather than heuristic methods and consider variants of the VRP with attributes similar to those of our problem.

Among the variants of the VRP, the VRPTW has received the most attention, and numerous researchers have applied column generation methodology. For the VRPTW, column generation was first used by Desrochers et al. [49] in a Dantzig–Wolfe decomposition framework. They devised a branch-and-bound algorithm to solve a number of original time-window constrained problems from Solomon [145] to optimality or near optimality. Kohl et al. [95] improved the method by adding 2-path inequalities to the LP relaxation of the set partitioning formulation. Kohl and Madsen [94] proposed a branch-and-bound algorithm in which subgradient and bundle methods were employed to compute the lower bounds. These methods were based on 2-cycle elimination algorithms. Irnich and Villeneuve [87] proposed a branch-and-price algorithm in which the subproblem is solved using a k-cycle elimination procedure. Branch-and-price has been the leading methodology for the VRPTW since the beginning of the 1990s.

Feillet et al. [55] improved the extension of the Ford–Bellman algorithm proposed by M. [110]. More precisely, they improved the labeling procedure for the elementary shortest path problem with resource constraints (ESPPRC), which is the backbone of a number of solution procedures based on column generation, by proposing new labels and dominance rules. Righini and Salani [132] proposed an improved bounded bidirectional label-correcting algorithm in which two sequential labeling processes starting from the depot and a copy of the depot (considered the sink node) cooperate to accelerate the solution of the ESPPRC.

The most efficient algorithms for the ESPPRC are based on a partial or complete relaxation of the elementarity condition. Boland et al. [23] and Righini and Salani [133] embedded a decremental state space relaxation (DSSR) scheme into the labeling procedure. In this method, the elementarity condition on the generated routes is initially relaxed, transforming the problem into a shortest path problem with resource constraints (SPPRC). After each iteration, using an augmentation strategy, restrictions are added to the problem to prevent the formation of cycles. Several state-space augmentation strategies were evaluated by Boland et al. [23]. Later, Desaulniers et al. [48] used heuristic dynamic programming and a tabu search (TS) heuristic to rapidly generate routes with negative reduced costs. The dynamic heuristic is based on making the graph more sparse by eliminating arcs that do not seem promising and applying aggressive dominance rules (relaxed conditions). Their method outperformed all previous algorithms in terms of the computational time. Moreover, they successfully solved 5 of 10 Solomon instances not previously solved.

Baldacci et al. [13] introduced a new state-space relaxation, called the *ng*-path relaxation, to compute lower bounds for routing problems such as the CVRP and the VRPTW. This relaxation partitions the set of all possible paths ending at a generic vertex. This is done according to prespecified neighborhoods of graph vertices, and a mapping function associates with each path a subset of the vertices that depends on the order in which these vertices are visited. These subsets of vertices are used to impose partial elementarity. This relaxation proved particularly effective in computing lower bounds for the CVRP, the VRPTW, and the traveling salesman problem with time windows (TSPTW). Baldacci et al. [14] proposed a new dynamic programming method to improve the *ng*-path relaxation. It iteratively defines the mapping function of the *ng*-path relaxation using the results from the previous iteration. This method is analogous to cutting plane methods, where the cuts violated by the *ng*-paths at a given iteration are incorporated into the new *ng*-path relaxation at the next iteration.

Martinelli [111] proposed an efficient *ng*-route pricing in which a DSSR technique is embedded into the *ng*-route relaxation. It consists of an *ng*-route relaxation procedure in which resources associated with the vertices' neighbors are initially deactivated. These neighborhoods are iteratively augmented using a DSSR scheme to ensure the *ng*-feasibility of all the columns.

A unified exact method capable of solving different classes of the VRP, including the multi-depot heterogeneous vehicle routing problem (MDHVRP), was proposed by Baldacci and Mingozzi [11]. It is based on the solution of an integer linear programming problem and on dual heuristics. It can solve instances with up to 100 customers to optimality; this takes several hours. Finally, Bettinelli et al. [22] proposed a branch-and-cut-and-price algorithm for the MDHVRPTW. The method allows for different combinations of cutting and pricing strategies, and both heuristic and exact approaches are proposed for the subproblems.

To summarize, we make the following observations:

- Our literature review supports our claim about the novelty of the problem con-

sidered in this paper. To the best of our knowledge, this variant has not been previously studied. The plant-assignment phase is more complex than in other variants.

- Many efficient techniques have been developed for classes of the VRP with features similar to those of our variant. Our algorithm is based on a specialization of a cutting-edge branch-and-price algorithm, and it incorporates techniques from the literature.
- We evaluate the efficiency and relevance of these techniques in the context of our problem.

3.4 Milk collection problem

We consider a deterministic subclass of the general problem, which is a real-world tactical planning problem in the context of milk collection. We consider two variants:

1. The depot associated with each producer is preassigned based on contractual agreements.
2. We remove the preassignments; this is a logical extension of the first variant. We claim that slight modifications in the data set can reduce variant 2 to variant 1.

We first consider variant 1 and in Section 3.5 we show how to adapt the approach for variant 2. We assume that the vehicles perform simple routes as described in Section 3.2, and collections and deliveries are made once a day. The problem is therefore a multi-depot vehicle routing problem with time windows, heterogeneous vehicle fleets, plant deliveries, and producer-depot preassignments.

Several carriers, based in different depots, collect milk from farms in a specific geographical region and deliver it to milk-processing plants. The model is defined on a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, where $\mathcal{V}$ and $\mathcal{A}$ are the node and arc sets, respectively. The node set contains the depots, producers, and plants: $\mathcal{V} = \mathcal{D} \cup \mathcal{N} \cup \mathcal{U}$ where $\mathcal{D} = (1, \ldots, d)$ represents the depot set, $\mathcal{N} = (1, \ldots, n)$ the producer set, and $\mathcal{U} = (1, \ldots, u)$ the plant set. The arc set $\mathcal{A} \subset \mathcal{V} \times \mathcal{V}$ defines feasible movements between different locations in $\mathcal{V}$. Associated with each arc (i, j) is a transportation cost c_{ij} that is proportional to the travel time between locations i and j. Each carrier has one or more vehicle types, and $\mathcal{K} = (1, \ldots, k)$ is the set of vehicle types. The capacity, the fixed cost, and the variable cost coefficient of the kth vehicle type are Q_k, c_k, and w_k, respectively. More precisely, w_k is the cost for vehicle type $k \in \mathcal{K}$ to travel one unit of distance. Associated with each plant is a daily demand, D_u, and we assume that there is sufficient supply to meet the demand.

We introduce a path-based formulation that yields a set partitioning model. Let $\mathcal{P}_{du}^k$ be the set of feasible routes from depot $d \in \mathcal{D}$ to plant $u \in \mathcal{U}$ operated by vehicle

Table 3.1: Notation for the set partitioning formulation

Notation	Description
	Parameters
a_{ip}	1 if producer i is visited on route p.
D_u	Daily demand of plant u.
c_k	Fixed cost of the vehicle type k.
c_p	Variable cost of route p.
g_p	Collected quantity via route p.
	Variables
y_p	1 if route p is selected in the optimal solution.

type $k \in \mathcal{K}_d$. Each route $p \in \mathcal{P}_{du}^k$ can serve only the producers assigned to depot d, and $\mathcal{C}_d$ is this set of producers. Let y_p be a binary variable such that y_p is 1 if route p is selected in the optimal solution and 0 otherwise. The quantity collected on route p, g_p, cannot exceed the capacity of the vehicle; g_p is 0 for all plants not visited on route p. Parameter a_{ip} is 1 if producer i is visited on route p and 0 otherwise. The variable cost of each route $p \in \mathcal{P}_{du}^k$ is c_p; it is the sum of the arc costs of the route. The path-based model is as follows:

$$\min \quad \sum_{u\in\mathcal{U}} \sum_{d\in\mathcal{D}} \sum_{k\in\mathcal{K}_d} \sum_{p\in\mathcal{P}_{du}^k} (c_p + c_k) y_p \tag{3.1}$$

subject to

$$\sum_{u\in\mathcal{U}} \sum_{d\in\mathcal{D}} \sum_{k\in\mathcal{K}_d} \sum_{p\in\mathcal{P}_{du}^k} a_{ip} y_p = 1 \quad (i \in \mathcal{N}); \tag{3.2}$$

$$\sum_{d\in\mathcal{D}} \sum_{k\in\mathcal{K}_d} \sum_{p\in\mathcal{P}_{du}^k} g_p y_p \geq D_u \quad (u \in \mathcal{U}); \tag{3.3}$$

$$y_p \in \{1, 0\} \quad (d \in \mathcal{D}; k \in \mathcal{K}_d; u \in \mathcal{U}; p \in \mathcal{P}_{du}^k). \tag{3.4}$$

Constraint (3.2) ensures that each producer is visited exactly once by exactly one route, and constraint (3.3) guarantees that the plant demands are satisfied. In the following sections, we describe our algorithm in detail.

3.5 Solution method

In this section, we present our algorithm. In the path-based integer model (3.1)–(3.4), the number of paths is so large that it is not practical to solve the model directly using an MIP solver. Thus, the usual solution method is based on the branch-and-price algorithm. This is a branch-and-bound algorithm where the lower bounds are computed using column generation (for a complete survey of column generation methods, see Lübbecke and Desrosiers [109]). Column generation is often successful when the associated integer programs are set partitioning (or set covering) problems. In the VRP, each variable of the set partitioning formulation represents a feasible route. However,

most successful decomposition approaches for the VRP formulate the pricing problem as an ESPPRC. At each iteration of the column generation, a restricted linear master problem (RLMP) is solved rather than the master problem itself. The columns in the RLMP are limited to those that have already been generated in the pricing problem.

The search tree is initialized at the root node. Initialization involves adding to the RLMP sufficient columns to obtain a feasible solution. At each node of the search tree, the RLMP contains a subset of the feasible columns, already priced out by the subproblem, which are in compliance with the branching decisions. It is solved by column generation. If the solution is integer, it is a valid solution to the original master problem, and it is compared to the incumbent solution. Otherwise, branching occurs to eliminate the current fractional point. In other words, when no column is available to enter the basis but the solution of the linear relaxation is not integer, branching occurs. A valid branching scheme will eliminate the current fractional solution, produce a balanced search tree, and keep the structure of the problem unchanged. At the end of the search process, the best integer solution found is the optimal solution for the original problem.

As mentioned in Section 3.4, variant 2 has no preassignments. The decision about the depot associated with each producer is made during the solution process. A slight modification allows our algorithm to solve this more general problem: we set $\mathcal{C}_d = \mathcal{N}$ for each depot $d \in \mathcal{D}$. Variant 2 may be useful for proposing a first set of assignments or revising an existing set in a strategic planning phase.

In Sections 3.5.1–3.5.4, we describe the branch-and-price algorithm.

3.5.1 Master problem

A relaxation of integrality constraints (3.8) yields the linear master problem. The RLMP, which is restricted to a subset of columns $\mathcal{P}'^{k}_{du} \subset \mathcal{P}^{k}_{du}$, takes the following form:

(RLMP)

$$\min \quad \sum_{u\in\mathcal{U}}\sum_{d\in\mathcal{D}}\sum_{k\in\mathcal{K}_d}\sum_{p\in\mathcal{P}'^{k}_{du}} (c_p + c_k)y_p \tag{3.5}$$

subject to

$$\sum_{u\in\mathcal{U}}\sum_{d\in\mathcal{D}}\sum_{k\in\mathcal{K}_d}\sum_{p\in\mathcal{P}'^{k}_{du}} a_{ip}y_p = 1 \quad (i \in \mathcal{N}); \tag{3.6}$$

$$\sum_{d\in\mathcal{D}}\sum_{k\in\mathcal{K}_d}\sum_{p\in\mathcal{P}'^{k}_{du}} g_p y_p \geq D_u \quad (u \in \mathcal{U}); \tag{3.7}$$

$$y_p \geq 0 \quad (d \in \mathcal{D}; k \in \mathcal{K}_d; u \in \mathcal{U}; p \in \mathcal{P}'^{k}_{du}) \tag{3.8}$$

where constraint (3.6) ensures at least one visit to each producer, and constraint (3.7) guarantees that the plant demands are satisfied. It is worth mentioning that contrary to the case of most problems formulated using the set partitioning formulation, in our

problem a transformation from the set partitioning formulation to the set covering scheme does not necessarily result in the same optimal solution. In fact the presence of constraints (3.7) may result in multiple visits to some producers. Accordingly, constraints (3.6) are kept under equality form.

3.5.2 Initialization

To obtain the first set of dual variables of the master problem, we add two sets of initial columns. The first set consists of routes that start and end at a given depot and visit one producer and one plant for all possible combinations of depots, producers, and plants. The second set is generated using the classical savings heuristic of Clarke and Wright [32]. For each depot-vehicle pair, the method starts with $|\mathcal{C}_d|$ routes, each serving a single producer and starting and ending at the depot. It then computes the cost reduction achieved by combining two routes (by connecting the end point of one to the end point of the other) via the following equation:

$$saving_{ij} = c_{0i} + c_{0j} - c_{ij} \tag{3.9}$$

where i and j represent the two connected end points. Here, the end point is defined to be the first or last producer in the route. The heuristic greedily selects the maximum saving and combines the associated routes provided the constraints on time windows and vehicle capacities are not violated. When no routes can be merged, the algorithm terminates by assigning the routes obtained to the members of the plant set.

3.5.3 Pricing problem

The pricing problem aims to find one or more master problem variables p with a negative reduced cost with respect to a given dual solution of the linear relaxation of the master problem. In our column generation approach, the pricing problem is decomposed into several similar subproblems. Each subproblem is an ESPPRC associated with a specific depot, plant, and vehicle type, where the set of resource constraints contains time windows and vehicle capacities. Consider the following dual variables of the RLMP (3.5)–(3.8):
λ_i: free-signed dual variable of (3.6) for producer $i \in \mathcal{N}$;
μ_u: nonnegative dual variable of (3.7) for plant $u \in \mathcal{U}$.
Let x_{ij} be a binary decision variable that is 1 if vertex v_j follows v_i on the shortest path, and 0 otherwise. Variable t_i is the time at which the service starts at vertex i if the shortest path visits this node. Binary variable f_u is 1 if the shortest path visits plant u, and 0 otherwise. A supply q_i is associated with each producer i. Each producer has a time window $[e_i, l_i]$ during which the service may occur. The node 0_d represents the depot and the node $0'_d$ represents a copy of the depot that plays the role of a fictitious sink node in the standard form of the shortest path problem.

Using this notation, the pricing problem for a vehicle type k, which leaves depot

Table 3.2: Notation for the pricing problem

Notation	Description
	Parameters
q_i	Supply of producer i.
$[e_i, l_i]$	Time window of producer i.
Q_k	Capacity of the vehicle type k.
c_k	Fixed cost of the vehicle type k.
c_{ij}	Length of arc (i, j).
w_k	Variable cost coefficient of the vehicle type k.
λ_i	Dual variable of (3.6) for producer $i \in \mathcal{N}$.
μ_u	Dual variable of (3.7) for plant $u \in \mathcal{U}$.
	Variables
x_{ij}	1 if vertex v_j follows v_i on the shortest path.
t_i	Time at which the service starts at vertex i.
f_u	1 if the shortest path visits plant u.

$0_d, d \in \mathcal{D}$, and services the producers of the set $\mathcal{C}_d$, is as follows:

$$\min \quad \sum_{i \in \mathcal{C}_d} \sum_{j \in \mathcal{C}_d} (w_k c_{ij} - \lambda_i) x_{ij} - \sum_{u \in \mathcal{U}} g_p \mu_u + c_k \tag{3.10}$$

subject to

$$\sum_{i \in \mathcal{C}_d} x_{ih} - \sum_{j \in \mathcal{C}_d} x_{hj} = 0 \quad (h \in \mathcal{C}_d); \tag{3.11}$$

$$\sum_{j \in \mathcal{C}_d} x_{0_d j} = 1; \tag{3.12}$$

$$f_u - \sum_{i \in \mathcal{C}_d} x_{iu} = 0 \quad (u \in \mathcal{U}); \tag{3.13}$$

$$\sum_{u \in \mathcal{U}} f_u = 1; \tag{3.14}$$

$$f_u - x_{u0'_d} = 0 \quad (u \in \mathcal{U}); \tag{3.15}$$

$$\sum_{i \in \mathcal{C}_d} \sum_{u \in \mathcal{U}} x_{ui} = 0; \tag{3.16}$$

$$x_{ij}(t_i + s_i + t_{ij} - t_j) \leq 0 \quad (i \in \mathcal{C}_d; j \in \mathcal{C}_d \cup \mathcal{U}); \tag{3.17}$$

$$e_i \leq t_i \leq l_i \quad (i \in \mathcal{C}_d \cup \mathcal{U}); \tag{3.18}$$

$$g_p \leq \sum_{i \in \mathcal{C}_d} q_i \sum_{j \in \mathcal{C}_d} x_{ij} \leq Q_k; \tag{3.19}$$

$$x_{ij} \in \{1, 0\} \quad (i, j \in \mathcal{C}_d); \tag{3.20}$$

$$f_u \in \{1, 0\} \quad (u \in \mathcal{U}). \tag{3.21}$$

Constraints 3.11–3.16 are flow constraints that result in a path from the depot 0_d to $0'_d$ ensuring that the shortest path visits a plant before returning to the depot. Constraints 3.17 and 3.18 are time-window constraints. Constraint 3.19 is the capacity constraint, and Constraints 3.20 and 3.21 ensure integrality.

The subproblems above are variants of the ESPPRC and thus are NP-hard problems in the strong sense [53]. To reduce the number of iterations to solve this complex problem to optimality, one may try to generate multiple negative-reduced-cost columns using fast heuristics. However, when the heuristic procedures fail to find a new column, we must perform at least one iteration of the exact procedure to prove the optimality of the lower bound. We solve the subproblems with a bilevel column generation procedure. The first level consists of a procedure based on heuristic dynamic programming (HDP), and it is followed by exact dynamic programming (EDP). We describe these modules below. We first summarize the EDP and then describe the heuristic strategies that speed up the procedure. Finally, the bilevel procedure is presented in Algorithm 1, which shows how these modules interact.

5.3.1 Exact dynamic programming (EDP)

Classical dynamic programming algorithms start from an initial label associated with the depot and extend the labels along arcs using extension functions. To avoid creating too many labels, dominated labels are eliminated by a dominance procedure. As described in Section 3.3, much research has focused on the computational efficiency of the labeling procedure for the ESPPRC subproblem. DSSR [133, 23] and the *ng*-route relaxation [13] have received the most attention.

DSSR can be considered a special case of the *ng*-route relaxation. However, we consider DSSR as a stand-alone procedure and DSSR embedded into the *ng*-route relaxation [111] as different strategies for the pricing problems. In both cases, the elementarity relaxation of the ESPPRC allows the generation of paths with cycles throughout the labeling procedure. The relaxation is iteratively tightened by considering new resources that forbid cycles. However, in the *ng*-route relaxation (but not DSSR), the *ng*-feasible columns may still contain cycles. Therefore, the lower bound obtained using the *ng*-route relaxation can be weaker than the DSSR bound, representing the optimal lower bound.

Decremental state-space relaxation (DSSR)

This relaxation of the ESPPRC allows the generation of paths with cycles throughout the labeling procedure. The relaxation is iteratively tightened by considering some nodes to be critical and forbidding multiple visits to them. Critical nodes are selected based on an augmentation policy, from those nodes involved in a cycle in at least one route. If at the end of a labeling iteration, the paths contain no cycles, the solution is valid for the ESPPRC. Otherwise, the relaxed state space is augmented by one or more resources associated with newly recognized critical nodes and the procedure restarts. At the end of each iteration of the labeling algorithm, the nodes with visit multiplicity greater than one on the lowest cost path are recognized as new critical nodes.

In our implementation, each label $\sigma = (C, T, L, \widehat{S}, \phi)$ has a component C to repre-

sent the reduced cost of the partial path, a resource T for the time, a resource L for the vehicle load, a resource $\widehat{S}$ for the number of unreachable critical nodes, and a set $\phi \subseteq \Phi$ that contains the critical nodes unreachable from the current label, where Φ represents the set of all the recognized critical nodes at a given state of the procedure. At the end of each iteration of the SPPRC, a state-space augmentation policy defines which nodes should be added into Φ, and a resource associated with each of the critical nodes is added into the resource set to prevent cycling on that node.

For $\sigma_1 = (C_1, T_1, L_1, \widehat{S}_1, \phi_1)$ and $\sigma_2 = (C_2, T_2, L_2, \widehat{S}_2, \phi_2)$ two labels corresponding to two partial paths from a depot to a given node, we say that σ_1 dominates σ_2 if the following criteria are met:

(a_{DSSR}) $T_1 \leq T_2$,

(b_{DSSR}) $L_1 \leq L_2$,

(c_{DSSR}) $C_1 \leq C_2$,

(d_{DSSR}) $C_1 - \mu_{u*}(L_2 - L_1) \leq C_2$, where $u* = arg\max_{u \in \mathcal{U}}\{\mu_u\}$,

(e_{DSSR}) $\widehat{S}_1 \leq \widehat{S}_2$,

(f_{DSSR}) $\phi_1 \subseteq \phi_2$.

Condition (d_{DSSR}) is used to prevent the dominance of partial paths that appear to be dominated by other paths with respect to the conditions (a_{DSSR})–(c_{DSSR}) and (e_{DSSR})–(f_{DSSR}) in a producer node, but that later become less costly by delivering more product to a plant.

ng-Route Decremental State-Space Relaxation (*ng*R-DSSR)

The *ng*-route relaxation [13], originally proposed for the CVRP and the VRPTW, provides a good compromise between obtaining good lower bounds and efficiently pricing routes that are not necessarily elementary.

The *ng*-route relaxation can be described as follows. Suppose that $\mathcal{V}_{rd}$ represents the set of producers visited by partial path r starting from depot d. Moreover, for each customer $i \in \mathcal{C}_d$, let $\mathcal{N}_i \subseteq \mathcal{C}_d$, the so-called original neighborhood of producer i, represent a set (with an a priori fixed size) of producers, selected according to a neighborhood criterion for producer i. For label σ, associated with a given partial path $r = (d, i_1, \ldots, i_n)$, we define a set $\Pi(r) \subseteq \mathcal{V}_{rd}$, containing all prohibited extensions from producer i_n. The set $\Pi(r)$ is

$$\Pi(r) = \{i_j \in \mathcal{V}_{rd} | i_j \in \bigcap_{k=j+1}^{n} \mathcal{N}_{i_k}, j = 1, \ldots, n-1\} \cup \{i_n\}. \tag{3.22}$$

Consequently, each label $\sigma = (C, T, L, S_{ng}, \Pi)$ has new members S_{ng}, representing the size of the Π set, where Π represents the set of inaccessible producers according to the ng-rules. Again, to reduce the number of possible labels, a dominance rule is incorporated into the algorithm.

Given two labels $\sigma_1 = (C_1, T_1, L_1, S_{ng1}, \Pi_1)$ and $\sigma_2 = (C_2, T_2, L_2, S_{ng2}, \Pi_2)$, representing two partial paths ending at a given producer, label σ_1 dominates label σ_2 if and only if any possible extension from σ_1 is feasible from label σ_2 with a lower reduced cost. This condition is satisfied if the following criteria are met:

(a_{ng}) $T_1 \leq T_2$,

(b_{ng}) $L_1 \leq L_2$,

(c_{ng}) $C_1 \leq C_2$,

(d_{ng}) $C_1 - \mu_{u*}(L_2 - L_1) \leq C_2$, where $u* = arg\max_{u \in \mathcal{U}}\{\mu_u\}$,

(e_{ng}) $S_{ng1} \leq S_{ng2}$,

(f_{ng}) $\Pi_1 \subseteq \Pi_2$.

ng-route decremental state-space relaxation (ngR-DSSR) consists of an ng-route relaxation procedure in which initially empty sets $\widehat{\mathcal{N}_i} \subseteq \mathcal{N}_i$, called applied neighborhoods, are considered rather than the original neighborhoods $\mathcal{N}_i$. At the end of each iteration, all columns with negative reduced costs and no cycles as well as those that satisfy the ng-rules with respect to the original neighborhoods (called ng-feasible columns) are added to the RLMP. If the best column according to its reduced cost is not ng-feasible, some of the applied neighborhoods are augmented and the procedure restarts. Two augmentation strategies are considered:

1. At the end of an iteration, the nodes that violate the ng-rules on the best columns are recognized as critical. Newly recognized critical node i is then added into the applied neighborhoods of all other nodes that consider i as their neighbor, according to their original neighborhoods.

2. Here we augment the applied neighborhoods of only those nodes forming a cycle involving i, when the ng-rules are violated, by adding i into these neighborhoods.

Our experiments showed that the second strategy is more efficient. The smaller sets of applied neighborhoods make the dominance easier by more easily satisfying condition (f_{ng}). Note that in our implementation, $\widehat{\mathcal{N}_i}$ is initialized (set to $\emptyset$) at the root node and not elsewhere in the search tree. This is because of the high likelihood of the recreation of cycles that violate the ng-rules, if $\widehat{\mathcal{N}_i}$ is reset to $\emptyset$ at each node of the tree; this is equivalent to extra iterations to augment the applied neighborhoods to ensure ng-feasibility.

5.3.2 heuristic dynamic programming (HDP)

To speed up the generation of the negative-reduced-cost columns, we implement a relaxed version of the labeling procedure described above. The relaxations are based on weakening the dominance rules (reducing the number of conditions tested) so that a larger number of labels are discarded. This may result in the generation of some but not all of the existing negative-reduced-cost paths, in a shorter computational time. We accelerate the labeling procedure by ignoring the dominance conditions corresponding to the comparison of unreachable nodes. For the DSSR, this is done by relaxing conditions (e_{DSSR}) and (f_{DSSR}), while for the ngR-DSSR, (e_{ng}) and (f_{ng}) are ignored. This harsh dominance accelerates the labeling process by extending a smaller set of labels from each node and by comparing new labels to a shorter list of existing labels.

5.3.3 Description of the bilevel column generation procedure

To schematically show how the column generators cooperate within our algorithm, we introduce the following notation:

$NBCOL_{HDP}$**:** Number of negative-reduced-cost columns generated by HDP.

$NBCOL_{EDP}$**:** Number of negative-reduced-cost columns generated by EDP.

We now present the procedure which, at each iteration, finds the non-dominated paths and adds them to the RLMP.

Algorithm 1 Solution of bilevel subproblem

repeat
  repeat
    $NBCOL_{HDP} = 0$
    HEURISTIC DYNAMIC PROGRAMMING()
    Update $NBCOL_{HDP}$
    Update and Solve the RLMP
  until $NBCOL_{HDP} == 0$
  $NBCOL_{EDP} = 0$
  EXACT DYNAMIC PROGRAMMING( )
  Update $NBCOL_{EDP}$
  Update and Solve the RLMP
until $NBCOL_{EDP} == 0$.

Clearly, the difficulty of this algorithm depends on the size of the problem: the number of depots, plants, producers, and vehicle types. The difficulty is also affected by the tightness of the time window and vehicle capacity constraints.

As mentioned in Section 3.4, variant 2 has no preassignments. The following modification to the algorithm for variant 1 makes it applicable to variant 2: we solve the ESPPRC for a depot, a specific vehicle type, and for the entire set of producers instead of a preassigned subset.

3.5.4 Branching strategy

As mentioned in Section 3.5, we find an integer solution via a branch-and-price algorithm. In the literature, binary branching strategies, which divide a problem into two more restricted problems, have been proposed for the VRPTW. Branching must be performed at each node where the optimal solution to the linear relaxation includes fractional path variables. The classical branching strategy is branching on the flow variables, i.e., $\sum_k x_{ij}^k$, where x_{ij}^k represents the flow on arc (i, j) for vehicle k. This results in two new nodes in the tree, one with the new constraint $\sum_k x_{ij}^k = 0$ and the other with $\sum_k x_{ij}^k = 1$. The advantage of this strategy is that the added constraints are easily integrated into both the master and pricing problems. Moreover, it finds an optimal integer solution if such a solution exists. However, this approach is not efficient enough to obtain integer solutions rapidly. In other words, the elimination of one arc from the graph via the branching constraint (especially the constraint $\sum_k x_{ij}^k = 0$) may have little effect on the solution and does not necessarily decrease the complexity of the problem [62].

To overcome this weakness, we study two bilevel branching procedures. In each case, the procedure is followed by branching on flow variables. We describe these procedures below.

Branching by plant assignment (BPA)

The special structure of our problem allows us to derive efficient new constraints through a branching scheme. Since there are multiple plants to which the products of a specific producer can be delivered, we can assign producers to plants via the branching procedure. Since producers are preassigned to depots, this strategy attempts to divide the problem into several smaller problems, each containing one depot, one plant, and a limited number of producers. We branch on the flow variables when there are no more producer-plant candidates for branching. The producers that are not permitted to serve a plant because of branching decisions are removed from the subgraph associated with the plant. We branch on the producer-plant candidate with flow closest to 0.5. This flow is obtained by summing the basic variables of the master problem associated with the routes containing a given producer and a given plant. The removal of a producer from the subgraph associated with a plant is much more restrictive than the elimination of a single arc. Therefore, we expect this strategy to be more effective than branching on the flow variables.

Branching on time windows (BTW)

This binary branching strategy, originally proposed by Gélinas et al. [62] for the VRPTW, splits the time window of a node into two new subintervals; each branch corresponds to one of the subintervals. Some routes become infeasible following a split in a producer's time window. Gélinas et al. [62] claimed that this strategy is stronger than branching on flow variables since constraints such as time and capacity have a major impact on the difficulty of the VRPTW.

3.6 Computational results

We have proposed different options for column generation and two branching strategies. To evaluate the performance of these approaches, we carried out a series of computational experiments, and we report the results in this section. First, we describe the creation of a large set of randomly generated instances for our tests. Then, we discuss the efficiency of DSSR and ngR-DSSR. Next, we compare the two branching strategies, BPA and BTW.

We ran the experiments on a computer with a 2.67 GHz processor and 24 GB of RAM. The algorithms were implemented in C++ and the linear models were solved using Cplex 12.2.

3.6.1 Test problems

Since, to the best of our knowledge, there is no prior study of the multi-depot vehicle routing problem with time windows and deliveries to plants, we generated new test problems. We considered narrow and wide time windows, where the wide windows are on average twice as wide as the narrow windows. We also considered different plant locations on the graph.

In the case that we call *inside plants*, the depots and plants are randomly located in a $(-50, 50)^2$ square, according to a continuous uniform distribution. The producers are randomly located in a $(-100, 100)^2$ square; they are placed one by one via a generation-validation procedure. Suppose that v_i is the current producer, min_d is the distance from v_i to its closest depot, and z is a number in the interval $[0, 1]$ chosen according to a continuous uniform distribution. This producer is retained if $z < exp(-h.min_d)$ where $h = 0.05$, and otherwise is dropped. The application of this probabilistic function, inspired by Cordeau et al. [34], increases the likelihood of producer clusters around the depots.

In the case that we call *outside plants*, the plant locations are randomly generated in the area beyond the region containing the producers: $(-150, 150)^2 - (-100, 100)^2$, where the producers are located randomly in a $(-100, 100)^2$ square via a new generation-validation procedure. We retain producers satisfying $z < exp(-h(min_d.\alpha + min_p(1 -$

α))) where min_d and min_p are the distances from the newly generated producer to the closest depot and the closest plant, respectively. Moreover, z and α are two uniform random numbers that are respectively generated in $[0, 1]$ and $[0.3, 0.7]$. Once again, this probabilistic function leads to clusters of producer nodes in the region between the plants and the depots. Figure 3.4 shows an example of an instance with three depots, three outside plants, and 100 producers.

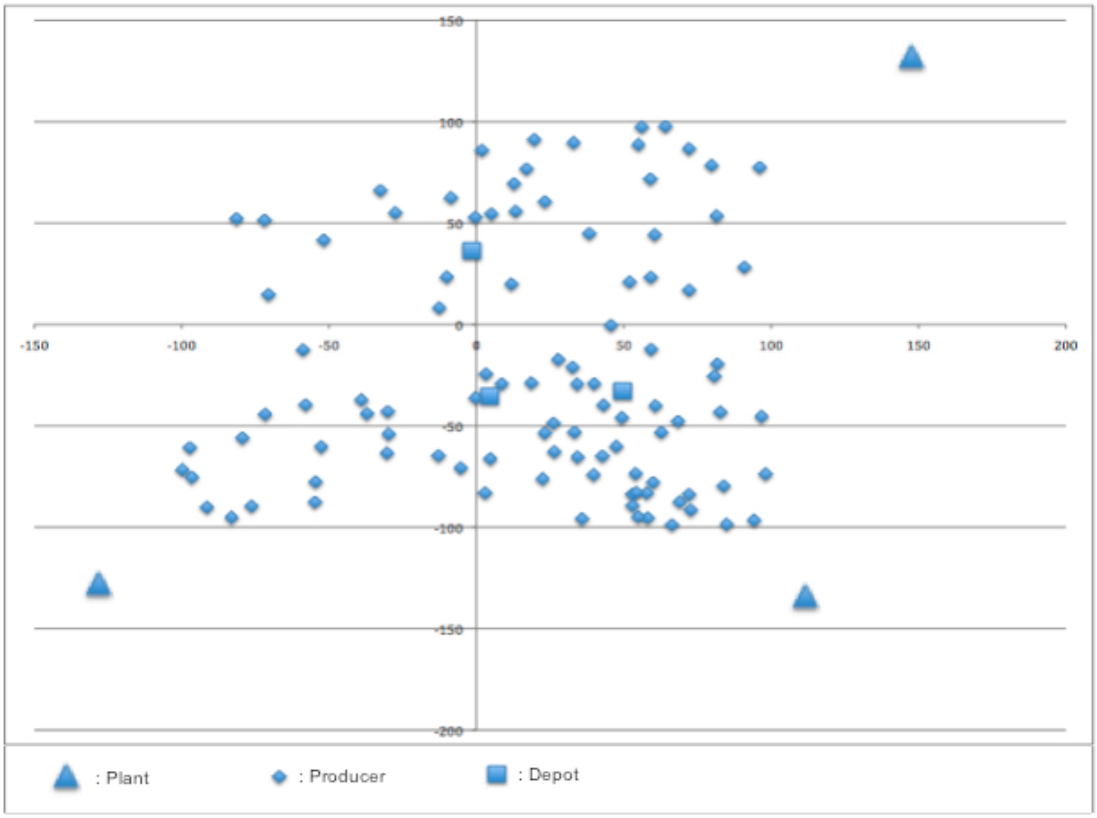

Figure 3.4: Producer locations in case with three outside plants

We use the Euclidean distance between two nodes. The preassignments of producers to depots for variant 1 are done one by one in numerical order: we greedily assign each producer to its closest depot while trying to ensure that each depot has the same number of producers.

The service duration and the quantity supplied by each producer are randomly and independently chosen according to a discrete uniform distribution on [1, 25]. To increase the probability of feasible instances, we set the sum of the plant demands to 90% of the total supply available.

Table 3.3 shows the characteristics of the four problem classes. The size of each instance is determined by the number of depots, producers, and plants; the values that we considered are presented in Table 3.4. Instances with the same number of depots and plants have those facilities in the same positions. We generated five instances for

each size combination of each problem class. For example, instance pr04-50-2D3P-5 represents the fifth instance of the fourth class with fifty producers, two depots, and three plants.

Table 3.3: Four problem classes

Class Number	Plant Location	Time Windows
pr01	inside	narrow
pr02	inside	wide
pr03	outside	narrow
pr04	outside	wide

Table 3.4: Specifications of test problems

Number of depots	Number of producers	Number of plants
2	$30, 40, 50$	$2, 3$
3	$30, 40, 50$	$2, 3$

3.6.2 Linear relaxation

To evaluate the efficiency of our column generation, we ran a group of tests for a set of problems with fifty producers, representing the most difficult instances. We considered one instance from each group of five for a given size combination of each class, forming a group of sixteen instances. We solved the root linear relaxation using either DSSR or ngR-DSSR with two different neighborhood sizes, $|\mathcal{N}_i| = 5$ and $|\mathcal{N}_i| = 8$ for each producer $i \in \mathcal{N}$. Table 3.5 gives the results both with and without HDP. The pairs in the "nb. Iter." column give the number of heuristic and exact column-generation iterations. Column "T" gives the computational time (in seconds) to solve the linear relaxation.

Table 3.5: Results for solution of linear relaxation for instances with fifty producers

	EDP_{DSSR}		$HDP_{DSSR} + EDP_{DSSR}$		EDP_{ng5}		EDP_{ng8}		$HDP_{ng5} + EDP_{ng5}$		$HDP_{ng8} + EDP_{ng8}$	
Class Number	T	nb. iter.	T	nb. iter.	T	nb. iter.	T	nb. iter.	T	nb. iter.	T	nb. iter.
pr01-50-2D2P	5.5	(0, 18)	2.4	(21, 1)	7.2	(0, 18)	7.0	(0, 18)	2.7	(17, 1)	2.5	(17, 1)
pr01-50-2D3P	9.1	(0, 14)	6.3	(19, 5)	10.7	(0, 14)	10.6	(0, 14)	7.0	(16, 5)	6.8	(16, 5)
pr01-50-3D2P	6.3	(0, 26)	2.3	(29, 1)	10.9	(0, 28)	10.1	(0, 28)	3.9	(33, 1)	3.9	(33, 1)
pr01-50-3D3P	12.2	(0, 16)	5.3	(17, 1)	16.1	(0, 15)	17.2	(0, 15)	6.6	(15, 1)	6.1	(16, 1)
pr02-50-2D2P	23.2	(0, 30)	9.4	(31, 3)	39.5	(0, 36)	45.8	(0, 26)	15.6	(33, 7)	18.4	(29, 6)
pr02-50-2D3P	95.8	(0, 15)	73.9	(24, 5)	119.8	(0, 16)	147.6	(0, 18)	101.7	(16, 6)	98.1	(17, 6)
pr02-50-3D2P	118.5	(0, 18)	65.3	(28, 4)	240.3	(0, 20)	225.8	(0, 23)	103.5	(21, 4)	88.7	(23, 5)
pr02-50-3D3P	394.8	(0, 13)	269.2	(16, 3)	570.2	(0, 13)	643.1	(0, 14)	386.1	(15, 4)	317	(15, 3)
pr03-50-2D2P	1.4	(0, 17)	0.9	(16, 1)	2.5	(0, 18)	2.1	(0, 18)	1.2	(15, 1)	1.1	(15, 1)
pr03-50-2D3P	2.0	(0, 15)	1.2	(18, 1)	2.8	(0, 14)	2.5	(0, 14)	1.3	(15, 1)	1.2	(15, 1)
pr03-50-3D2P	1.6	(0, 15)	1.0	(21, 1)	2.8	(0, 18)	2.4	(0, 17)	1.3	(17, 1)	1.4	(17, 1)
pr03-50-3D3P	1.0	(0, 14)	0.8	(15, 1)	2.0	(0, 13)	1.6	(0, 13)	1.0	(15, 1)	0.9	(15, 1)
pr04-50-2D2P	23.2	(0, 20)	23.8	(34, 8)	38.0	(0, 23)	30.7	(0, 21)	22.2	(26, 7)	26	(27, 9)
pr04-50-2D3P	3.6	(0, 11)	3.1	(13, 2)	6.6	(0, 14)	6.6	(0, 14)	5.0	(15, 4)	5.3	(15, 4)
pr04-50-3D2P	23.1	(0, 18)	13.8	(23, 3)	40.8	(0, 23)	38.1	(0, 21)	21.2	(23, 5)	22.7	(24, 6)
pr04-50-3D3P	21.1	(0, 13)	21.1	(20, 5)	30.5	(0, 14)	29.6	(0, 14)	23.3	(13, 6)	23	(13, 5)
Average	46.4	(0, 17)	31.2	(22, 3)	71.3	(0, 19)	76.3	(0, 18)	44.0	(19, 3)	38.9	(19, 4)

On average the use of HDP improves the computational time by decreasing the

number of calls to EDP. We also studied the use of metaheuristics to generate columns; we implemented a method based on TS. This approach, inspired by the procedure of Desaulniers et al. [48], attempts to generate new negative-reduced-cost columns from the set of existing columns. However, our experiments showed that it did not improve the computational time. Our results support those reported by Desaulniers et al. [48]. However, our results for instances with 100 and 200 nodes show that TS is more efficient for larger instances and longer routes.

3.6.3 Branching and integer solution

As mentioned in Section 3.5, a branching scheme is often necessary. We now evaluate the performance of the two branching strategies introduced in Section 3.5.4. We present the results for three approaches. All three use EDP and HDP, because they decrease the average computational time. The first method uses DSSR, the second uses ngR-DSSR with $|\mathcal{N}_i| = 5$, and the third uses ngR-DSSR with $|\mathcal{N}_i| = 8$. Our experiments have shown that $|\mathcal{N}_i| > 8$ increases the computational time and therefore reduces the number of instances solved to optimality within the time limit.

As previously noted, in variant 1 the producers are preassigned to the depots; we consider both variant 1 and variant 2 in this section. We set the maximum computational time for each instance to five hours. There are three possibilities:

(a) The optimal solution is attained.

(b) The optimal solution is not attained, but one or more integer solutions are found during the branching process.

(c) No integer solutions are found.

Tables 3.11–3.14 present the results for variant 1, and Tables 3.15–3.18 present the results for variant 2. The branching strategies BPA and BTW are compared using the following metrics:

1. Computational time (CPU): This is reported for the problems that achieved optimality (case (a)) and represents the time to obtain the optimal solution. The CPU is set to 18000 (s) when the optimal solution is not found within five hours (cases (b) and (c)).

2. Root gap: This is calculated via *(optimal solution - root solution)/root solution*. For cases (b) and (c), the root gap is set to ∞.

3. Optimality gap: This is obtained via *(best upper bound - best lower bound)/best lower bound*. For case (a), the gap is zero and for case (c) it is infinity and therefore not reported.

4. Lower bound (LB) improvement: This is obtained via *(best lower bound - root solution)/root solution*, and it is presented for case (c). In a best-first branching strategy, this value represents the improvement in the lower bound; it allows us to compare the performance of different branching strategies for problems with no integer solution. Recall that in a best-first branching strategy the node with the best LB is treated first.

T1 is the mean time to solve the five instances in a class, and T2 is the mean time for the instances that achieved optimality. Moreover, # Opt. Sol. and # Int. Sol. are the number of instances corresponding to case (a) and case (b), respectively. The root gap, Opt. gap, and LB Imp. columns give the mean percentages for the root gaps, optimality gaps, and LB improvements when relevant. Note that, because of the significant ratio of fixed costs to variable costs, the gaps are generally small.

Variant 1: Preassignment

Generally the performance of the algorithm decreases as the number of producers increases, for both branching strategies. However, given a fixed number of producers, increasing the number of depots generally reduces the difficulty. This is because of a decrease in the producer-depot ratio when the producers are preassigned.

We now group the instances from classes pr01–pr04 according to the number of producers; this gives three groups of 30, 40, and 50 producers with 80 instances in each group. Table 3.6 presents the percentage of instances solved to optimality by each branching strategy; the percentage of instances in which at least one integer solution was found; T1; and T2.

It can be seen that BTW is more successful for larger instances. Table 3.7 shows that both branching strategies weaken in terms of the number of problems solved to optimality and the mean CPU time when the time windows are wider, i.e., pr02 and pr04. BTW weakens more significantly because although we split the time windows during the branching, the new windows are still wide enough that numerous routes with similar costs may be feasible.

Based on Table 3.6, using DSSR with BTW gives the best results. It represents the highest percentage of solved instances to optimality and therefore the least T1 and T2, among the six different combination of subproblem solving strategies and branching strategies. Table 3.7 also shows that except for the class pr02, when BTW is used, DSSR almost always outperforms *ng*R-DSSR in terms of the percentage of solved problems. DSSR also has a smaller T1.

Table 3.6: Comparison of BPA and BTW for 30 to 50 producers (Variant 1)

		BPA				BTW			
	No. producers	Optimal solution	Integer solution	T1	T2	Optimal solution	Integer solution	T1	T2
DSSR	30	75%	23%	6483	3433	90 %	5 %	2918	1251
	40	45%	34%	11329	4882	75%	11%	5587	1790
	50	30 %	38%	13288	3597	64%	9%	8288	3757
ng5	30	65%	33%	6848	1132	88 %	5%	2790	692
	40	43%	28%	11946	5016	74 %	11%	5453	1220
	50	34%	18%	13393	8708	61%	14%	8011	3310
ng8	30	78%	20%	6090	3444	88%	8%	2677	544
	40	48 %	33%	11112	4890	74%	13%	5429	1498
	50	30 %	36%	13027	3036	64%	10%	7932	3940

Table 3.7: Comparison of BPA and BTW (Variant 1)

		BPA				BTW			
	Problem class	Optimal solution	Integer solution	T1	T2	Optimal solution	Integer solution	T1	T2
DSSR	pr01	57%	35%	8942	1462	95%	2%	1595	791
	pr02	40%	28%	12698	7203	45%	18%	10905	3125
	pr03	60%	30%	8446	3325	97%	3%	1612	1014
	pr04	43%	32%	11381	3893	68%	10%	8279	4133
ng5	pr01	48 %	42%	10001	1176	95 %	3%	1275	459
	pr02	28%	22%	13780	8705	43%	18%	10961	2761
	pr03	68%	23%	6981	1637	95%	5%	1294	396
	pr04	43%	17%	12154	8291	63%	13%	8143	3347
ng8	pr01	60 %	33%	8843	2031	95%	3%	1287	474
	pr02	42%	27%	12254	6818	43%	20%	10953	3137
	pr03	60%	30%	8102	2588	95%	5%	1293	395
	pr04	45%	28%	11106	3724	67%	12%	7851	3969

Variant 2: No preassignment

We solved the same instances using the algorithm adapted for variant 2. We used the same two branching strategies. Tables 3.15–3.18 report the results for pr01–pr04 with 30, 40, and 50 producers.

We again group the instances according to the number of producers. Tables 3.8 and 3.9 compare the performance of BPA and BTW in terms of the instances solved to optimality and those with at least one integer solution. BTW generally outperforms BPA in terms of the number of instances solved. However, BPA is competitive with BTW when the time windows are wider (pr02 and pr04).

Table 3.8 shows that the combination of BTW and $ng5$ definitely outperforms the other combinations. A comparison of pr01 and pr03 with pr02 and pr04 shows the higher difficulty of the instances with wider time windows. The results indicate that the plant location (inside or outside) has no significant impact on the difficulty of the problem.

3.6.4 Value of preassignment

To evaluate the impact of preassignment in terms of solution quality and computational time, we selected a subset of instances with 40 producers (instances with identifier "2" among the 5 instances with the same size combination). The results for this subset of

Table 3.8: Comparison of BPA and BTW for 30 to 50 producers (Variant 2)

		BPA				BTW			
	No. producers	Optimal solution	Integer solution	T1	T2	Optimal solution	Integer solution	T1	T2
DSSR	30	65%	35%	6827	1035	83%	9%	4944	2650
	40	43 %	33%	12012	5166	54 %	13%	10193	6023
	50	31%	19%	13610	8404	45%	6%	10841	7492
ng5	30	66%	34%	6863	1574	88%	5%	2804	710
	40	43 %	33%	11943	5019	74%	11%	5418	1231
	50	33 %	16%	13508	8569	61 %	14%	8149	3506
ng8	30	65%	34%	6757	850	83%	8%	4681	2309
	40	44 %	34%	11724	5062	53%	11%	10041	5380
	50	33 %	19%	13507	8563	46 %	6%	10887	7854

Table 3.9: Comparison of BPA and BTW (Variant 2)

		BPA				BTW			
	Problem class	Optimal solution	Integer solution	T1	T2	Optimal solution	Integer solution	T1	T2
DSSR	pr01	47%	45%	10003	739	83%	12%	4656	2440
	pr02	27%	23%	14039	8641	28%	5%	14237	9487
	pr03	68%	23%	6976	1627	85%	10%	3779	1758
	pr04	43%	23%	12248	8466	45%	10%	11967	7869
ng5	pr01	48 %	42%	9979	1140	95%	3%	1440	665
	pr02	28%	22%	14154	9424	43%	17%	10980	2765
	pr03	68%	23%	6872	1498	95%	5%	1299	401
	pr04	43%	23%	12080	8152	63%	15%	8109	3432
ng8	pr01	48 %	40%	9978	1139	87%	10%	4379	2712
	pr02	27%	28%	13814	7903	28%	3%	14107	9305
	pr03	70%	22%	6851	2224	85 %	10%	3732	1685
	pr04	43%	25%	12007	8036	42%	10%	11928	7022

instances, using the DSSR and BTW strategies for both variant 1 and 2 are reported in Table 3.10. In this table, column "st." represents the status of solution based on the three possible cases described in Section 3.6.3. The columns "LB", "UB" and "CPU" represents the value of the solution in the root node, best upper bound obtained in the time limit and the computational time, respectively.

The performance of our algorithm depends on the producer-depot ratio, and variant 2 is therefore more difficult to solve. In fact, a higher producer-depot ratio represents larger subproblems, and consequently, the overall computation is more time demanding. This fact is illustrated by the larger number of instances with optimal solutions in a much smaller average computational time in the case of instances in variant 1, compared to the instances in variant 2.

However, one must notice that the quality of producer-depot preassignments has a significant impact on the value of the solution. As mentioned in Section 3.6.1, in our instances, we assigned producers to the depots based on a greedy heuristics. Table 3.10 compares also the solution quality of the same instances with and without the mentioned preassignment strategy. The comparison shows that there is a trade-off between computational effort and solution quality following a simple producer-depot preassignment. In fact, as this is the case in the most of MDVRPs, the producer-depot assignment may have a significant impact on the quality of the solution (37% deterioration in average in the value of the solution in the case of the considered instances). This result provides practical managerial insights to evaluate the value of integrating assignment decisions into the model, where the decision maker has this choice.

Table 3.10: Comparison of results with and without preassignment

	Instances with preassignment				Instances without preassignment			
Instance	st.	LB	UB	CPU	st.	LB	UB	CPU
pr01-40-2D2P-2	a	18164.3	19365.2	1	a	14159.1	14159.6	2
pr01-40-2D3P-2	a	20028	20074.3	3	b	17856.3	19743.2	18000
pr01-40-3D2P-2	a	20724	22271.3	3	a	15341.7	16691.9	1045
pr01-40-3D3P-2	a	21915.4	21936.5	3	a	16665.6	16730.9	11723
pr02-40-2D2P-2	a	15930.7	16712.5	588	c	12586.4	∞	18000
pr02-40-2D3P-2	c	15557.2	∞	18000	c	12758.2	∞	18000
pr02-40-3D2P-2	a	17938.2	19607.4	10	a	13764.4	14150.8	849
pr02-40-3D3P-2	a	18053.9	19674.9	7	c	13253.6	∞	18000
pr03-40-2D2P-2	a	25797	27154.7	37	a	18367.6	18393.9	13
pr03-40-2D3P-2	a	23657.6	23686.5	8	c	16921.9	∞	18000
pr03-40-3D2P-2	a	27103.5	29765.1	1	a	18076	18098.9	32
pr03-40-3D3P-2	a	29404.1	32145.5	503	a	18235.2	18308.3	28
pr04-40-2D2P-2	b	16602.6	18286.6	18000	c	13775.5	∞	18000
pr04-40-2D3P-2	a	24753.2	27014.4	9337	a	18428.7	18504.8	21
pr04-40-3D2P-2	a	24157.9	26788.8	2472	c	13421.3	∞	18000
pr04-40-3D3P-2	c	19333.7	∞	18000	a	14128.8	14922.9	11282
Average	13 a, 1 b, 2c	21195.1		4186	9 a, 1 b, 6 c	15483.8		9437

3.7 Conclusions

We have considered a new variant of the vehicle routing problem with attributes such as multiple depots, heterogeneous fleets of vehicles, time windows, and deliveries to plants. Its main novelty is the need to satisfy the plant demands by delivering the supplies collected earlier. We introduced a new set covering model for this problem, and we proposed a specialized cutting-edge column generation procedure to solve its linear relaxation. We also presented a new branching strategy based on the special structure of the problem and compared its performance with the well-known BTW.

To evaluate our algorithm, we developed randomly generated test problems with and without producer-depot preassignments. We obtained promising results in terms of solution quality and computational time, especially for problems with up to 50 producers.

Future research will focus on developing more intelligent branching strategies and considering the more complex route structures presented in Section 3.2. Our long-term goal is the effective solution of the given problem in the presence of stochastic parameters, which would make the model more realistic.

Acknowledgements

Partial funding for this project was provided by the Natural Sciences and Engineering Research Council of Canada (NSERC), through its Industrial Research Chair, Collaborative Research and Development, and Discovery Grant programs. We also received support from the Fonds de recherche du Québec - Nature et technologies (FRQ-NT)

through its Team Research Project program, and from the EMME/2-STAN Royalty Research Funds (Dr. Heinz Spiess). We also gratefully acknowledge the support of Fonds de recherche du Québec through infrastructure grants and the support of Calcul Québec and Compute Canada through access to their high-performance computing infrastructure.

Table 3.11: Results for instances with inside plants and narrow time windows (pr01); variant 1

		DSSR							ng5							ng8						
Instance set	Branching	Opt.	Int.	T1	T2	root gap	opt. gap	LB Imp.	Opt.	Int.	T1	T2	root gap	opt. gap	LB Imp.	Opt.	Int.	T1	T2	root gap	opt. gap	LB Imp.
pr01-30-2D2P	BPA	5	0	5756.3	5756.3	3.6	–	–	4	1	3601.3	1.6	0.1	7.8	–	5	0	5378.3	5378.3	3.6	–	–
	BTW	5	0	4.9	4.9	3.6	–	–	5	0	8.4	8.4	3.6	–	–	5	0	8.3	8.3	3.6	–	–
pr01-30-2D3P	BPA	2	3	11430	1575	3.4	4.4	–	3	2	7205.2	8.6	0.2	6.4	–	2	3	11311.9	1279.8	3.4	3.0	–
	BTW	5	0	8.9	8.9	6.9	–	–	5	0	13.9	13.9	6.9	–	–	5	0	13.5	13.5	6.9	–	–
pr01-30-3D2P	BPA	5	0	7.5	7.5	1.9	–	–	3	2	7201.6	2.7	0.2	3.7	–	5	0	6.4	6.4	1.9	–	–
	BTW	5	0	0.1	0.1	1.9	–	–	5	0	0.1	0.1	1.9	–	–	5	0	0.1	0.1	1.9	–	–
pr01-30-3D3P	BPA	4	1	5529.3	2411.6	1.6	3.6	–	2	3	10801.5	3.6	0.3	7.8	–	4	1	5207.8	2009.8	1.6	3.4	–
	BTW	5	0	0.5	0.5	2.7	–	–	5	0	0.5	0.5	2.7	–	–	5	0	0.5	0.5	2.7	–	–
pr01-40-2D2P	BPA	3	1	8587.4	2312.3	2.2	2.0	2.8	4	1	3605	6.2	0.1	11.3	–	4	0	8242.8	5803.6	3.3	–	4.4
	BTW	5	0	24.5	24.5	4.3	–	–	5	0	10.9	10.9	4.3	–	–	5	0	11	11	4.3	–	–
pr01-40-2D3P	BPA	4	1	7202.9	4503.6	0.5	4.9	–	1	2	14418.4	92	0.3	8.1	1.1	4	1	7203.4	4504.3	0.5	4.7	–
	BTW	5	0	5.2	5.2	1.5	–	–	5	0	4.1	4.1	1.5	–	–	5	0	4.1	4.1	1.5	–	–
pr01-40-3D2P	BPA	2	2	11168.3	920.6	3.8	2.2	7.0	1	4	14400.8	4	0.1	5.6	–	2	3	11132.5	831.3	3.8	4.2	–
	BTW	5	0	5.3	5.3	6.6	–	–	5	0	5.5	5.5	6.6	–	–	5	0	5.4	5.4	6.6	–	–
pr01-40-3D3P	BPA	2	3	10801	2.4	0.1	5.1	–	2	1	10829	72.4	0.3	6.3	1.9	2	3	10801	2.5	0.1	5.8	–
	BTW	5	0	14.2	14.2	4.4	–	–	5	0	16.6	16.6	4.4	–	–	5	0	16.4	16.4	4.4	–	–
pr01-50-2D2P	BPA	2	2	10804.7	11.8	0.0	2.8	1.7	2	2	10807.1	17.9	0.1	4.2	0.7	2	2	10804.5	11.3	0.0	2.5	6.7
	BTW	4	0	3603	3.8	1.8	–	7.5	4	0	3604.1	5.2	1.8	–	10.4	4	0	3604	5	1.8	–	8.4
pr01-50-2D3P	BPA	1	3	14402.4	12.1	0.1	6.3	0.9	3	2	11149.2	6582.1	0.7	6.0	–	1	3	14403	14.8	0.1	6.5	0.8
	BTW	4	1	8063.9	5579.9	6.6	3.8	–	4	1	7461.2	4826.5	6.6	1.6	–	4	1	7629.7	5037.1	6.6	3.4	–
pr01-50-3D2P	BPA	1	3	14400.1	0.5	4.2	4.6	8.4	1	4	14401.9	9.3	0.1	7.2	–	1	3	14400.1	0.6	4.2	4.9	3.6
	BTW	4	0	3766.6	208.2	5.6	–	15.8	4	1	3777.7	222.1	5.6	6.3	–	4	1	3774.5	218.1	5.6	6.2	–
pr01-50-3D3P	BPA	3	2	7217.5	29.2	0.1	2.6	–	3	1	11587.1	7311.9	0.9	6.5	0.5	4	1	7223.3	4529.2	0.7	3.7	–
	BTW	5	0	3638.3	3638.3	1.4	–	–	5	0	396.4	396.4	1.4	–	–	5	0	371.2	371.2	1.4	–	–

Table 3.12: Results for instances with inside plants and wide time windows (pr02); variant 1

		DSSR							ng5							ng8						
Instance set	Branching	Opt.	Int.	T1	T2	root gap	opt. gap	LB Imp.	Opt.	Int.	T1	T2	root gap	opt. gap	LB Imp.	Opt.	Int.	T1	T2	root gap	opt. gap	LB Imp.
pr02-30-2D2P	BPA	3	2	8478.7	2131.2	5.1	6.3	–	4	1	4579.4	1224.3	4.4	21.1	–	4	1	8116.4	5645.5	9.8	7.9	–
	BTW	2	2	10801.3	3.2	0.2	4.5	19.8	2	2	10801.1	2.9	0.2	5.9	14.8	2	2	10801.2	3.1	0.2	4.0	14.6
pr02-30-2D3P	BPA	3	2	10837.1	6061.8	5.1	6.7	–	2	2	11379.4	1448.5	2.8	3.1	5.1	3	2	10853.2	6088.6	5.3	3.0	–
	BTW	4	1	7214.5	4518.2	6.0	3.4	–	4	1	7207.4	4509.3	6.3	2.8	–	4	1	7207.4	4509.2	6.1	2.8	–
pr02-30-3D2P	BPA	4	1	8516.6	6145.7	16.1	2.6	–	1	4	16012.6	8063.1	8.1	7.7	–	4	1	6717.9	3897.4	16.1	1.4	–
	BTW	3	1	7822.9	1038.1	10.6	8.2	14.5	3	0	7906.1	1176.9	10.6	–	14.4	3	1	7619.3	698.8	10.6	9.7	16.3
pr02-30-3D3P	BPA	4	1	7842.9	5303.6	9.9	6.8	–	2	2	10925.7	314.2	1.1	5.9	5.1	4	1	6192	3240	9.9	6.6	–
	BTW	4	0	3896.3	370.4	6.3	–	22.9	4	0	3893.7	367.1	6.4	–	5.9	4	0	3812.6	265.8	6.3	–	15.2
pr02-40-2D2P	BPA	1	1	18000	18000	4.9	3.5	3.5	0	3	18000	18000	–	12.1	2.3	1	1	18000	18000	4.9	4.7	5.0
	BTW	1	1	14520	600	4.9	2.4	8.8	1	1	14599	995	5.1	2.2	7.5	1	1	16126.9	8634.5	4.9	3.1	6.2
pr02-40-2D3P	BPA	2	2	10802.1	5.3	0.1	5.5	5.9	1	0	16718.3	11591.6	2.8	–	3.0	2	2	10802.2	5.5	0.1	5.3	6.5
	BTW	3	1	10077.9	4796.4	2.1	3.5	9.0	3	1	10804.2	6007	2.1	3.4	7.4	3	1	9755.6	4259.3	2.1	3.8	15.3
pr02-40-3D2P	BPA	2	3	13964.9	7912.3	4.7	4.1	–	3	1	10259	5098.3	2.8	14.6	3.6	2	3	13079.4	5698.6	4.7	4.7	–
	BTW	3	2	10802	6003.4	7.6	4.3	–	2	2	10803.9	9.8	5.0	3.3	11.6	2	3	10805.1	12.7	4.7	4.6	–
pr02-40-3D3P	BPA	2	1	12657.1	4642.7	8.0	5.7	3.5	2	0	12480	4200.1	2.2	–	3.3	2	1	12023.9	3059.7	8.0	5.9	5.3
	BTW	3	0	7521.1	535.1	7.6	–	14.6	3	0	7286.9	144.8	7.6	–	12.4	3	0	7284.9	141.5	7.6	–	14.9
pr02-50-2D2P	BPA	0	1	18000	18000	–	1.6	5.1	0	0	18000	18000	–	–	1.7	0	1	18000	18000	–	1.4	2.9
	BTW	1	0	14404.5	22.6	3.1	–	4.6	1	0	14405.6	27.9	3.1	–	4.2	1	0	14403.1	15.6	3.1	–	4.2
pr02-50-2D3P	BPA	0	1	18000	18000	–	7.1	2.1	0	0	18000	18000	–	–	1.7	0	1	18000	18000	–	7.4	2.7
	BTW	0	2	18000	18000	–	8.3	6.0	0	1	18000	18000	–	7.4	5.5	0	2	18000	18000	–	8.4	6.2
pr02-50-3D2P	BPA	2	1	10857.6	144	0.4	7.2	2.8	2	0	11007	517.5	0.9	–	2.3	2	1	10842	104.9	0.4	7.0	3.0
	BTW	2	1	11342.8	1357	0.4	1.9	7.9	2	1	11287.9	1219.8	0.4	1.7	7.2	2	0	11209.6	1024	0.4	–	7.4
pr02-50-3D3P	BPA	1	1	14418.5	92.5	0.1	8.5	4.1	0	0	18000	18000	–	–	1.2	1	1	14415.6	78	0.1	8.1	2.1
	BTW	1	0	14451.9	259.3	0.1	–	8.7	1	2	14533.4	667.1	0.1	6.4	6.6	1	1	14414.8	74.2	0.1	5.0	10.1

Table 3.13: Results for instances with outside plants and narrow time windows (pr03); variant 1

		DSSR							ng5							ng8						
Instance set	Branching	Opt.	Int.	T1	T2	root gap	opt. gap	LB Imp.	Opt.	Int.	T1	T2	root gap	opt. gap	LB Imp.	Opt.	Int.	T1	T2	root gap	opt. gap	LB Imp.
pr03-30-2D2P	BPA	5	0	19.2	19.2	2.5	–	–	5	0	0.7	0.7	0.1	–	–	5	0	21.7	21.7	2.5	–	–
	BTW	5	0	0.5	0.5	2.5	–	–	5	0	0.3	0.3	2.5	–	–	5	0	0.3	0.3	2.5	–	–
pr03-30-2D3P	BPA	1	4	18000	18000	11.9	4.0	–	3	2	7211.5	19.2	0.5	4.5	–	1	4	18000	18000	5.8	3.4	–
	BTW	5	0	3183.4	3183.4	10.0	–	–	4	1	3681.6	102	9.7	1.7	–	4	1	3680.4	100.5	9.7	1.2	–
pr03-30-3D2P	BPA	5	0	11.9	11.9	2.9	–	–	4	1	4391.3	989.2	2.3	8.5	–	5	0	12.2	12.2	2.9	–	–
	BTW	5	0	0.6	0.6	2.9	–	–	5	0	0.5	0.5	2.9	–	–	5	0	0.5	0.5	2.9	–	–
pr03-30-3D3P	BPA	4	1	3600.7	0.9	0.2	6.3	–	5	0	7.7	7.7	0.4	–	–	4	1	3600.8	1	0.2	6.1	–
	BTW	5	0	6	6	3.2	–	–	5	0	6.7	6.7	3.2	–	–	5	0	6.6	6.6	3.2	–	–
pr03-40-2D2P	BPA	3	2	7438	396.6	2.0	3.2	–	4	1	7245.4	4556.8	2.0	4.6	–	3	2	7437.3	395.5	2.0	3.3	–
	BTW	4	1	3610.5	13.1	2.8	5.3	–	4	1	3608	9.9	2.8	3.4	–	4	1	3607.8	9.8	2.8	5.5	–
pr03-40-2D3P	BPA	3	2	7448.1	413.6	1.8	6.3	–	3	1	7416.7	361.1	0.5	4.6	5.2	3	2	7458.5	430.9	1.8	6.4	–
	BTW	5	0	442.7	442.7	4.0	–	–	5	0	349.8	349.8	4.1	–	–	5	0	355.6	355.6	4.0	–	–
pr03-40-3D2P	BPA	3	1	7200.1	0.2	0.0	3.2	10.2	1	3	14401.6	8.1	0.1	2.3	3.3	3	1	7200.1	0.2	0.0	3.9	8.5
	BTW	5	0	0.8	0.8	4.6	–	–	5	0	0.9	0.9	4.6	–	–	5	0	0.9	0.9	4.6	–	–
pr03-40-3D3P	BPA	3	1	7202.3	3.8	0.3	1.0	5.7	2	2	10810.3	25.8	0.3	3.6	2.6	4	0	7202.4	4503	1.2	–	10.4
	BTW	5	0	257.6	257.6	2.7	–	–	5	0	137.6	137.6	2.8	–	–	5	0	135.3	135.3	2.8	–	–
pr03-50-2D2P	BPA	3	1	10800.6	6000.9	2.1	4.7	2.8	4	1	10268.9	8336.1	2.0	2.4	–	2	2	10800.6	1.6	0.1	2.9	2.6
	BTW	4	1	3667.1	83.8	3.4	5.5	–	4	1	3623.5	29.4	3.4	1.8	–	4	1	3623.8	29.8	3.4	1.8	–
pr03-50-2D3P	BPA	3	1	10827.2	6045.4	1.2	1.6	0.9	2	2	11048.6	621.5	0.9	3.9	1.6	3	1	7232.7	54.5	1.2	1.7	0.9
	BTW	5	0	1151.8	1151.8	2.5	–	–	5	0	457.2	457.2	2.4	–	–	5	0	448.1	448.1	2.4	–	–
pr03-50-3D2P	BPA	2	2	14400.1	9000.1	1.4	1.3	0.3	4	0	7209.7	4512.2	0.6	–	1.1	2	2	13851.7	7629.4	1.3	1.3	1.7
	BTW	5	0	5.5	5.5	3.8	–	–	5	0	2.5	2.5	3.8	–	–	5	0	2.4	2.4	3.8	–	–
pr03-50-3D3P	BPA	1	3	14401.2	6.2	0.1	3.3	7.2	4	1	3765.4	206.8	0.3	9.4	–	1	3	14401.4	6.8	0.1	3.3	3.2
	BTW	5	0	7020.7	7020.7	4.2	–	–	5	0	3656	3656	4.2	–	–	5	0	3654.7	3654.7	4.2	–	–

Table 3.14: Results for instances with outside plants and wide time windows (pr04); variant 1

		DSSR							ng5							ng8						
Instance set	Branching	Opt.	Int.	T1	T2	root gap	opt. gap	LB Imp.	Opt.	Int.	T1	T2	root gap	opt. gap	LB Imp.	Opt.	Int.	T1	T2	root gap	opt. gap	LB Imp.
pr04-30-2D2P	BPA	4	1	8307.5	5884.4	6.9	3.8	–	3	2	7403.7	339.5	3.9	6.2	–	4	1	7104.2	4380.3	6.9	3.6	–
	BTW	5	0	2867.7	2867.7	8.4	–	–	5	0	58.9	58.9	8.4	–	–	5	0	29.5	29.5	8.4	–	–
pr04-30-2D3P	BPA	3	0	7212.3	20.4	0.9	–	9.3	4	1	3883.2	354.1	3.2	3.7	–	3	0	7210.8	17.9	0.9	–	7.2
	BTW	4	0	6484.4	3605.5	3.5	–	19.6	4	0	7202.8	4503.6	3.6	–	13.6	4	0	5793.8	2742.2	3.5	–	11.9
pr04-30-3D2P	BPA	5	0	38.6	38.6	1.3	–	–	4	1	7465.6	4831.9	4.2	6.9	–	5	0	28.2	28.2	1.3	–	–
	BTW	5	0	2	2	1.3	–	–	5	0	0.7	0.7	1.3	–	–	5	0	0.7	0.7	1.3	–	–
pr04-30-3D3P	BPA	3	2	8136.9	1561.5	2.8	3.0	–	3	2	7503.3	505.6	6.0	5.0	–	4	1	7680.6	5100.8	3.7	3.9	–
	BTW	5	0	4398.6	4398.6	5.5	–	–	4	0	3861.1	326.4	5.5	–	4.2	4	1	3856.8	321	5.5	1.4	–
pr04-40-2D2P	BPA	2	2	10964.5	411.1	1.4	2.8	4.8	3	1	8365.3	1942.2	2.6	8.1	4.3	2	2	10925.8	314.5	1.4	3.3	4.8
	BTW	2	2	10802.1	5.3	1.4	3.1	10.7	2	1	10802.1	5.3	1.4	1.9	8.8	2	1	10802	5	1.4	1.2	10.5
pr04-40-2D3P	BPA	0	2	18000	18000	–	6.8	4.7	3	0	9558.3	3930.6	2.9	–	4.4	0	2	18000	18000	–	4.5	3.3
	BTW	2	0	12578.2	4445.4	8.9	–	6.1	2	1	12388.9	3972.3	8.9	4.9	10.7	2	1	12367.5	3918.8	8.9	4.9	8.9
pr04-40-3D2P	BPA	2	3	17181.8	15954.6	4.1	5.4	–	1	2	18000	18000	2.2	7.2	3.8	2	3	16076.2	13190.6	7.8	3.9	–
	BTW	4	1	7736.6	5170.8	8.3	4.0	–	4	1	7288.6	4610.8	8.3	4.1	–	4	1	7234	4542.5	8.3	3.7	–
pr04-40-3D3P	BPA	2	0	12650.5	4626.3	2.4	–	9.5	3	0	14621.8	12369.7	4.0	–	8.8	2	0	12200.6	3501.5	2.4	–	9.3
	BTW	3	1	10990.1	6316.8	5.7	1.1	17.4	3	1	9139.9	3233.1	5.7	1.9	18.5	3	1	8345.1	1908.4	5.7	1.9	16.9
pr04-50-2D2P	BPA	1	4	14402.4	12	0.1	3.9	–	0	1	18000	18000	–	4.2	3.1	1	4	14402.9	14.6	0.1	3.8	–
	BTW	4	0	7481.1	4851.4	4.7	–	5.7	3	2	11191.5	6652.5	4.5	4.0	–	4	0	9887	7858.7	4.7	–	5.8
pr04-50-2D3P	BPA	1	2	14403.5	17.6	0.2	6.8	2.1	1	0	18000	18000	2.4	–	2.4	1	1	14403.9	19.6	0.2	7.4	2.1
	BTW	1	2	14407.2	36	0.2	5.0	5.1	1	1	14409.8	48.9	0.2	2.9	3.1	2	0	14409.5	9023.7	3.7	–	4.2
pr04-50-3D2P	BPA	2	2	10866.6	166.4	1.4	2.9	8.4	1	0	15043.7	3218.7	1.8	–	2.9	2	2	10838.2	95.6	1.4	2.5	6.4
	BTW	4	0	3678.2	97.8	3.0	–	9.4	4	0	3633.7	42.1	3.0	–	9.5	4	0	3633.2	41.5	3.0	–	8.7
pr04-50-3D3P	BPA	1	1	14403.7	18.6	0.3	4.2	3.9	0	0	18000	18000	–	–	2.6	1	1	14404.1	20.4	0.3	4.6	4.3
	BTW	2	0	17921	17802.4	4.9	–	9.8	1	1	17742.5	16712.6	0.3	2.8	10.8	1	2	17846.9	17234.6	0.3	1.9	13.3

Table 3.15: Results for instances with inside plants and narrow time windows (pr01); variant 2

		DSSR							ng5							ng5						
Instance set	Branching	Opt.	Int.	T1	T2	root gap	opt. gap	LB Imp.	Opt.	Int.	T1	T2	root gap	opt. gap	LB Imp.	Opt.	Int.	T1	T2	root gap	opt. gap	LB Imp.
pr01-30-2D2P	BPA	4	1	3601.1	1.3	0.1	9.2	–	4	1	3601.3	1.6	0.1	9.0	–	4	1	3601.3	1.6	0.1	9.2	–
	BTW	4	1	3601.4	1.7	0.1	3.5	–	5	0	6.9	6.9	3.6	–	–	4	1	3601.5	1.9	0.1	4.3	–
pr01-30-2D3P	BPA	3	2	7203.9	6.5	0.2	6.7	–	3	2	7204.1	6.9	0.2	6.6	–	3	2	7204.2	7	0.2	6.5	–
	BTW	5	0	1238.6	1238.6	4.8	–	–	5	0	10.6	10.6	6.9	–	–	5	0	515.3	515.3	4.8	–	–
pr01-30-3D2P	BPA	3	2	7201.6	2.6	0.2	5.0	–	3	2	7201.7	2.8	0.2	4.8	–	3	2	7201.6	2.7	0.2	5.0	–
	BTW	5	0	12.7	12.7	3.3	–	–	5	0	0.1	0.1	1.9	–	–	5	0	14.4	14.4	3.3	–	–
pr01-30-3D3P	BPA	2	3	10801.1	2.8	0.3	7.6	–	2	3	10801.4	3.5	0.3	7.7	–	2	3	10801.4	3.5	0.3	7.5	–
	BTW	5	0	152.9	152.9	6.0	–	–	5	0	0.5	0.5	2.7	–	–	5	0	119	119	6.0	–	–
pr01-40-2D2P	BPA	4	1	3604.2	5.2	0.1	11.5	–	4	1	3605	6.3	0.1	11.3	–	4	1	3604.8	6	0.1	11.3	–
	BTW	4	1	3602.2	2.7	0.1	9.0	–	5	0	7.3	7.3	4.3	–	–	5	0	3602.3	3602.3	2.7	–	–
pr01-40-2D3P	BPA	1	3	14417.1	85.4	0.3	7.9	2.0	1	3	14419.9	99.7	0.3	8.3	0.4	1	2	14418.3	91.7	0.3	8.4	1.2
	BTW	3	1	7679.8	799.7	5.1	2.8	10.1	5	0	4.4	4.4	1.5	–	–	3	1	8013.1	1355.2	5.1	3.9	9.5
pr01-40-3D2P	BPA	1	4	14400.8	3.8	0.1	5.3	–	1	4	14400.7	3.6	0.1	5.7	–	1	4	14400.7	3.7	0.1	5.7	–
	BTW	4	1	8644.7	6305.9	7.7	5.4	–	5	0	5.8	5.8	6.6	–	–	4	1	7742.3	5177.9	7.7	3.0	–
pr01-40-3D3P	BPA	2	1	10830.2	75.5	0.3	6.3	1.9	2	1	10829.2	73	0.3	6.3	2.2	2	1	10828.7	71.8	0.3	6.1	1.8
	BTW	5	0	3893.2	3893.2	4.9	–	–	5	0	15.9	15.9	4.4	–	–	5	0	1765.9	1765.9	4.9	–	–
pr01-50-2D2P	BPA	2	2	10806.7	16.8	0.1	4.2	0.9	2	2	10808.2	20.5	0.1	4.3	0.7	2	2	10807.8	19.5	0.1	4.3	0.7
	BTW	5	0	746.3	746.3	4.2	–	–	4	0	3604	5	1.8	–	10.0	5	0	967	967	4.2	–	–
pr01-50-2D3P	BPA	3	2	11100.7	6501.1	0.7	5.8	–	3	2	11121.6	6536.1	0.7	5.9	–	3	2	11114.9	6524.8	0.7	6.0	–
	BTW	5	0	1065.3	1065.3	2.9	–	–	4	1	9266.3	7082.9	6.6	3.2	–	5	0	971.7	971.7	2.9	–	–
pr01-50-3D2P	BPA	1	4	14402.1	10.6	0.1	7.1	–	1	3	14402.4	12.1	0.1	6.5	2.0	1	3	14402.4	11.9	0.1	6.5	2.1
	BTW	3	2	10821.6	6035.9	3.5	6.7	–	4	1	3962.7	453.4	5.6	5.4	–	3	2	10824.1	6040.2	3.5	5.0	–
pr01-50-3D3P	BPA	2	2	11661.3	2153.3	0.4	3.9	0.5	3	1	11350.7	6917.9	0.8	6.6	0.4	3	1	11354.1	6923.5	0.9	6.6	0.5
	BTW	2	1	14407.8	9019.5	0.9	6.3	1.6	5	0	391.8	391.8	1.4	–	–	3	1	14406.9	12011.6	3.6	6.9	0.3

Table 3.16: Results for instances with inside plants and wide time windows (pr02); variant 2

		DSSR							ng5							ng5						
Instance set	Branching	Opt.	Int.	T1	T2	root gap	opt. gap	LB Imp.	Opt.	Int.	T1	T2	root gap	opt. gap	LB Imp.	Opt.	Int.	T1	T2	root gap	opt. gap	LB Imp.
pr02-30-2D2P	BPA	4	1	4261.7	827.2	4.3	1.7	–	4	1	4544.8	1181	4.4	20.6	–	4	1	4267.4	834.3	4.3	21.5	–
	BTW	4	1	4048.3	560.4	4.3	25.1	–	2	2	10801.2	2.9	0.2	5.6	11.4	4	0	4013.6	517	4.3	–	3.7
pr02-30-2D3P	BPA	2	3	11504.5	1761.3	2.8	5.8	–	2	3	11358.1	1395.2	2.8	6.5	–	2	2	11357.3	1393.2	2.8	2.9	5.5
	BTW	2	0	13355.3	6388.3	4.8	–	8.4	4	1	7207.1	4508.9	6.3	1.3	–	2	0	12653.1	4632.6	4.8	–	7.9
pr02-30-3D2P	BPA	1	4	15685	6424.8	8.1	7.0	–	1	4	16253.8	9268.8	8.1	7.5	–	1	4	15331.3	4656.7	8.1	6.9	–
	BTW	2	2	11819.9	2549.8	8.6	5.1	7.0	3	0	7916.4	1194	10.6	–	19.1	2	2	12668.5	4671.3	8.6	4.6	7.9
pr02-30-3D3P	BPA	2	3	10858.8	146.9	1.0	8.2	–	3	2	10956.6	6260.9	3.6	13.4	–	2	3	10871.4	178.4	1.1	8.9	–
	BTW	3	0	14831.1	12718.5	3.6	–	8.3	4	0	4129.5	661.8	6.4	–	23.6	3	0	13044.1	9740.2	3.5	–	8.9
pr02-40-2D2P	BPA	0	2	18000	18000	–	11.9	3.8	0	2	18000	18000	–	12.1	3.4	0	4	18000	18000	–	11.7	2.5
	BTW	0	0	18000	18000	–	–	5.9	1	1	14538.9	694.3	5.1	3.4	6.0	0	0	18000	18000	–	–	6.5
pr02-40-2D3P	BPA	1	0	16706.4	11532.1	2.7	–	2.7	1	0	16790.8	11953.8	2.8	–	3.0	1	0	16062.5	8312.4	2.7	–	3.0
	BTW	1	0	18000	18000	2.7	–	5.0	3	1	10804.4	6007.4	2.1	4.2	10.8	1	0	18000	18000	2.7	–	5.7
pr02-40-3D2P	BPA	3	1	9598.8	3998	2.5	11.9	2.3	3	1	11061.1	6435.1	2.8	14.4	3.6	3	2	9059.4	3099	2.5	13.0	–
	BTW	3	0	7896.7	1161.1	2.5	–	4.2	2	1	10804	10.1	5.0	3.6	7.8	3	0	7919.7	1199.5	2.5	–	4.1
pr02-40-3D3P	BPA	2	0	13297	6242.4	2.2	–	3.1	2	0	12324.8	3812.1	2.2	–	3.2	2	0	12289.3	3723.3	2.2	–	3.5
	BTW	1	0	14430.7	153.7	3.5	–	4.4	3	0	7312.1	186.8	7.6	–	14.4	1	0	14458	289.8	3.5	–	4.7
pr02-50-2D2P	BPA	0	0	18000	18000	–	–	1.8	0	0	18000	18000	–	–	1.6	0	1	18000	18000	–	15.3	2.1
	BTW	0	0	18000	18000	–	–	2.1	1	0	14403.8	19.2	3.1	–	4.0	0	0	18000	18000	–	–	2.6
pr02-50-2D3P	BPA	0	0	18000	18000	–	–	1.6	0	0	18000	18000	–	–	1.9	0	0	18000	18000	–	–	1.7
	BTW	0	0	18000	18000	–	–	2.8	0	1	18000	18000	–	5.4	5.9	0	0	18000	18000	–	–	2.7
pr02-50-3D2P	BPA	1	0	14552.4	762.2	1.0	–	1.9	1	0	14556.5	782.4	1.1	–	2.3	1	0	14527.3	636.7	1.0	–	2.4
	BTW	1	0	14463.1	315.4	1.0	–	3.0	2	1	11337.4	1343.4	0.4	1.7	7.2	1	0	14522.7	613.5	1.0	–	3.2
pr02-50-3D3P	BPA	0	0	18000	18000	–	–	1.2	0	0	18000	18000	–	–	1.3	0	0	18000	18000	–	–	1.3
	BTW	0	0	18000	18000	–	–	3.0	1	2	14510.5	552.5	0.1	6.7	4.3	0	0	18000	18000	–	–	3.0

Table 3.17: Results for instances with outside plants and narrow time windows (pr03); variant 2

		DSSR							ng5							ng5						
Instance set	Branching	Opt.	Int.	T1	T2	root gap	opt. gap	LB Imp.	Opt.	Int.	T1	T2	root gap	opt. gap	LB Imp.	Opt.	Int.	T1	T2	root gap	opt. gap	LB Imp.
pr03-30-2D2P	BPA	5	0	0.6	0.6	0.1	–	–	5	0	0.6	0.6	0.1	–	–	5	0	0.7	0.7	0.1	–	–
	BTW	5	0	0.3	0.3	0.1	–	–	5	0	0.3	0.3	2.5	–	–	5	0	0.4	0.4	0.1	–	–
pr03-30-2D3P	BPA	3	2	7209.5	15.8	0.5	4.3	–	3	2	7209.3	15.6	0.5	4.2	–	3	2	7209.3	15.5	0.5	4.0	–
	BTW	5	0	979.3	979.3	4.1	–	–	4	1	3683.4	104.3	9.7	2.3	–	5	0	779.9	779.9	4.1	–	–
pr03-30-3D2P	BPA	4	1	4530	1162.5	2.3	8.9	–	4	1	4452.3	1065.4	2.3	8.7	–	4	1	4194.8	743.6	2.3	8.7	–
	BTW	5	0	226	226	4.5	–	–	5	0	0.4	0.4	2.9	–	–	5	0	228.3	228.3	4.5	–	–
pr03-30-3D3P	BPA	5	0	7.1	7.1	0.4	–	–	5	0	7.2	7.2	0.4	–	–	5	0	7.2	7.2	0.4	–	–
	BTW	5	0	34.7	34.7	0.4	–	–	5	0	6.5	6.5	3.2	–	–	5	0	35.3	35.3	0.4	–	–
pr03-40-2D2P	BPA	4	1	7270.9	4588.6	2.0	4.1	–	4	1	7251.2	4564	2.0	4.5	–	4	1	7251	4563.8	2.0	4.6	–
	BTW	4	1	3605.5	6.9	2.0	1.8	–	4	1	3607.5	9.4	2.8	5.6	–	4	1	3606.3	7.9	2.0	1.4	–
pr03-40-2D3P	BPA	3	1	7364.2	273.6	0.5	3.5	4.9	3	1	7425.6	375.9	0.5	3.3	5.2	3	1	7427.7	379.6	0.5	2.7	5.0
	BTW	2	1	14580.7	9451.6	0.5	2.3	3.3	5	0	345.2	345.2	4.1	–	–	2	1	14581.3	9453.3	0.5	1.8	3.3
pr03-40-3D2P	BPA	1	3	14401.9	9.5	0.1	2.4	3.3	1	3	14401.8	9	0.1	2.3	3.4	2	2	14401.8	9004.5	4.0	2.7	3.3
	BTW	3	2	10809.7	6016.2	4.9	1.6	–	5	0	0.9	0.9	4.6	–	–	3	2	10334.7	5224.5	4.9	1.9	–
pr03-40-3D3P	BPA	2	2	10810.3	25.9	0.3	3.3	2.4	2	2	10810.3	25.9	0.3	3.1	2.6	2	2	10810.5	26.2	0.3	3.6	3.2
	BTW	4	1	3625.2	31.4	3.4	1.8	–	5	0	128.5	128.5	2.8	–	–	4	1	3629.4	36.8	3.4	1.5	–
pr03-50-2D2P	BPA	4	1	10134.1	8167.6	2.0	2.4	–	4	1	8896.8	6621	2.0	2.3	–	4	1	8902.6	6628.3	2.0	2.3	–
	BTW	5	0	37.5	37.5	2.4	–	–	4	1	3627.6	34.5	3.4	4.7	–	5	0	40.4	40.4	2.4	–	–
pr03-50-2D3P	BPA	2	2	11035.6	588.9	0.9	3.8	1.5	2	2	11033.8	584.4	0.9	3.7	1.5	2	2	11039.1	597.6	0.9	3.9	1.6
	BTW	5	0	3993.8	3993.8	3.7	–	–	5	0	505.9	505.9	2.4	–	–	5	0	4062.2	4062.2	3.7	–	–
pr03-50-3D2P	BPA	4	0	7208.4	4510.5	0.6	–	1.1	4	0	7208.4	4510.5	0.6	–	1.1	4	0	7208.5	4510.6	0.6	–	1.7
	BTW	4	0	3609.6	11.9	0.6	–	5.1	5	0	2.4	2.4	3.8	–	–	4	0	3610.1	12.7	0.6	–	5.3
pr03-50-3D3P	BPA	4	1	3740.5	175.7	0.3	9.9	–	4	1	3761.4	201.7	0.3	10.5	–	4	1	3763.9	204.8	0.3	10.1	–
	BTW	4	1	3844.3	305.4	0.3	3.6	–	5	0	3676.9	3676.9	4.2	–	–	4	1	3870.5	338.1	0.3	4.8	–

Table 3.18: Results for instances with outside plants and wide time windows (pr04); variant 2

		DSSR							ng5							ng5						
Instance set	Branching	Opt.	Int.	T1	T2	root gap	opt. gap	LB Imp.	Opt.	Int.	T1	T2	root gap	opt. gap	LB Imp.	Opt.	Int.	T1	T2	root gap	opt. gap	LB Imp.
pr04-30-2D2P	BPA	3	2	7442.4	404	3.9	5.7	–	3	2	7399.4	332.4	3.9	5.9	–	3	2	7355.6	259.3	3.9	6.2	–
	BTW	4	0	7469.2	4836.5	4.5	–	12.3	5	0	68.2	68.2	8.4	–	–	4	0	6329.9	3412.4	4.5	–	21.0
pr04-30-2D3P	BPA	4	1	3898.7	373.4	3.2	5.0	–	4	1	3880.9	351.1	3.2	3.8	–	4	1	3814.1	267.6	3.2	4.1	–
	BTW	5	0	4543.8	4543.8	4.6	–	–	4	0	7195.8	4494.7	3.5	–	11.8	5	0	4017.8	4017.8	4.6	–	–
pr04-30-3D2P	BPA	4	1	7495.6	4869.5	4.2	6.6	–	4	1	7451.7	4814.6	4.2	6.8	–	4	1	7428.1	4785.1	4.2	6.8	–
	BTW	4	1	8006.4	5508	4.3	5.4	–	5	0	0.7	0.7	1.3	–	–	4	1	8049.7	5562.1	4.3	5.1	–
pr04-30-3D3P	BPA	3	2	7533.1	555.2	5.8	5.5	–	3	2	7482.2	470.3	6.0	5.2	–	3	2	7470.8	451.3	5.8	5.3	–
	BTW	3	2	8791.1	2651.9	6.8	9.9	–	4	0	3841.6	301.9	5.5	–	5.6	3	2	8822.1	2703.5	6.8	15.0	–
pr04-40-2D2P	BPA	3	2	8942.7	2904.6	2.6	10.5	–	3	2	8425.3	2042.1	2.6	10.3	–	3	2	8339.7	1899.6	2.6	10.4	–
	BTW	3	1	10967.8	6279.6	2.6	7.9	6.3	3	1	10802	6003.3	4.3	4.8	9.4	2	2	10954.9	387.4	0.8	7.4	6.9
pr04-40-2D3P	BPA	3	0	10038.4	4730.7	2.8	–	4.0	3	0	9390.9	3651.5	2.9	–	4.2	3	0	9250.7	3417.8	2.8	–	5.0
	BTW	3	0	11038.7	6397.8	2.9	–	4.5	2	1	12111.5	3278.8	8.9	6.3	7.5	2	0	11010.3	525.6	1.4	–	5.1
pr04-40-3D2P	BPA	1	4	18000	18000	2.1	8.4	–	1	4	18000	18000	2.3	8.4	–	1	4	18000	18000	2.0	8.2	–
	BTW	0	1	18000	18000	–	4.0	4.2	3	2	7364.2	273.7	6.4	2.0	–	0	0	18000	18000	–	–	4.9
pr04-40-3D3P	BPA	3	1	14508.8	12181.4	3.9	1.7	8.5	3	1	13952	11253.3	3.9	2.8	7.5	3	1	13436.7	10394.4	3.9	3.1	8.3
	BTW	3	0	8318.1	1863.5	3.9	–	6.8	3	1	8837.7	2729.5	5.7	2.1	22.4	3	0	9031.2	3052	3.9	–	7.5
pr04-50-2D2P	BPA	0	1	18000	18000	–	4.4	3.1	0	1	18000	18000	–	3.8	3.3	0	1	18000	18000	–	4.3	3.0
	BTW	0	0	18000	18000	–	–	3.0	3	2	11188	6646.7	4.5	4.3	–	0	0	18000	18000	–	–	3.2
pr04-50-2D3P	BPA	1	0	18000	18000	2.4	–	2.8	1	0	18000	18000	2.6	–	2.5	1	0	18000	18000	2.6	–	2.3
	BTW	1	0	14627.6	1138.1	2.2	–	4.0	1	0	14409.3	46.5	0.2	–	3.6	1	0	14552	760.2	2.2	–	3.8
pr04-50-3D2P	BPA	1	0	15114	3570.1	1.8	–	2.8	1	0	14982.1	2910.5	1.8	–	2.8	1	0	14991.3	2956.4	1.8	–	3.0
	BTW	1	0	15841.8	7209.1	1.8	–	3.4	4	0	3633	41.2	3.0	–	9.8	1	0	16369.5	9847.3	1.8	–	4.2
pr04-50-3D3P	BPA	0	0	18000	18000	–	–	2.4	0	0	18000	18000	–	–	2.6	0	1	18000	18000	–	4.8	2.3
	BTW	0	1	18000	18000	–	2.6	3.1	1	2	17860.1	17300.4	0.3	2.9	11.8	0	1	18000	18000	–	1.8	2.9

Chapter 4

A Branch-and-Price Approach for a Multi-Period Vehicle Routing Problem

This paper has been submitted to Journal of Computers & Operations Research.

Abstract

In this paper, we consider tactical planning for a class of the multi-period vehicle routing problem (MPVRP). This problem involves optimizing daily product collections from several production locations over a given planning horizon. In this context, a single vehicle routing plan for the whole horizon must be prepared, and the seasonal variations in the producers' supplies must be taken into account. The production variations over the horizon are approximated using a sequence of periods, each corresponding to a production season, while the intra-period variations are neglected. We propose a mathematical model that is based on the two-stage *a priori* optimization paradigm. The first stage corresponds to the design of a plan which, in the second stage, takes the different periods into account. The proposed set-partitioning-based formulation is solved using a branch-and-price approach. The subproblem is a multi-period elementary shortest path problem with resource constraints (MPESPPRC), for which we propose an adaptation of the dynamic-programming-based label-correcting algorithm. Computational results show that this approach is able to solve instances with up to twenty producers and five periods.

Keywords: Multi-period vehicle routing problem, Branch-and-price, Seasonality. Tactical planning.

4.1 Introduction

The vehicle routing problem (VRP) is a well-studied topic in operations research; it has been investigated by many researchers since it was introduced by [42]. Given an unlimited fleet of vehicles based in a single depot, the VRP, in its general form, involves finding a set of least-cost feasible routes to deliver goods to (or collect goods from) a set of customers. A feasible solution consists of a set of routes in which each customer is visited once by exactly one vehicle, and the quantity of delivered (collected) goods on each route does not exceed the vehicle capacity [151].

Several variants and extensions of the classical VRP have been introduced, and they are supported by a well-developed literature [37]. The problem setting that we consider is the design of tactical plans for a large class of real-life routing problems; it incorporates several new attributes and characteristics. This problem setting is inspired by a real-life application of the VRP in the dairy industry in Quebec. Milk is collected from the producers' farms and then delivered to a set of processing plants. At a given time, denoted t, a tactical plan is prepared for a horizon T, consisting of several collection days, starting at $t + \delta$. This plan is executed on a daily basis during the horizon T. We must design routes for a set of vehicles departing from different depots in a geographical region, each visiting a subset of producers and collecting a single product type, which is then delivered to the processing plants. The objective is to minimize the transportation cost while meeting the plants' demands. Every producer is visited by exactly one vehicle, and each vehicle visits only one plant per day. We assume that the daily quantity supplied by the producers satisfies the total plant demand.

The main challenge is the design of a single plan for a horizon, when the supply is subject to seasonal variations. The plan may provide service consistency and regularity by attempting to always follow the same sequence of visits. Moreover, in the dairy industry, because of contractual arrangements, the plan is the basis of the negotiations between the stakeholders involved in each contract.

To summarize, the main goal of this paper is to address the above problem by proposing routing plans that account for the seasonal variations in the production levels. We approximate the production fluctuations over the horizon T using a sequence of periods (day clusters), with the same production level within each period, forming a multi-period vehicle routing problem (MPVRP). The formulation appears similar to the scenario-based formulation of the vehicle routing problem with stochastic demands (VRPSD). However, in our problem each production level occurs only in a specific period. The main contributions of this paper are:

- We investigate the characteristics of the problem.
- We propose a mathematical programming model for the problem and the seasonal behavior of the supply.
- We propose a state-of-the-art branch-and-price algorithm. It includes a series of bounds as well as structural modifications in the multi-period problem, allowing

us to take advantage of technical advances in single-period VRPs.

- We perform an extensive analysis using a large set of randomly generated instances, to illustrate the performance and the limits of our algorithm.

The remainder of this paper is organized as follows. In Section 4.2, we describe the problem, and Section 4.3 presents the proposed multi-stage formulation. The algorithm is presented in Section 4.4, and its components are described in Sections 4.5 to 4.7. The experimental results are reported in Section 4.8, and Section 4.9 provides concluding remarks.

4.2 Problem statement

In this section, we introduce the problem; it is inspired by a dairy problem in Quebec. It involves building an *a priori* tactical plan for a given horizon over which certain parameters may vary.

For a detailed description of the dairy transportation problem in Quebec (DTPQ), the reader is referred to Lahrichi et al. [101] and Dayarian et al. [44]. The DTPQ can be briefly described as follows: In Quebec, the Fédération des producteurs de lait du Québec (FPLQ), a coalition of milk producers, is responsible for managing the collection and transportation of milk produced in the province. This involves negotiating, on behalf of the producers, annual transportation contracts with the carriers [2]. Each contract with a carrier is based on a plan containing multiple routes. Each route specifies an origin and a destination (the vehicle's depot) and consists of collection from producers followed by delivery to a processing plant and then return to the depot. The contractual regulations require a single plan to be prepared for every six-month horizon; it is the basis of negotiations and payments. Its routes are the basis of the collection-delivery operations, executed on a daily basis over the horizon covered by the contract. The goal is to minimize the transportation costs while satisfying the plant demand and visiting all the producers.

The supply may vary daily as well as seasonally. The daily variations, caused by exceptional situations such as meteorological variations, cattle nutrition, or cattle diseases, are quite minor. The seasonal variations, caused by seasonal meteorological changes and animal birth cycles are more significant and may have a greater impact on the plan. Note that, in this context, seasons are subperiods determined based on the production level of the cattle; they can be shorter or longer than calendar seasons.

Currently, these variations are not accounted for during the planning phase, and the routes are designed based on the annual average production. In seasons with a higher supply, it may not be possible to complete the planned routes because of insufficient residual capacity in the vehicles. In this case the vehicle usually travels to a plant to unload its tank and then visits the remaining producers of the planned route.

The problem considered in this paper can be formally described as follows: We wish

to design a tactical plan for a given horizon T containing several collection days. A plan consists of a set of routes, each performed by a single vehicle on every collection day of T. An unlimited fleet of identical vehicles is assumed to be based in multiple depots. On every collection day, each vehicle departs from a depot, collects a single product type from a subset of producers, delivers the collected product to a plant, and then returns to its depot. This can be seen as an extension of the well-known multi-depot vehicle routing problem (MDVRP) with additional deliveries to multiple plants. As an extension of the VRP, this problem is NP-hard [107].

We assume that a year can be divided into several periods, each corresponding to a seasonal production level. We take inter-period production variations into account; the potential intra-period fluctuations are neglected. Intra-period fluctuations can often be handled by leaving a spare capacity of 5%–10% on each vehicle when designing the routes. Daily fluctuations may vary from one producer to another, but seasonal fluctuations are strongly linked and are here assumed to be perfectly positively correlated. This correlation arises because almost all the producers in a given geographical region are exposed to similar seasonal cycles. The plants must adjust their seasonal demands according to the supply so that the total supply always covers the total demand.

The objective is to design a collection-delivery plan for a given horizon, providing a certain level of service consistency and quality while taking into account the seasonal variation and minimizing the total routing costs.

The most consistent strategy is to design the plan based on the highest production level of the horizon. The resulting routes can be performed in any period without adjustment. The main drawback of this strategy is its cost: it may require a large number of vehicles. An alternative is allow a limited number of *failures* per route, i.e., situations where a lack of vehicle capacity prevents the completion of the route. A correction, called a *recourse*, is then necessary to adjust the route to the current situation. We define a feasible route to be a route that is executable in any period of the horizon with at most one failure. The cost of a route consists of a fixed part, representing the sum of the fixed vehicle costs and the costs of the planned route arcs, and the weighted cost of the recourse actions necessary in different periods of the horizon. The period weight is proportional to the ratio of the period length and the length of the horizon.

We control the desired service quality over a given horizon by setting a *service reliability threshold* (SRT), indicating the minimum percentage of days over the horizon T that the planned routes should be executable without encountering any failures. The SRT is a tool provided to the decision-maker to govern the robustness of the plan over different periods of the horizon.

In Section 4.3, we present our model, which takes into account both the seasonal variations and the service quality.

4.3 Model

In this section, we present a model for the problem introduced in Section 4.2. The formulation takes into account the routing characteristics and specifications and the variations in the supply over a given horizon T.

4.3.1 Multi-period scheme

A convenient way to model the seasonal fluctuations is to represent the horizon as a finite set of periods. More precisely, we aggregate several days with similar seasonal characteristics to form a period.

Our multi-period scheme concatenates several periods, each corresponding to a production season. Let $\mathcal{S}$ be the set of all periods in the horizon T; within each period $s \in \mathcal{S}$, the production levels are assumed to be fixed. Accordingly, we may associate with each period s a production coefficient, P_s, which is defined to be the ratio of the production level in period s to the average annual production level. We also associate with each period a weight W_s, representing the share of period s in horizon T. It is calculated by dividing the length of period s by the length of horizon T. In other words, W_s indicates the occurrence frequency of a given production level over the horizon T.

For a given SRT, we perform the following procedure:

step 1 Let Γ be an empty period set.

step 2 The periods $s \in \mathcal{S}$ are sorted in ascending order of production level.

step 3 Following the order from **step 2**, the periods are removed from $\mathcal{S}$ and are added to Γ until their cumulative weight covers the SRT: ($\Sigma_{s\in\Gamma} W_s \geq$ SRT).

step 4 All the periods in Γ are aggregated into a mega-period, referred to as the *reference period*. It has production coefficient Pt_{ref}, the *reference production level*, defined to be the largest P_s, where $s \in \Gamma$. The reference period is added to $\mathcal{S}$.

step 5 All the production coefficients $P_s : s \in \mathcal{S}$ are normalized by division by Pt_{ref}, so that Pt_{ref} equals 1.

The plan is designed in such a way that no failure is permitted in the reference period. This procedure also allows us to reduce the number of periods. Figure 4.1(a) shows an example of the period distribution in a given horizon, where the SRT is set to 40%. In this example, periods 1 to 3 are added to Γ. Routes that have no failures in period 3 will have no failures in periods 1 and 2. Therefore, we can merge periods 1–3 into a mega-period with the production equal to 1.1 and the weight equal to the cumulative weight of periods 1–3, i.e., 60%. Figure 4.1(b) shows the normalized distribution of the periods.

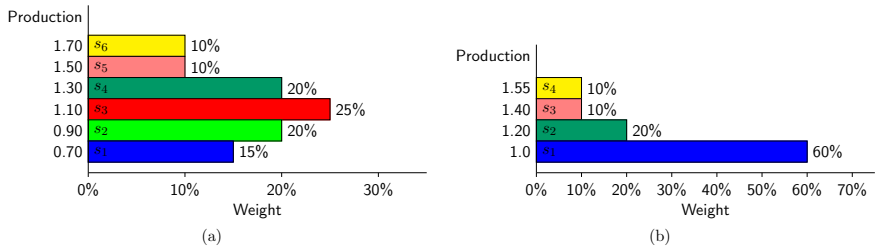

Figure 4.1: (a) Example of period distribution.
(b) Normalized distribution of periods based on reference period.

Our model is based on the *a priori* optimization framework [21, 89], originally introduced for stochastic problems. This multi-stage framework assumes that a plan is designed in the first stage. New decisions are then taken over several stages, as exact information is revealed about uncertain parameters. In a classical two-stage stochastic programming model, the first-stage decision consists of an *a priori* plan, which is executed during the second stage. During the second stage, called the execution phase, as the real values of stochastic parameters are revealed, new decisions and adjustments are made to make the plan more accurate. These second-stage adjustments (recourse) usually generate a cost or a saving that should be taken into account in the first-stage plans. The objective of a stochastic programming model is to find a first-stage plan that minimizes the expected sum of all the costs associated with the plan and the second-stage corrective actions.

Although the information for our problem is assumed to be known a priori, its multi-period nature makes it similar to a stochastic problem. We therefore develop a two-stage approach: in the first stage we design an *a priori* plan, and in the second stage we execute this plan in all of the parallel independent periods of the horizon. Our recourse policy is similar to strategy (a) in the context of the VRPSD [21]. In this strategy, the vehicle visits the producers in the same fixed order as in the *a priori* planned route. Consequently, the total traveled distance corresponds to the fixed length of the planned route plus the extra distance that must be covered when the load exceeds the vehicle capacity. We assume that if a vehicle is full after a collection, it continues to the subsequent producer, where the failure occurs. The extra distance traveled corresponds to a return trip to the plant where the vehicle empties its tank before resuming the planned route at the failure point. We selected this simple recourse because it provides high service consistency.

4.3.2 Two-stage formulation

The model is defined on a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, where $\mathcal{V}$ and $\mathcal{A}$ are the node and arc sets, respectively. The node set contains the depots, producers, and plants: $\mathcal{V} = \mathcal{D} \cup \mathcal{N} \cup \mathcal{P}$. The arc set $\mathcal{A} \subset \mathcal{V} \times \mathcal{V}$ defines feasible movements between different locations in $\mathcal{V}$. For each pair of locations $i, j \in \mathcal{V}$, $i \neq j$, there exists an arc $(i, j) \in \mathcal{A}$. Each arc $(i, j) \in \mathcal{A}$ has a nonnegative travel cost c_{ij} that is proportional to the travel

time. We assume throughout this paper that the triangle inequality holds for costs and travel times. In each period, each producer $j \in \mathcal{N}$ produces a limited quantity of product on a daily basis. The production levels in period $s \in \mathcal{S}$ are given by a vector in which the jth entry, q_j^s, is the supply from producer j. Moreover, the production level of each producer j in the reference period is given by q_j^{ref}. Therefore, based on the assumption of a perfect correlation among the production levels of different producers within each period, the supply of producer j in period s is

$$q_j^s = P_s.q_j^{ref}, \qquad j \in \mathcal{N} s \in \mathcal{S}. \tag{4.1}$$

Each plant $p \in \mathcal{P}$ receives, again on a daily basis, the collected product, and the plant demands are adjusted to the seasonal production. Each route is performed using a vehicle $k \in \mathcal{K}$, with capacity Q, where $\mathcal{K}$ represents an unlimited homogeneous fleet of vehicles. The routes are designed to have no failures in the reference periods and at most one failure in the periods with $P_s > 1$. In other words, for each route r, the following inequalities must hold:

$$\sum_{j \in r} q_j^s \leq 2Q, \qquad s \in \mathcal{S} \tag{4.2}$$

and

$$\sum_{j \in r} q_j^{ref} \leq Q. \tag{4.3}$$

Let $\mathcal{R}_{dp}$ be the set of all feasible routes from depot $d \in \mathcal{D}$ to plant $p \in \mathcal{P}$. Each feasible route corresponds to a path from a depot $d \in \mathcal{D}$ to itself and consists of collection from a subset of producers followed by delivery to a single plant. Let $\mathcal{R} = \bigcup_{d \in \mathcal{D}, p \in \mathcal{P}} \mathcal{R}_{dp}$.

Let y_r be a binary variable such that y_r is 1 if route $r \in \mathcal{R}$ is selected in the optimal solution and 0 otherwise. The collection on route r in the reference period that is delivered to plant p is denoted l_{pr}. The total delivery to each plant p must completely cover the plant's demand D_p. As mentioned before, the demand at each plant is proportional to the production level in the corresponding period. Therefore, only the demands in the reference period should be taken into account. Accordingly, let l_{pr}^{ref} and D_p^{ref} represent, respectively, the quantity collected on route r and the quantity demanded by plant p in the reference period.

Parameter a_{ir} is 1 if route r visits producer i and 0 otherwise. Associated with each vehicle $k \in \mathcal{K}$ is a fixed cost c_k that applies whenever the vehicle is employed. In general, most of the variable costs are positively correlated with the distance traveled, and thus minimizing the total distance traveled is a reasonable objective function. The deterministic cost of each route r, denoted c_r, is the sum of the costs on the arcs of the route.

Our model is then as follows:

$$\min \quad \sum_{r\in\mathcal{R}}(c_r + c_k)y_r + \mathcal{F}(x) \tag{4.4}$$

subject to

$$\sum_{r\in\mathcal{R}} a_{ir}y_r = 1 \quad (i \in \mathcal{N}); \tag{4.5}$$

$$\sum_{r\in\mathcal{R}} l_{pr}^{ref}y_r \geq D_p^{ref} \quad (p \in \mathcal{P}); \tag{4.6}$$

$$y_r \in \{1, 0\} \quad (r \in \mathcal{R}). \tag{4.7}$$

Here $\mathcal{F}(x)$, the recourse function defined below by (RF), represents the total recourse cost incurred in the different periods for a given x, and x is the set of arcs used in the construction of the routes forming the solution of problem (4.4)–(4.7). In fact, $\mathcal{F}(x)$ represents the value of the second-stage problem given a first-stage solution x. Constraint (4.5) ensures that each producer is visited exactly once by exactly one route, and constraint (4.6) guarantees that the plant demands are satisfied.

For the second-stage problem (the recourse problem), let x_{ijk} be a binary parameter obtained from a given first-stage solution; it is 1 if customer $j \in \mathcal{N}$ follows customer $i \in \mathcal{N}$ on a route performed by vehicle $k \in \mathcal{K}$. The vector q^s represents the supply in period s. Moreover, suppose that z_{ijk}^s is the flow on arc (i, j) for all $i, j \in \mathcal{V}$ traveled by vehicle k in period s. Also, let w_{ik}^s be a parameter that is 1 if a failure occurs as producer i is served by vehicle k in period s and 0 otherwise. Therefore, z^s and w^s represent the vectors z_{ijk}^s and w_{ik}^s, respectively. The recourse problem is defined as follows:

(RF)

$$\mathcal{F}(x) = \sum_{s\in\mathcal{S}} P_s F(x, q^s) \tag{4.8}$$

where

$$F(x, q^s) = \min \quad \sum_{k\in\mathcal{K}}\sum_{i\in\mathcal{N}} 2c_{ip_k}w_{ik}^s \tag{4.9}$$

subject to

$$z_{ijk}^s \leq Qx_{ijk} \qquad (i, j \in \mathcal{V}, k \in \mathcal{K}, s \in \mathcal{S}), \tag{4.10}$$

$$w_{ik}^s \leq \sum_{j\in\mathcal{N}} x_{ijk} \qquad (i \in \mathcal{V}, k \in \mathcal{K}, s \in \mathcal{S}), \tag{4.11}$$

$$\sum_{j\in\mathcal{N}\cup\mathcal{U}} z_{ijk}^s = \sum_{j\in\mathcal{N}\cup\mathcal{D}} z_{jik}^s + q_i^s - Qw_{ik}^s \quad (i \in \mathcal{N}, k \in \mathcal{K}, s \in \mathcal{S}), \tag{4.12}$$

$$\sum_{j\in\mathcal{N}} z_{djk}^s = 0 \qquad (d \in \mathcal{D}, k \in \mathcal{K}, s \in \mathcal{S}), \tag{4.13}$$

$$z^s \geq 0 \qquad (s \in \mathcal{S}), \tag{4.14}$$

$$w_{ik}^s \in \{0, 1\} \qquad (i \in \mathcal{N}, k \in \mathcal{K}, s \in \mathcal{S}). \tag{4.15}$$

Constraint (4.10) shows that the flows are nonzero only on planned routes and do not exceed the vehicle capacity. Constraint (4.11) specifies that a failure for producer i on route k can occur only if i belongs to route k. Constraint (4.12) defines when a failure occurs on a given node i. Constraint (4.13) ensures that vehicles depart from the depots with empty tanks.

As previously mentioned, our model is similar to a two-stage paradigm for a stochastic planning problem over a time horizon. However, in stochastic programming, there is a set of scenarios, and the size of the scenario set is based on the statistical dimension of the problem. Moreover, any scenario may occur, with a certain probability determined via a probability distribution, in any period of the horizon. Therefore, the cost of a plan is the sum of the first-stage cost, representing the routing costs and the fixed vehicle costs, and the second-stage cost, indicating the expected recourse cost with respect to different realizations of the probabilistic parameters. In our problem, the size of the period set is based on the number of seasons. We have a set of weights indicating the portion of the horizon represented by a given period. Moreover, each production level occurs only in a given period known a priori. This similarity between our formulation and the stochastic formulation may result in similar solution approaches. In the following section, we describe our solution approach based on the branch-and-price paradigm. We believe that this approach may be suitable for stochastic problems with a similar structure.

4.4 Solution approach

Since the cardinality of $\mathcal{R}$ is extremely large, the approach used to solve (4.4)–(4.7) is based on a branch-and-price algorithm. We apply a branch-and-bound scheme, and the lower bound at each node of the search tree is found by using column generation to solve the linear relaxation. The column generation procedure is based on iteratively solving a restricted master problem and one or more subproblems. The restricted linear master problem (RLMP) consists of the linear relaxation of the augmented model restricted to a subset of its variables. This subset simply contains those variables generated by solving the subproblems. Solving the RLMP using a linear programming solver, usually based on the simplex algorithm, results in primal and dual solutions. Each subproblem, often taking the form of an elementary shortest path problem with resource constraints (ESPPRC), is typically solved using an algorithm based on dynamic programming (DP) [see 55, 86]. The resulting negative reduced cost columns are then added into the RLMP and another iteration begins. The process stops when no subproblem is able to find new negative reduced cost variables for the RLMP.

We initialize the branch-and-price search tree by adding a set of initial columns to the RLMP at the root node. At each node of the tree, the lower bound is calculated through the iterative solution of the master problem and the subproblems, as described above. If the solution of the RLMP is integer, it is a feasible solution to the original problem, and the current incumbent is updated if necessary. If the solution is not integer, a branching procedure is applied to cut off the fractional part of the solution.

If the RLMP is infeasible, the node is fathomed. The optimal solution is the current incumbent solution after all the branches have been explored.

4.4.1 Literature review

In this section, we review research into different VRPs. This will allow us to take advantage of recent advances in exact methods for VRPs that are similar to our model. The most closely related problem is the consistent vehicle routing problem (ConVRP) proposed by [77]. In the ConVRP, customers with known demands receive service either once or with a predefined frequency over a multiple-day horizon. Frequent customers must receive consistent service, which is defined as visits from the same driver (vehicle) at approximately the same time throughout the planning horizon [148].

There is little literature on the MPVRP, in which decisions span multiple time periods. In most of these studies, customers request a service that could be done over a multi-period horizon [see 153, 4, 158, 5]. The MPVRP is closely related to the periodic vehicle routing problem (PVRP) in which the customers specify a service frequency and allowable combinations of visit days. A complete survey of the PVRP and its extensions can be found in Francis et al. [59]. The best-known algorithms for the PVRP are those of Cordeau et al. [34], Hemmelmayr et al. [80], and Vidal et al. [155].

In our problem, all the producers need to be served every period on a daily basis. Moreover, the definition of the periods is based on production variations. To the best of our knowledge, no prior work has considered this setting. Therefore, a review of the state of the art of the MPVRP would be irrelevant. Hence, this literature survey is divided into two parts: 1) a review of exact methods for VRPSDs, and 2) a review of advances in the solution of the ESPPRC, which is the core of exact methodologies based on branch-and-price, in the context of deterministic VRPs.

Vehicle routing problem with stochastic demands

The first study of the VRPSD was carried out by Tillman [150] for the multi-depot problem; the solution method was based on the savings heuristic. Dror and Trudeau [54] proposed other heuristics and showed the impact of the direction of travel on the expected travel cost, even in VRPSDs with symmetric distance matrices.

A few authors proposed exact algorithms. Séguin [140] and Gendreau et al. [66] presented integer L-shaped algorithms capable of solving instances with up to 70 nodes. Their Benders-decomposition-based approach follows the L-shaped algorithm of Laporte and Louveaux [103], which itself is an extension of the integer L-shaped method of Slyke and Wets [144]. More recently, Jabali et al. [88] proposed an integer L-shaped method based on the branch-and-cut scheme, in which some lower-optimality cuts are generated to eliminate feasible solutions. Moreover, lower-bounding functionals are used to improve the efficiency of the algorithm.

Christiansen and Lysgaard [29] introduced a new branch-and-price-based exact algorithm for the VRPSD. In their approach, the columns are generated through a label-correcting scheme on an exploded auxiliary graph containing several copies of each customer. More precisely, they create a new copy of each node for each quantity of product up to the capacity of the vehicle and each potential value of the demand. The graph is constructed assuming that all the labels arriving at a given node have collected the same product load. The customers' demands are given by Normal or Poisson distributions. This algorithm was recently reimplemented by Gauvin et al. [60] using state-of-the-art techniques for branch-cut-and-price.

The classical recourse, proposed in the context of the VRPSD, is based on a simple return to the depot to replenish (empty) the vehicle when a failure occurs. However, more sophisticated recourse actions have been proposed by various authors [160, 139, 3, 90].

In the next section, we describe the state of the art of solution methodologies for the deterministic ESPPRC.

Elementary shortest path problem with resource constraints

Many recent advances in branch-and-price for the classical deterministic variants of the VRP such as CVRP and VRPTW have provided promising results. Most propose efficient methodologies to optimally solve the subproblem, which takes the form of an ESPPRC. The elementarity condition adds an extra layer of complexity to the shortest path problem with resource constraints (SPPRC), itself an NP-hard problem. The most promising methodologies for the deterministic ESPPRC are based on one of the following strategies:

1. The elementarity conditions are completely or partially relaxed and the relaxation is iteratively tightened to obtain an optimal elementary solution.
2. The elementarity conditions are partially relaxed, and the optimality of the lower bound is sacrificed for the sake of time efficiency. After this relaxation, near-elementary routes are often generated in a fraction of the computational effort.

The decremental state-space relaxation (DSSR) proposed by Boland et al. [23] and Righini and Salani [133] is based on the first strategy. In this method, the elementarity conditions of the generated routes are initially relaxed, turning the problem into an SPPRC. After each iteration, using a state-space augmentation policy, restrictions are added to the problem to prevent the formation of cycles. Baldacci et al. [13] introduced a new state-space relaxation, called ng-path relaxation and based on the second strategy, to compute lower bounds to routing problems such as the CVRP and the VRPTW. It partitions the set of all possible paths ending at a generic vertex according to prespecified neighborhoods of graph vertices and a mapping function. The latter associates with each path a subset of the visited vertices that depends on the order in

which the vertices are visited. The subset associated with each ng-path is used to impose partial elementarity. This relaxation is particularly effective in computing lower bounds for the CVRP, the VRPTW, and the traveling salesman problem with time windows (TSPTW).

Martinelli [111] proposed a new ng-route pricing in which a DSSR technique is embedded into the ng-route relaxation. It consists of an ng-route relaxation procedure in which resources associated with the vertices' neighbors are initially deactivated. These neighborhoods are iteratively augmented based on a DSSR scheme to ensure the ng-feasibility of all the columns. The ng-path relaxation is based on a compromise between the computational efficiency of the procedure and the quality of the lower bound. Our solution methodology is built on this approach, which will be referred to as ng-route decremental state-space relaxation (ngR-DSSR) throughout this paper. In our implementation, we have proposed a new DSSR layer on top of the ngR-DSSR to guarantee the elementarity of the columns obtained. This modification is detailed in Section 4.6.2.

In the following sections we discuss the different modules of our branch-and-price-based methodology and extra features that help us to deal with the stochastic nature and the period-based formulation of the problem.

4.5 Master Problem

The master problem is in fact the linear relaxation of the set partitioning formulation (4.5)–(4.7). Often, the set partitioning formulation is transformed into a set covering formulation by relaxing constraints (4.5), at it can be shown the the optimal solution of the set covering formulation coincides with the optimal solution of the set partitioning formulation. However, in the case of this problem, the presence of constraints (4.6), in some special situations, may result in multiple visits to certain producers. Therefore, constraints (4.5) cannot be relaxed and should stay under equality form. The master problem becomes

(MP)

$$\min \quad \sum_{r \in \mathcal{R}'} (c_r + c_k) y_r + \mathcal{F}(x) \tag{4.16}$$

subject to

$$\sum_{r \in \mathcal{R}'} a_{ir} y_r = 1 \quad (i \in \mathcal{N}); \tag{4.17}$$

$$\sum_{r \in \mathcal{R}'} l_{pr}^{ref} y_r \geq D_p^{ref} \quad (p \in \mathcal{P}); \tag{4.18}$$

$$y_r \geq 0 \quad (r \in \mathcal{R}') \tag{4.19}$$

where constraint (4.17) specifies that there should be at least one visit to each producer. Moreover, $\mathcal{R}'$ represents all the existing variables of the model. The following section describes in detail the subproblems and the proposed solution method.

4.6 Subproblems

In a column generation method, the subproblems must be able to find master-problem variables that have negative reduced costs with respect to a given dual solution to the RLMP. We solve a subproblem for each plant $p \in \mathcal{P}$. The subproblem takes the form of a multi-period shortest path problem with resource constraints (MPESPPRC). Consider the following dual variables:
λ_i: free-signed dual variable of (4.17) for producer $i \in \mathcal{N}$;
μ_p: nonnegative dual variable of (4.18) for plant $p \in \mathcal{P}$.

In each subproblem, the objective function is given by

$$\min \quad Z = \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{V}} (c_{ij} - \lambda_i) x_{ij} - \sum_{p \in \mathcal{P}} l_{pr}^{ref} \mu_p + c_k + \mathcal{F}(x) \tag{4.20}$$

where Z represents the reduced cost of the generated column. Since the ESPPRC is an NP-hard problem in the strong sense [86], this is the most computationally demanding part of the branch-and-price process. Therefore, we propose a multi-phase column generation procedure. Dayarian et al. [44] have discussed the efficiency of a bi-level column generation process, consisting of heuristic DP (HDP) followed by exact DP (EDP), for the single-period variant of the problem.

The proposed multi-phase procedure uses three column generators, each finding negative reduced cost columns. In Section 4.8.2 we compare this procedure with the bi-level process. As is common for hybrid procedures, we initiate the procedure with a heuristic solver, here a tabu search (TS), which rapidly generates a subset of negative reduced cost columns. It is followed by two other column generators, a procedure based on HDP and one based on EDP. EDP, based on total enumeration, visits all possible permutations of the nodes to obtain all the negative reduced cost columns. HDP, a relaxation of EDP, explores the graph partially and more quickly but does not guarantee to generate all the columns. EDP is much more time-consuming than TS and HDP. These three generators are described in detail below.

4.6.1 Tabu search

TS [see 72, 25] is a powerful tool for difficult combinatorial optimization problems. It has been successfully used in hybrid algorithms to speed up column generation given a set of existing columns [see 48]. It starts with an initial solution that is improved through one or several neighborhood searches. This iterative approach selects the least-cost neighbor of the current solution given a set of allowable moves at each iteration. The new solution replaces the current solution even if it is not as good as that of the previous iteration. To prevent the algorithm from becoming trapped in a local minimum, a tabu list prohibits the reversal of the latest moves. The number of iterations for which a move is tabu is called the tabu tenure. To control the length of the tabu list, we can select the tabu tenure of each move randomly in the interval [$minTabu$, $maxTabu$], where $minTabu$ and $maxTabu$ are specified by the user.

Our tabu search, an iterative multi-start procedure, starts with an initial set of columns, representing the basic variables of the current RLMP. We perform a limited number of iterations, I_{max}, each covering a series of moves, on each variable to find new routes with negative reduced costs, knowing that the search region is restricted to the set of feasible solutions. Our neighborhood search allows the following moves:

Insertion: This move inserts new producer vertices in different positions of the route. We restrict it by inserting only a predefined set of producer vertices at each potential position. For a given position $\kappa + 1$, the successor set contains the $nbSucc$ producer vertices closest to vertex i at position κ. The closeness of vertex i is found from the value of $c_{ij} - \lambda_j$ for all $j \in \mathcal{N}$.

Deletion: This simple move evaluates a route after the removal of a vertex. All the possible deletions in a route are evaluated one by one, and a removed vertex is replaced before the next deletion.

Swap: This move swaps the position of two producers in a route.

To maintain feasibility, each route derived from a move should satisfy the routing constraints and the branching constraints (discussed in Section 4.7). Clearly, the former check is not needed for deletions. The procedure stops when either the predefined maximum number of iterations I_{max} is attained or no new route with a negative reduced cost is obtained for I_{stop} iterations, where I_{max} and I_{stop} are predefined values.

New routes are checked to ensure that they satisfy inequality (4.3), and then their expected costs and reduced costs, with respect to the different periods, are calculated as explained above. This procedure tends to generate a large number of columns with negative reduced costs, especially in the first nodes of the branch-and-price tree where the dual variables are larger and the branching constraints are less restrictive. If we add all the columns to the RLMP we increase the size of the model and therefore the computational time necessary. We instead apply a procedure that attempts to select a smaller number of generated routes that are not dominated by the others. Let χ_1 and χ_2 be respectively the producers visited on two routes r_1 and r_2 obtained through TS, and C_1 and C_2 their reduced costs. The selection procedure ignores r_2 if the following conditions hold:

(a) $C_1 \leq C_2$,

(b) $\chi_1 \subseteq \chi_2$.

This approach allows us to eliminate a large number of routes that may not improve the solution. The remaining columns are then added to the RLMP.

4.6.2 Exact dynamic programming

As mentioned in Section 4.4, an MPESPPRC is solved for each plant. We solve it using a DP-based procedure, *ng*-route decremental state-space relaxation. In DP-based approaches, new paths, encoded by labels, are systematically built from a source node toward a sink node. To simultaneously construct the routes from all the depots toward a given plant, we start the labeling at the plant. In other words, in our implementation, the source node is a plant and a copy of the same plant plays the role of the sink node. The route construction process begins with a label associated with a plant that is then extended to the depots, to the producers, and back to the plant. The labels are extended in feasible directions via extension functions. Define the different components of a label $\sigma = (L, F, C, S, \chi)$ as:

L**:** vehicle load in the reference period,

F**:** vector of size $|\mathcal{S}|$ indicating periods in which a failure has occurred,

C**:** reduced cost of the partial path with respect to different periods,

Nb**:** number of unreachable nodes,

χ**:** set containing the nodes $j \in \mathcal{N}$ unreachable from the current label.

The real load of each period can be obtained from the product of its production coefficient and the load in the reference period. Therefore, one may track only the collected load in the reference period. Suppose that L_i, F_i, and C_i are the loads, failures, and cost components of a label σ_i associated with producer $i \in \mathcal{N}$. Extending this label along the arc $(i, j) \in A$ produces a new label σ_j associated with producer j, with components L_j, F_j, and C_j. The extension functions Ext^L_{ij}, Ext^F_{ij}, and Ext^C_{ij} are given by

$$Ext^L_{ij} : L + q^{ref}_j$$

$$Ext^F_{ij} : \begin{cases} F_{js} = 1, & \text{if } F_{is} = 0 \text{ and } L.P_s > Q \\ F_{js} = F_{is}, & \text{otherwise} \end{cases} \qquad s \in \mathcal{S}$$

$$Ext^C_{ij} : \begin{cases} C + C_{ij} - \lambda_j + \sum_{s \in \mathcal{S}} W_s (F_{js} - F_{is}) C_{jp}, & \text{if } j \in \mathcal{N} \\ C + C_{ij} + L\mu_j, & \text{if } j \in \mathcal{P}. \end{cases}$$

To increase the efficiency of the algorithm, we use a dominance subalgorithm to disable paths that are not useful either for building a Pareto-optimal set of paths or for being extended into Pareto-optimal paths. For a given set $\mathcal{M} \subset \mathbb{R}^R$, an element $m \in \mathcal{M}$ is Pareto-optimal if $x \not\leq m$ holds for all $x \in \mathcal{M}$, $x \neq m$. A disabled label is

neither extended nor compared with other new labels on the node. Dominance rules identify the paths that are not useful for defining the set of Pareto-optimal solutions, i.e., the paths that are not part of the Pareto-optimal set and whose extension cannot lead to paths in the Pareto-optimal set. The structure of the dominance rules is problem-dependent and is related to the path structural constraints. Many efficient dominance rules have been proposed by different researchers for elementary and not necessarily elementary SPPRCs [see 86]. We will now describe a special phenomenon that occurs when extending labels using the above extension functions in the case of the MPESPPRC.

Nonmonotonic resource consumption

In almost all ESPPRC cases solved using the DP-based label-correcting procedure, the resource consumption on the arcs can be shown to be monotonic. The monotonicity property ensures that the resource consumption is identical along a given common extension from two different labels on a given node. However, in the cost extension function presented above, based on our recourse structure, the reduced cost of a label depends on the positions where failures occur in each period. Therefore, the cost resource consumption is not necessarily monotonic for the same extension from two different labels on a given node. Thus, two labels with the possibility of future failure in at least one of the periods cannot be compared using the dominance rules for the classical ESPPRC [see 55]. To see this, consider the following example.

Example 1. *Consider a graph with five producers, one depot* (D)*, a plant* (P)*, and a vehicle with capacity* $Q = 8$*, as depicted in Figure 4.2. For a given period, each producer's supply is given in parentheses. Two labels* σ_1 *and* σ_2 *representing the partial paths* $(D, 1, 3)$ *and* $(D, 1, 2, 3)$ *are in the label list of node* 3*. According to the classical dominance rules,* σ_2 *is dominated by* σ_1*. However, for the common extension* $(4, 5)$ *from node* 3*,* σ_1 *encounters a failure in node* 5*, while* σ_2 *encounters a failure in node* 4*. The failure cost from node* 5 *(a return trip to the plant) covers the cost difference between* σ_2 *and* σ_1 *plus the failure cost from node* 4 *associated with* σ_2*. This example shows that the classical conditions do not take into account nonmonotonic consumption of the cost resource.*

Two labels can be compared using the classical dominance rules in the following cases:

Case 1: $L_1 = L_2$;

Case 2: For all $s \in \{s \in \mathcal{S} | P_s > 1\}$, $F_{s1} = F_{s2} = 1$.

Two labels satisfying Case 1 will always encounter simultaneous failures, while two labels satisfying Case 2 will not face failure in the remainder of the label-extension procedure. In all other situations, relying on the classical dominance rules may result

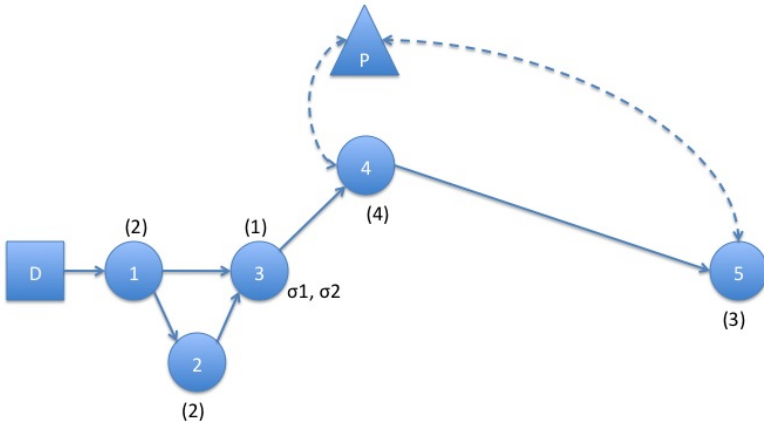

Figure 4.2: Example 1

in the discarding of partial paths that seem more costly but will become beneficial later when they encounter a less costly failure.

Note that the nonmonotonicity is directly related to the nature of the recourse. In applications where the recourse cost is based on a fixed penalty that is independent of the location where failure occurs, the resource consumption is monotonic.

Bounding the recourse cost

To deal with nonmonotonic resource consumption, we must consider a look-ahead condition based on the labels' potential failures. For a given label σ_1, let $\widehat{\mathcal{S}}_1 = \{s \in \mathcal{S} | P_s > 1, F_{s1} = 0\}$ and $\widehat{\mathcal{V}}_1 = \{i \in \mathcal{V} | i \notin \chi_1\}$. The following extra condition is required to prevent the incorrect dominance of σ_2 by σ_1:

$$C_1 + \sum_{s \in \widehat{\mathcal{S}}_1} W_s[\max_E(Cr) - \min_E(Cr)] - \mu_p(L_2 - L_1) \leq C_2 \quad (\forall E \subseteq \widehat{\mathcal{V}}_2)$$

$$\Rightarrow \quad C_1 + \sum_{s \in \widehat{\mathcal{S}}_2} W_s[\max_{\widehat{\mathcal{V}}_2}(Cr) - \min_{\widehat{\mathcal{V}}_2}(Cr)] + \sum_{s \in \widehat{\mathcal{S}}_1 \setminus \widehat{\mathcal{S}}_2} W_s[\max_{\widehat{\mathcal{V}}_2}(Cr)] - \mu_p(L_2 - L_1) \leq C_2$$

where Cr represents the recourse cost corresponding to a return trip to the plant. Moreover, $\min_{\widehat{\mathcal{V}}_2}(Cr)$ and $\max_{\widehat{\mathcal{V}}_2}(Cr)$ represent the minimum and maximum potential recourse costs with respect to the set of nodes reachable by σ_2. It should be noted that $\min(Cr)$ and $\max(Cr)$ are calculated only for $\widehat{\mathcal{V}}_2$, since $\chi_1 \subseteq \chi_2$ and therefore $\widehat{\mathcal{V}}_2 \subseteq \widehat{\mathcal{V}}_1$. In fact, $\sum_{s \in \widehat{\mathcal{S}}_2} W_s[\max_{\widehat{\mathcal{V}}_2}(Cr) - \min_{\widehat{\mathcal{V}}_2}(Cr)] + \sum_{s \in \widehat{\mathcal{S}}_1 \setminus \widehat{\mathcal{S}}_2} W_s[\max_{\widehat{\mathcal{V}}_2}(Cr)]$ represents an upper bound on the failure cost difference that may occur for any common extension from σ_1 and σ_2.

Therefore, for $\sigma_1 = (L_1, F_1, C_1, Nb_1, \chi_1)$ to dominate $\sigma_2 = (L_2, F_2, C_2, Nb_2, \chi_2)$, we must have:

(a) $L_1 \leq L_2$;

(b) $C_1 - \mu_p(L_2 - L_1) \leq C_2$;

(c) $Nb_1 \leq Nb_2$;

(d) $\chi_1 \subseteq \chi_2$;

(e) $C_1 + \sum_{s \in \widehat{\mathcal{S}}_2} W_s[\max_{\widehat{\mathcal{V}}_2}(Cr) - \min_{\widehat{\mathcal{V}}_2}(Cr)] + \sum_{s \in \widehat{\mathcal{S}}_1 \setminus \widehat{\mathcal{S}}_2} W_s[\max_{\widehat{\mathcal{V}}_2}(Cr)] - \mu_p(L_2 - L_1) \leq C_2$.

Taking into account the nonmonotonic cost resource consumption by including condition (e) prevents any incorrect label discarding during the label-correcting procedure. However, the bound $\max_{\widehat{\mathcal{V}}_2}(Cr) - \min_{\widehat{\mathcal{V}}_2}(Cr)$ could be so large that satisfying condition (e) is almost impossible, and consequently little domination occurs. Based on our experiments, the low rate of dominance and the costly verification of conditions (a)–(e), depending on the structure of the graph, sometimes make it impossible to perform a single labeling iteration in a reasonable time. This is directly related to the combinatorial nature of solving the MPESPPRC using the label-correcting procedure. We now explain how to efficiently manage the labels on the graph by potentially keeping a larger number of labels on each node to significantly accelerate the verification of dominance.

Matrix-based implementation

As mentioned in Section 4.6.2, on a given node of the graph, labels with equal loads following a common path extension will always encounter simultaneous failures. Further to this property, at each node of the graph, following Denardo and Fox [46], one may create buckets that store labels with the same loads. We implement a matrix-based data structure of size $|\mathcal{N}| \times Q$ to store the labels on the graph. Each cell (i, q) of this matrix contains a sorted list (w.r.t. the reduced cost) of all the labels arriving at node $i \in \mathcal{N}$ with collected quantity equal to q, for $0 \leq q \leq Q$. The advantage of this data structure is that since each cell stores labels with the same load, the labels in a cell can be compared using the classical dominance rules (see Case 1 of Section 4.6.2). More precisely, when we compare two labels from the same cell, the verification of condition (e) becomes unnecessary. The disadvantage of the data structure is the large number of labels stored. On a given node i, certain labels in cell (i, q) could dominate labels in the cells (i, q') for $q' > q$. We do not detect this because the domination verification considers only the labels in the same cell.

With this approach, σ_2 in a given cell is dominated by σ_1 in the same cell if conditions (b)–(d) hold. Moreover, condition (b) can be simplified to

(b) $C_1 \leq C_2$.

It is worth mentioning that all the labels on a plant node can be compared since there will be no further failure. Therefore, applying the dominance conditions (a)–(d) to labels on plant nodes allows us to identify a large number of dominated labels.

Maintaining elementarity in the label-correcting procedure

For a given label σ, $\chi \subseteq \mathcal{N}$ contains the nodes unreachable from the label, and Nb is the number of these nodes. DSSR and the ng-route relaxation aim to keep this set as small as possible without losing the elementarity of the routes.

Let $\mathcal{V}_r$ be the set of producers visited by the partial path r. Moreover, for each customer $i \in \mathcal{N}$, let $\mathcal{N}_i \subseteq \mathcal{N}$ be the so-called *original neighborhood* of producer i; it is a set of producers selected according to a neighborhood criterion for producer i. For the label σ associated with a given partial path $r = (d, i_1, \ldots, i_n)$ we can define a set $\Pi(r) \subseteq \mathcal{V}_r$ containing all the prohibited extensions from producer i_n. According to the ng-route relaxation rules, the set $\Pi(r)$ is defined as

$$\Pi(r) = \{i_j \in \mathcal{V}_r | i_j \in \bigcap_{k=j+1}^{n} \mathcal{N}_{i_k}, j = 1, \ldots, n-1\} \cup \{i_n\}. \tag{4.21}$$

The ngR-DSSR procedure starts with *applied neighborhood* sets in which the members of the original neighborhoods are present but initially deactivated. At the end of each iteration, if the best route found does not contain any cycles, it is added to the RLMP. If it contains at least one cycle that violates the ng-rules according to the original neighborhoods, the applied neighborhoods of all the nodes in the cycle are augmented by reactivating the resource corresponding to the node on which the cycle occurred.

In our implementation, since the ng-rules do not guarantee the elementarity of the routes, a new layer of the DSSR is added to the solution procedure. In this layer, if a cycle on the best route found respects the ng-rules according to the original neighborhoods, the node on which the cycle occurred is recognized as a critical node. A new iteration of the labeling procedure begins by prohibiting cycles on critical nodes. The recognition of a node as critical is equivalent to augmenting all the applied neighborhoods of all the nodes by an active resource for the corresponding critical node. This additional layer ensures the elementarity of all the columns, which provides a better lower bound. In our implementation, each label contains the components L, F, and C as described earlier, while Nb and χ are decomposed into the four following components:

Π: set of prohibited immediate extensions of the corresponding partial path;

Nb_{ng}: size of Π;

ϕ: set of critical nodes unreachable from the current label;

Nb_{SSR}**:** number of unreachable critical nodes.

Therefore, for the domination of $\sigma_2 = (L_2, F_2, C_2, Nb_{ng2}, \Pi_2, Nb_{SSR2}, \phi_2)$ by $\sigma_1 = (L_1, F_1, C_1, Nb_{ng1}, \Pi_1, Nb_{SSR1}, \phi_1)$, we require

(a) $L_1 \leq L_2$,

(b) $C_1 - \mu_p(L_2 - L_1) \leq C_2$,

(c) $Nb_{ng1} \leq Nb_{ng2}$,

(d) $\Pi_1 \subseteq \Pi_2$,

(e) $Nb_{SSR1} \leq Nb_{SSR2}$,

(f) $\phi_1 \subseteq \phi_2$.

In a matrix-based implementation, we may omit condition (a) and drop the second term in the left-hand side of (b).

4.6.3 Heuristic dynamic programming

To speed up the generation of the negative reduced cost columns, we implement a relaxed version of the labeling procedure described above. The relaxations are based on weakening the dominance rules or ignoring or simplifying the assumptions or the problem characteristics. They may allow us to discard more labels more rapidly and therefore accelerate the column generation. Our HDP is based on the following relaxations:

1. Ignoring the nonmonotonic behavior of the cost resource consumption;
2. Dropping conditions (c)–(f).

Relaxation 1 allows us to simply consider the deterministic dominance conditions. Therefore, the matrix-based label storing becomes unnecessary. Relaxation 2 makes the dominance much easier. Note that the extension of labels throughout the graph follows the same extension functions while the storage of the labels on each node and the dominance conditions have been modified. This allows us to price out a significant portion of the negative reduced cost columns. However, as a result of the two relaxations we may miss some negative reduced cost columns.

4.6.4 Skeleton of the multi-phase subproblem algorithm

Algorithm 2 describes the column generation procedure for each node of the search tree up to the branching phase. Note that $NBCOL$ is the number of newly generated columns with negative reduced costs.

As shown in Algorithm 2, the TS-based column generator is called while it finds negative reduced cost columns. We then call the HDP-based column generator. After each HDP iteration, if any column is priced out, the algorithm returns to TS. When both TS and HDP fail to add a new column, the EDP-based generator is called. The main aim of this multi-phase procedure is to reduce the number of calls to EDP, which is the bottleneck.

Algorithm 2 Solution of multi-phase subproblem

```
repeat
  repeat
    NBCOL = 0;
    TABU SEARCH( );
    Update NBCOL;
    Update and Solve the RLMP;
  until NBCOL == 0
  HEURISTIC DYNAMIC PROGRAMMING( );
  Update NBCOL;
  Update and Solve the RLMP;
until NBCOL == 0
repeat
  EXACT DYNAMIC PROGRAMMING( );
  Update NBCOL;
  Update and Solve the RLMP;
until NBCOL == 0
```

4.7 Strong branching

The column generation does not guarantee the integrality of the solution. Therefore, as described in Section 4.3, we use a branching scheme to cut off the fractional part of the solution. It is crucial to reduce the number of search tree nodes to reduce the number of calls to EDP. Therefore, we carefully choose the fractional variable to branch on. The strong branching strategy applied in this paper is a multilevel branching scheme based on the branching by plant assignment (BPA) proposed by Dayarian et al. [44] and the well-known strategy based on flow variables. In the BPA, on one branch, a producer is assigned to a specific plant, and on the other branch, that producer is removed from the subproblem associated with the plant. Note that assigning producer i to plant p means that all the routes visiting producer i must deliver to plant p. These decisions are easily imposed in the subproblems, and the existing columns that do not respect

the branching decision are removed before we solve the new RLMP. When no branching candidate is identified by the BPA, we branch on a flow variable.

The variable branched on often has a significant impact on performance, especially in the lower levels of the search tree. At these levels, it is reasonable to use a more sophisticated selection.

Strong branching may allow us to find better lower bounds faster. It uses a candidate set of variables and determines which provides the best lower-bound progress before actually branching. For each candidate, it solves the linear relaxations of the two child nodes that would be created by the branch. If all the fractional variables are included in the candidate set, the locally best branch candidate can be identified. However, this approach may be computationally expensive. To accelerate the process, we may evaluate only a subset of the fractional variables. We evaluate the candidates using the following score function, based on the function proposed by Linderoth and Savelsbergh [108]:

$$score = \gamma \max(\Delta^R, \Delta^L) + (1 - \gamma) \min(\Delta^R, \Delta^L), \tag{4.22}$$

where Δ^R and Δ^L respectively represent the increase in the objective function value before column generation in the right and left children of the node, if this candidate is branched on. The function can be calibrated using different values for the parameter $\gamma \in [0, 1]$; in our experiments, $\gamma = 0.25$ worked well. We select the three best candidates with respect to this score function. We then choose among these candidates using a procedure that depends on the depth of the node in the tree. If the node's depth is less than a prespecified parameter $dep*$, we call TS and HDP to generate as many columns as possible for each candidate, and we select the best candidate according to the score function. If the node's depth is greater than $dep*$, we call only the TS column generator and otherwise proceed as above. Our experiments indicate that the best results are obtained when $dep* = 10$. The selected branching variable j satisfies

$$j = \arg\max\{\gamma \max(\Delta^R, \Delta^L) + (1 - \gamma) \min(\Delta^R, \Delta^L)\}. \tag{4.23}$$

4.8 Computational results

To assess the performance of our algorithm, we carried out a computational study of test problems with different characteristics. Section 4.8.1 explains the characteristics of these problems, and Section 4.8.2 discusses the contribution of the different column generators. The results from the branch-and-price procedure are discussed in Section 4.8.3. In Section 4.8.4, we study the value of the multi-period approach.

The tests were run on computers with a 2.67 GHz processor and 24 GB of RAM. The linear programs were solved using Cplex 12.2 and the time limit was set to 18000 s.

4.8.1 Test problems

We generated the instances using the generator of Dayarian et al. [44] with some modifications to adapt it to the structure of our problem. As shown in Table 4.1, Dayarian et al. [44] generated four groups of instances with different plant positions (inside or outside) and time windows (narrow or wide). The *inside* plants are placed in the central region of the graph, and the *outside* plants are placed in the outlying region.

Table 4.1: Four problem classes generated by Dayarian et al. [44]

Class Number	Plant Location	Time Windows
pr01	inside	narrow
pr02	inside	wide
pr03	outside	narrow
pr04	outside	wide

For the tests in this paper, we generate instances based on pr03, ignoring the time windows. We add parameters defining the characteristics of the periods to each instance. We assume that the data for each instance represent the production levels in the reference period ($P_s = 1$). We then consider a set of period distributions with the SRT ranging from 20% to 60%. We recall that the SRT indicates the percentage of days without a failure. The W_s and P_s values of these different distributions are shown in Table 4.2. We generate five versions of each size combination (number of depots, number of plants, number of producers). We consider all the period distributions for every instance.

Table 4.2: Probability and production-level distribution of the periods

# periods	Type 1		Type 2		Type 3		Type 4		Type 5	
	P_s	W_s%	P_s	W_s%	P_s	W_s%	P_s	W_s%	P_s	W_s%
	1.00	60	1.00	50	1.00	40	1.00	30	1.00	20
4	1.30	20	1.30	25	1.20	35	1.10	30	1.10	40
	1.50	10	1.50	15	1.35	20	1.20	25	1.30	30
	1.70	10	1.70	10	1.50	15	1.40	15	1.70	10
	P_s	W_s%	P_s	W_s%	P_s	W_s%	P_s	W_s%	P_s	W_s%
	1.00	60	1.00	50	1.00	40	1.00	30	1.00	20
5	1.30	15	1.30	20	1.20	25	1.10	25	1.10	35
	1.50	15	1.50	15	1.35	20	1.20	20	1.20	25
	1.70	5	1.70	10	1.50	10	1.40	15	1.40	15
	1.90	5	1.90	5	1.65	5	1.70	10	1.70	5

4.8.2 Linear relaxation

To evaluate the TS column generator, we ran a sample of the instances with and without it. Table 4.3 gives the value of the different parameters used; these values were obtained

through a series of trial-and-error tests.

Table 4.3: Parameter values used in tabu search

Parameter	Tuned value
$[minTabu, maxTabu]$	[3, 8]
I_{max}	25
I_{stop}	5
$nbSucc$	0.5 $\|\mathcal{N}\|$

Table 4.4 reports the results for the linear relaxation: the computational time (in seconds) and the number of iterations of HDP and EDP. In 34 of the 40 instances, TS has improved the computational efficiency. This is often accompanied by a decrease in the overall number of HDP and EDP iterations. It is interesting to compare this with the behavior of the single-period variant of the problem [see 44], where the HDP and EDP were more efficient in the absence of TS. This indicates the extra complexity of the multi-period approach.

4.8.3 Integer solution

For each group of five instances of the same size, Tables 3.3 and 4.8 give:

 # producers: number of producers in each group.

 # size comb.: number of depots (D) and plants (P), e.g., "$2D3P$" indicates two depots and three plants.

 # periods: number of periods (four or five) in each instance.

period type: type of period distribution (see Table 4.2).

 # opt. sol.: number of optimal solutions found in each group.

 # int. sol.: number of integer solutions found without achieving optimality.

CPU1: average computational time for every group with a given distribution, whether or not the optimal solution is found. The computational time for unsolved instances is set to the time limit.

CPU2: average computational time for solved instances.

opt. gap: The optimality gap is zero for a solved instance, infinity for an instance with no integer solution, and otherwise calculated via

$$\text{optimality gap} = \frac{\text{(best upper bound - best lower bound)}}{\text{best lower bound}}. \tag{4.24}$$

Table 3.3 gives the results for the instances with fifteen producers, and Table 4.8 gives the results for twenty producers.

The results indicate that the algorithm is sensitive to an increase in the number of producers. For fifteen producers, the variations in terms of the number of solved instances are not significant because optimality was usually reached. However, the algorithm found the optimal solution for all instances with three plants, while for the instances with two plants, there is always one for which the algorithm could not close the gap within the time limit.

This variation is more noticeable for the instances with twenty producers. The results also show that the structural configuration is generally more important than the number and distribution of the periods. This can be seen in Table 4.8: changing the number of periods or their distributions in the instances with 2D3P or 3D3P has a minor impact on performance (number of optimal solutions in cases with 4 or 5 periods for 2D3P: 21 vs. 20 and for 3D3P: 18 vs. 18). These results also show that the lower the weight of the reference period (e.g., T5), the easier the problem. Moreover, for instances with twenty producers, the algorithm performed better when the numbers of plants and depots were larger. An increase in the number of plants is equivalent to an increase in the number of subproblems and branching candidates. However, in instances with more plants, the average cost of failure is lower, and thus the impact of the recourse component of the route cost is lower. Another explanation is based on the branching strategy. In the first layers of the search tree, we attempt to assign the producers to plants. An increase in the number of plants reduces the number of producers per plant, leading to smaller subproblems per plant. This explains the higher number of successes for instances with three plants compared to instances with the same number of producers but two plants.

4.8.4 Value of the multi-period solution

To evaluate the multi-period approach, we compare the multi-period solutions (MPS) to the solutions of

1. Worst case scenario (WCS): WCS represents the most conservative strategy. A deterministic VRP is solved by considering the highest production level. The routing cost is higher and there is no recourse cost.
2. Reference Period (RP): RP considers only the reference period. This is similar to solving a chance-constrained program when the service quality is fixed to the weight of the reference period.

As mentioned in Section 4.3, we calculate the route cost based on 1) the fixed vehicle costs, 2) the routing costs, and 3) the recourse costs. Let these elements be $c_f(x)$, $c(x)$, and $\mathcal{F}(x)$, respectively, and let the optimal solution for the multi-period formulation

be x_{MPS}. For any feasible solution x, we have

$$c_f(x_{MPS}) + c(x_{MPS}) + \mathcal{F}(x_{MPS}) \leq c_f(x) + c(x) + \mathcal{F}(x). \tag{4.25}$$

Since the fixed cost of the vehicles is high compared to the routing cost, RP and MPS use the same (minimum) number of vehicles. However, the routing cost in the RP case provides a lower bound on the routing cost of the MPS:

$$c(x_{RP}) \leq c(x_{MPS}). \tag{4.26}$$

The multi-period formulation finds a solution x_{MPS} that minimizes the corresponding costs in the first and second stages, with respect to the different production levels (inequality (4.25)). We also have

$$c(x_{RP}) \leq c(x_{MPS}) \leq c(x_{WCS}), \tag{4.27}$$

$$\mathcal{F}(x_{WCS}) \leq \mathcal{F}(x_{MPS}) \leq \mathcal{F}(x_{RP}). \tag{4.28}$$

Therefore, solving MPS rather than RP also leads to a lower recourse, i.e., smaller modifications to the *a priori* plan.

We ran the problems of Section 4.8.2 using WCS and RP. The routes obtained were then costed assuming different periods with different production levels and weights. Table 4.9 shows the results for MPS, WCS, and RP. Column ST gives the solution status for each of the procedures, where $T_{max} = 18000\,s$ and "✓" indicates an optimal solution, "•" indicates a solution that is not necessarily optimal, and "–" indicates no integer solution. The comparison is based on computational time and solution quality in terms of 1) the total solution cost ($c_f(x) + c(x) + \mathcal{F}(x)$) and 2) the recourse cost part ($\mathcal{F}(x)$). The cost gaps for WCS and RP are calculated via the following formulas and are reported in Column "gap%" of Table 4.9:

$$gap_{(WCS)} = \frac{[c_f(x_{WCS}) + c(x_{WCS}) + \mathcal{F}(x_{WCS})] - [c_f(x_{MPS}) + c(x_{MPS}) + \mathcal{F}(x_{MPS})]}{c_f(x_{MPS}) + c(x_{MPS}) + \mathcal{F}(x_{MPS})}, \tag{4.29}$$

$$gap_{(RP)} = \frac{[c_f(x_{RP}) + c(x_{RP}) + \mathcal{F}(x_{RP})] - [c_f(x_{MPS}) + c(x_{MPS}) + \mathcal{F}(x_{MPS})]}{c_f(x_{MPS}) + c(x_{MPS}) + \mathcal{F}(x_{MPS})}. \tag{4.30}$$

The gaps based on the recourse cost are computed only for RS. They are reported in Column "recourse gap%" and obtained via the following formula:

$$\text{Recourse gap (RP)} = \frac{\mathcal{F}(x_{RP}) - \mathcal{F}(x_{MPS})}{\mathcal{F}(x_{MPS})}. \tag{4.31}$$

WCS is robust to potential variations in the production levels, but it is significantly more expensive than MPS. In WCS, no failure occurs ($\mathcal{F}(x_{WCS}) = 0$), but more vehicles are required. For RP, the total cost gaps are not large, but the large recourse gaps and the comparable computational times indicate that it is preferable to solve MPS.

4.9 Conclusions

We have explored a vehicle routing problem in which producers' supplies vary on a seasonal basis. Moreover, the variations between producers are strongly correlated. We have proposed a multi-period model based on a set partitioning formulation. Our solution method is based on a branch-and-price algorithm; new columns are generated through a hybrid multi-phase column generation process. We have also introduced the issue of nonmonotonic resource consumption in the DP-based subproblem algorithm. We use a strong branching rule to find integer solutions.

Our solution method was able to solve problems with up to twenty producers and five periods. Real-life applications, such as the DTPQ, involve hundreds of producers. However, smaller problems with small districts may be solved by our algorithm. For larger problems, we plan to develop metaheuristic-based approaches to obtain good solutions in a reasonable computational time. The results obtained in this paper will help to evaluate future approaches.

We also plan to develop a stochastic formulation that considers the intra-period variation. This will increase the applicability of our formulation, since uncertainties exist in many real-life planning applications.

acknowledgements

Partial funding for this project was provided by the Natural Sciences and Engineering Research Council of Canada (NSERC), through its Industrial Research Chair, Collaborative Research and Development, and Discovery Grant programs. We also received support from the Fonds de recherche du Québec - Nature et technologies (FRQ-NT) through its Team Research Project program, and from the EMME/2-STAN Royalty Research Funds (Dr. Heinz Spiess). We also gratefully acknowledge the support of Fonds de recherche du Québec through infrastructure grants and the support of Calcul Québec and Compute Canada through access to their high-performance computing infrastructure.

Table 4.4: Comparing results of linear relaxation with and without TS

				With TS				Without TS		
# producers	size comb.	# periods	period type	CPU	# TS	# HDP	# EDP	CPU	# HDP	# EDP
15	2D2P	4	T1	2.72	18	13	12	1.55	27	6
			T2	11.22	37	21	12	21.02	27	16
			T3	2.19	34	20	8	7.05	23	14
			T4	5.65	38	25	10	8.96	23	15
			T5	5.77	46	22	11	19.65	33	13
		5	T1	4.57	29	21	9	10.45	31	11
			T2	5.05	37	27	8	6.46	29	11
			T3	2.12	35	21	6	6.9	24	13
			T4	17.19	32	19	20	20.09	26	18
			T5	5.91	26	17	12	4.77	20	13
	2D3P	4	T1	2.57	20	12	6	6.09	17	9
			T2	1.23	27	18	3	4.29	19	10
			T3	6.89	28	19	8	5.87	23	6
			T4	18.39	33	17	10	121.3	15	13
			T5	47.17	30	18	14	43.58	17	15
		5	T1	36.38	31	18	12	44.47	19	15
			T2	37.16	25	17	12	121.32	20	18
			T3	4.35	33	19	3	6.56	22	7
			T4	1.99	25	16	5	3.32	23	7
			T5	6.57	34	16	9	5.38	19	7
	3D2P	4	T1	3.34	28	20	9	7.33	22	13
			T2	16.67	32	18	14	23.45	23	16
			T3	2.71	37	24	6	12.02	27	15
			T4	4.66	38	18	5	5.09	26	7
			T5	17.52	37	16	14	20.82	30	14
		5	T1	11.22	33	22	10	29.08	29	14
			T2	4.76	30	21	6	5.42	26	8
			T3	3.43	40	22	10	4.17	28	11
			T4	8.99	34	21	10	27.1	26	16
			T5	5.69	38	26	9	7.03	30	14
	3D3P	4	T1	20.37	36	21	11	36.81	21	17
			T2	22.5	47	25	9	97.66	21	15
			T3	7.12	27	13	12	5.41	19	12
			T4	6.57	29	17	8	8.69	20	12
			T5	28.19	39	17	9	47.76	17	13
		5	T1	65.41	44	21	8	154.69	17	14
			T2	12.92	25	18	10	23.76	19	12
			T3	4.87	33	17	8	8.03	19	13
			T4	21.1	39	14	10	55.14	23	17
			T5	18.97	33	20	11	25.47	18	17
Avg.				12.80	33	19	9	26.85	23	13

Table 4.5: Computational results for instances with 15 producers

# producers	size comb.	# scen.	scen. type	# opt. sol.	# int. sol.	CPU1	CPU2	opt. gap %
15	2D2P	4	T1	4	1	3642.2	52.8	19.1
			T2	4	1	3648	60.1	18.0
			T3	4	1	3659.5	74.4	15.3
			T4	4	1	3671.2	89	17.2
			T5	4	1	3703.8	129.8	15.1
		5	T1	4	1	3654.3	67.9	19.5
			T2	4	1	3646.9	58.7	17.2
			T3	4	1	3660.1	75.1	15.8
			T4	4	1	3671.1	88.9	15.8
			T5	4	1	3696.7	120.9	16.0
	2D3P	4	T1	5	0	988.5	988.5	0.0
			T2	5	0	947.3	947.3	0.0
			T3	5	0	816.6	816.6	0.0
			T4	5	0	765.2	765.2	0.0
			T5	5	0	838.3	838.3	0.0
		5	T1	5	0	1032.6	1032.6	0.0
			T2	5	0	1063.4	1063.4	0.0
			T3	5	0	882.4	882.4	0.0
			T4	5	0	860.5	860.5	0.0
			T5	5	0	790	790	0.0
	3D2P	4	T1	3	2	7273.8	123.1	25.5
			T2	3	2	7294.3	157.2	24.0
			T3	3	2	7323	205	22.3
			T4	3	2	7354.5	257.5	23.8
			T5	3	2	7296.1	160.2	23.6
		5	T1	3	2	7371	285.1	25.1
			T2	3	2	7295.4	159	24.1
			T3	3	2	7313.8	189.7	23.2
			T4	3	2	7263.3	105.5	22.6
			T5	3	2	7266.7	111.2	22.3
	3D3P	4	T1	5	0	1457.9	1457.9	0.0
			T2	5	0	1383.1	1383.1	0.0
			T3	5	0	1180.4	1180.4	0.0
			T4	5	0	1189.3	1189.3	0.0
			T5	5	0	1076	1076	0.0
		5	T1	5	0	1325.1	1325.1	0.0
			T2	5	0	1377.2	1377.2	0.0
			T3	5	0	1296.4	1296.4	0.0
			T4	5	0	1247.9	1247.9	0.0
			T5	5	0	1280.7	1280.7	0.0

Table 4.6: Computational results for instances with 20 producers

# producers	size comb.	# scen.	scen. type	# opt. sol.	# int. sol.	CPU1	CPU2	opt. gap %
20	2D2P	4	T1	0	0	T_{max}	T_{max}	∞
			T2	0	0	T_{max}	T_{max}	∞
			T3	0	0	T_{max}	T_{max}	∞
			T4	0	0	T_{max}	T_{max}	∞
			T5	0	0	T_{max}	T_{max}	∞
		5	T1	0	0	T_{max}	T_{max}	∞
			T2	0	0	T_{max}	T_{max}	∞
			T3	0	0	T_{max}	T_{max}	∞
			T4	0	0	T_{max}	T_{max}	∞
			T5	0	0	T_{max}	T_{max}	∞
	2D3P	4	T1	3	2	11877.8	7796.3	4.4
			T2	3	2	11759.5	7599.1	3.1
			T3	4	1	8747.8	6434.8	4.8
			T4	5	0	7688.8	7688.8	0.0
			T5	5	0	9052.4	9052.4	0.0
		5	T1	2	2	8461.4	3153.5	4.1
			T2	3	1	11368.4	6947.4	2.2
			T3	4	0	11302.1	9627.6	∞
			T4	5	0	7015.4	7015.4	0.0
			T5	5	0	7664.6	7664.6	0.0
	3D2P	4	T1	0	0	T_{max}	T_{max}	∞
			T2	0	0	14400	T_{max}	∞
			T3	0	0	14400	T_{max}	∞
			T4	0	0	14400	T_{max}	∞
			T5	0	0	18000	T_{max}	∞
		5	T1	0	0	14400	T_{max}	∞
			T2	0	0	14400	T_{max}	∞
			T3	0	0	14400	T_{max}	∞
			T4	0	0	14400	T_{max}	∞
			T5	0	0	18000	T_{max}	∞
	3D3P	4	T1	3	0	11807.9	7679.8	∞
			T2	3	0	11681.7	7469.6	∞
			T3	4	0	6437.9	8047.4	∞
			T4	4	0	9602.3	7502.8	∞
			T5	4	0	8254.2	5817.8	∞
		5	T1	3	0	8117	7528.3	∞
			T2	3	0	11822.6	7704.3	∞
			T3	4	0	6038.6	7548.3	∞
			T4	4	0	10672	8840	∞
			T5	4	0	8874.7	6593.4	∞

Table 4.7: Computational results for instances with 20 producers

# producers	size comb.	# scen.	scen. type	# opt. sol.	# int. sol.	CPU1	CPU2	opt. gap %
20	2D2P	4	T1	0	0	T_{max}	T_{max}	∞
			T2	0	0	T_{max}	T_{max}	∞
			T3	0	0	T_{max}	T_{max}	∞
			T4	0	0	T_{max}	T_{max}	∞
			T5	0	0	T_{max}	T_{max}	∞
		5	T1	0	0	T_{max}	T_{max}	∞
			T2	0	0	T_{max}	T_{max}	∞
			T3	0	0	T_{max}	T_{max}	∞
			T4	0	0	T_{max}	T_{max}	∞
			T5	0	0	T_{max}	T_{max}	∞
	2D3P	4	T1	3	2	11877.8	7796.3	4.4
			T2	3	2	11759.5	7599.1	3.1
			T3	4	1	8747.8	6434.8	4.8
			T4	5	0	7688.8	7688.8	0.0
			T5	5	0	9052.4	9052.4	0.0
		5	T1	2	2	8461.4	3153.5	4.1
			T2	3	1	11368.4	6947.4	2.2
			T3	4	0	11302.1	9627.6	∞
			T4	5	0	7015.4	7015.4	0.0
			T5	5	0	7664.6	7664.6	0.0
	3D2P	4	T1	0	0	T_{max}	T_{max}	∞
			T2	0	0	14400	T_{max}	∞
			T3	0	0	14400	T_{max}	∞
			T4	0	0	14400	T_{max}	∞
			T5	0	0	18000	T_{max}	∞
		5	T1	0	0	14400	T_{max}	∞
			T2	0	0	14400	T_{max}	∞
			T3	0	0	14400	T_{max}	∞
			T4	0	0	14400	T_{max}	∞
			T5	0	0	18000	T_{max}	∞
	3D3P	4	T1	3	0	11807.9	7679.8	∞
			T2	3	0	11681.7	7469.6	∞
			T3	4	0	6437.9	8047.4	∞
			T4	4	0	9602.3	7502.8	∞
			T5	4	0	8254.2	5817.8	∞
		5	T1	3	0	8117	7528.3	∞
			T2	3	0	11822.6	7704.3	∞
			T3	4	0	6038.6	7548.3	∞
			T4	4	0	10672	8840	∞
			T5	4	0	8874.7	6593.4	∞

Table 4.8: Evaluating the impact of considering the nonmonotonic resource consumption

# producers	size comb.	# periods	period type	LB 1	LB 2	st.
15	2D2P	4	T1	4952.43	4952.31	×
			T2	4055.46	4055.46	
			T3	4457.55	4457.55	
			T4	4006.2	4006.2	
			T5	4058.9	4054.38	×
		5	T1	3958.3	3958.3	
			T2	3957.57	3957.57	
			T3	4420.36	4417.62	×
			T4	4100.53	4100.19	×
			T5	5061.29	5061.29	
	2D3P	4	T1	4326.77	4326.77	
			T2	4619.79	4619.58	×
			T3	4126.15	4126.15	
			T4	4625.38	4625.26	×
			T5	3978.01	3972.43	×
		5	T1	3832.21	3832.21	
			T2	4566.53	4566.53	
			T3	4095.59	4095.59	
			T4	4688.17	4688.17	
			T5	4448.74	4440.62	×
	3D2P	4	T1	4403.94	4403.94	
			T2	3944.89	3944.89	
			T3	4106.95	4106.95	
			T4	4599.3	4599.3	
			T5	4113.8	4113.73	×
		5	T1	4014.23	4014.23	
			T2	4540.44	4540.31	×
			T3	4078.01	4078.01	
			T4	4024.66	4024.57	×
			T5	4573.2	4568.03	×
	3D3P	4	T1	4558.04	4558.04	
			T2	3803.59	3803.59	
			T3	4751.48	4751.38	×
			T4	4407.22	4402.88	×
			T5	3835.37	3835.17	×
		5	T1	3745.21	3744.48	×
			T2	4345	4345	
			T3	4716.23	4716.23	
			T4	3843.6	3843.6	
			T5	4647.65	4647.49	×

Table 4.9: Comparing MPS, RP, and WCS

				Multi-period		Reference period				Worst Case		
# producers	size comb.	# periods	period type	ST	CPU	ST	gap %	recourse gap %	CPU	ST	gap %	CPU
15	2D2P	4	T1	•	T_{max}	•	0.8	55.8	T_{max}	✓	27.8	11
			T2	✓	40	✓	0	0.0	43	•	81.7	T_{max}
			T3	✓	24	✓	0.1	9.9	15	•	125.1	T_{max}
			T4	✓	97	✓	0	0.0	35	–	22.6	T_{max}
			T5	✓	213	✓	0.8	17.2	193	•	81.8	T_{max}
		5	T1	✓	164	✓	0.1	2.6	157	•	82.5	T_{max}
			T2	✓	56	✓	0	0.0	55	•	81.8	T_{max}
			T3	✓	16	✓	0	2.0	17	•	74	T_{max}
			T4	✓	73	✓	0.6	13.8	47	•	79.3	T_{max}
			T5	•	T_{max}	•	1.2	147.1	T_{max}	✓	27.3	12
	2D3P	4	T1	✓	1175	✓	0.2	134.6	923	•	64.8	T_{max}
			T2	✓	534	✓	0	0.0	541	–	–	T_{max}
			T3	✓	988	✓	0.5	508.7	1018	✓	0.1	92
			T4	✓	541	✓	0	4.0	787	✓	2.5	6
			T5	✓	1125	✓	1.2	706.6	1217	•	32.2	T_{max}
		5	T1	✓	1368	✓	0	52.2	1300	•	71	T_{max}
			T2	✓	783	✓	0	0.0	779	–	–	T_{max}
			T3	✓	975	✓	0.3	166.3	1217	•	31	T_{max}
			T4	✓	499	✓	0.7	65.5	531	•	38.1	T_{max}
			T5	✓	741	✓	0	0.0	934	•	70.7	T_{max}
	3D2P	4	T1	•	T_{max}	•	0.1	24.3	T_{max}	•	29	T_{max}
			T2	✓	1182	✓	0.2	7.1	999	–	33.6	T_{max}
			T3	✓	35	✓	0	0.0	21	–	36.3	T_{max}
			T4	•	T_{max}	•	-0.1	85.3	T_{max}	•	32.1	T_{max}
			T5	✓	191	✓	0.1	2.3	93	•	75.6	T_{max}
		5	T1	✓	68	✓	0	0.0	64	•	81.1	T_{max}
			T2	•	T_{max}	•	-0.2	6.8	T_{max}	✓	28	1
			T3	✓	28	✓	0	0.0	15	•	84.9	T_{max}
			T4	✓	229	✓	1.5	42.3	1348	–	34.4	T_{max}
			T5	•	T_{max}	•	-0.3	57.4	T_{max}	•	26.2	T_{max}
	3D3P	4	T1	✓	469	•	0	2.3	442	–	–	T_{max}
			T2	✓	1975	✓	0.7	383.3	2144	✓	2.7	7
			T3	✓	599	✓	0	0.0	944	–	–	T_{max}
			T4	✓	897	✓	0.1	246.7	767	✓	0.6	25
			T5	✓	2261	✓	0	0.0	2179	✓	0.6	48
		5	T1	✓	3029	✓	0	0.0	2780	•	32	T_{max}
			T2	✓	1002	✓	1.7	105.6	736	–	–	T_{max}
			T3	✓	615	✓	0	0.0	918	•	69.9	T_{max}
			T4	✓	1685	✓	0	102.4	2163	✓	2.7	8
			T5	✓	353	✓	0.4	131.6	463	–	–	T_{max}
Average				**40/40**	**3301.2**	**40/40**	**1.3**	**77.1**	**3347.5**	**30/40**	**46.6**	**12008**

Chapter 5

An Adaptive Large Neighborhood Search Heuristic for a Multi-Period Vehicle Routing Problem

This paper has been submitted to Journal of Heuristics.

Abstract

We consider tactical planning for a particular class of multi-period vehicle routing problems (MPVRP). This problem involves optimizing product collection and redistribution from several production locations to a set of processing plants over a planning horizon. Each horizon consists of several days, and the collection-redistribution are performed on a repeating daily basis. In this context, a single routing plan must be prepared for the whole horizon, taking into account the seasonal variations in the supply. We model the problem using a sequence of periods, each corresponding to a season, and intra-season variations are neglected. We propose an adaptive large neighborhood search with several special operators and features. To evaluate the performance of the algorithm we performed an extensive series of numerical tests. The results show the excellent performance of the algorithm in terms of solution quality and computational efficiency.

Keywords: Multi-period vehicle routing problem, Tactical planning, Seasonal variation, Adaptive large neighborhood search.

5.1 Introduction

The vehicle routing problem (VRP) is a difficult combinatorial optimization problem that is used in many practical applications relating to the design and management of distribution systems. Studies of the classical VRP and its many variants and extensions, starting with the seminal work of Dantzig and Ramser [42], represent a significant portion of the operations research literature [151]. The classical VRP, referred to as the capacitated vehicle routing problem (CVRP), concerns the determination of routes for a fleet of homogeneous vehicles, stationed at a central depot, that must serve a set of customers with known demands (supplies). The goal is to design a collection of least-cost routes such that: 1) each route, performed by a single vehicle, begins at a depot, 2) each customer is visited once by exactly one vehicle, and 3) the quantity of goods delivered (collected) on each route does not exceed the vehicle capacity [74].

In the classical VRP, the routing plan is executed repeatedly over the planning horizon. The parameters of the problem, such as the quantities to be delivered (collected) at each customer location, are assumed fixed over the horizon and known a priori. However, in many real-life applications, this assumption may result in poor-quality routing plans. Our problem setting requires routing over relatively long horizons, in environments with significant seasonal fluctuations. This setting, milk collection and redistribution in the dairy industry of Quebec, initially introduced by Dayarian et al. [45], has several problem-specific attributes and characteristics. The routing corresponds to the collection of milk from producers' farms followed by the distribution of the product to a set of processing plants. The routes must be designed in such a way that the plant demands are completely satisfied, while every producer is visited by exactly one vehicle and each vehicle delivers to just one plant per day. We assume that the daily quantity of milk produced satisfies the total plant demand.

The first studies of this problem were performed by Lahrichi et al. [101] and Dayarian et al. [44]; both studies assumed that the annual production is fixed. Dayarian et al. [45] addressed a variant of the problem that accounted for seasonal variations in the supply. Because of contractual and service-consistency requirements, a single routing plan must be prepared for a given horizon. The contractual negotiations between the different stakeholders (producers, carriers, and plants) are based on a single routing plan. For service consistency, each producer should always be included in the same route and served by the same vehicle. The drivers to plan their daily operations also use this routing plan.

Dayarian et al. [45] proposed an exact methodology based on a branch-and-price approach and a multi-period model. They divided the horizon into a series of periods, each a cluster of days with similar seasonal characteristics. The horizon can then be represented as a sequence of periods. The need to design a single plan for changing contexts recalls the *a priori* optimization framework for stochastic optimization problems. In stochastic programming, a two-stage model is often considered. The solution from the first stage is updated at the second stage as the values of the stochastic parameters are revealed.

The solution approach proposed by Dayarian et al. [45] provides optimal solutions for instances with up to twenty producers. However, real-life problems may have several hundred producers. Therefore, we need solution approaches that can find good but not necessarily optimal solutions to larger problems. The main goal of this paper is to find such solutions using an effective adaptive large neighborhood search (ALNS) framework [121, 134]. Our main contributions are as follows:

- We design a new metaheuristic based on the ALNS for our problem setting, which is described in more detail in Section 5.2.
- We design several new operators based on the special structure of the problem. We also adapt some existing operators in the literature.
- We propose a new adaptive layer for the ALNS in which destruction and construction heuristics are coupled to form the operators, rather than being treated independently.
- We propose a new diversity management system for the ALNS, which is based on extracting information from a list of diverse solutions and restarting the search from a diverse solution when it seems to be trapped in a local optimum.
- To evaluate the quality of the solution, we compute a series of lower and upper bounds on the value of the multi-period solution. We compare the solutions obtained through the ALNS with these bounds.
- We perform a series of extensive numerical tests for a large set of randomly generated instances, to illustrate the performance of the algorithm in terms of computational time and solution quality.

The remainder of this paper is organized as follows. In Section 5.2, we describe the problem and the notation that we will use. Section 5.3 discusses the state-of-the-art of work in this field. In Section 5.4, we present the classical ALNS for the VRP, and in Section 5.5 we present our approach to the problem. In Section 5.6, we propose a series of bounds that allow us to evaluate the performance of the algorithm. The experimental results are reported in Section 5.7, and Section 5.8 provides concluding remarks.

5.2 Problem statement and notation

In this section, we introduce the problem; it is inspired by a dairy problem in Quebec. For a detailed description of the dairy transportation problem in Quebec (DTPQ), the reader is referred to Lahrichi et al. [101] and Dayarian et al. [45, 44].

The problem can be formally described as follows: We wish to design a single tactical routing plan for a given horizon T. A plan consists of a set of routes, each performed by a single vehicle on every collection day of T. An unlimited fleet of identical vehicles

is assumed to be available in multiple depots. On every collection day, each vehicle departs from a depot, collects a single product type from a subset of producers, delivers the collected product to a single plant, and then returns to its depot. This can be seen as an extension of the VRP with additional deliveries to multiple plants, and it is therefore NP-hard [107].

The producers' supply over the horizon may vary seasonally. The seasonal variations are often significant and may have a major impact on the routing. We assume that a year can be divided into several periods, each representing a seasonal production level. We take inter-period production variations into account; the potential intra-period fluctuations are neglected. Intra-period fluctuations can often be handled by leaving a spare capacity of 5%–10% on each vehicle when designing the routes. The producers' seasonal fluctuations are assumed to be perfectly positively correlated. This correlation arises because almost all the producers in a given geographical region are exposed to similar seasonal cycles. The plants must adjust their seasonal demands according to the supply so that the total supply always meets the total demand.

The proposed multi-period model has some similarities to an *a priori* optimization framework in the context of the vehicle routing problem with stochastic demand (VRPSD). In a two-stage formulation of a stochastic problem, the solution from the first stage is updated at the second stage as the exact values of the stochastic parameters are revealed. We seek a solution that minimizes the total expected cost of the original plan and the potential adjustments in the second stage. Similarly to algorithms for the VRPSD, in the context of our multi-period problem at the first stage we design a single plan for the planning horizon, taking into account possible supply changes between periods. At the second stage, the plan is adjusted based on the specificities of each period. In seasons with higher supply levels, at a given producer location a vehicle may have insufficient residual capacity to collect the supply. We refer to this as a *failure*. Following a failure, the vehicle usually travels to a plant to empty its tank and then proceeds to visit the remaining producers of the planned route. We refer to this extra travel as a *recourse* action.

Under our recourse policy, the vehicle always visits the producers in the order of the planned route; when a failure occurs, it travels to its assigned plant. Consequently, the total distance traveled corresponds to the fixed length of the planned route plus the length of the return trip to the plant.

The goal is to design a single least-cost collection-delivery plan for a given horizon, providing a certain level of service consistency and service quality, and taking into account the existence of several periods. We define a feasible plan to be one that is executable over the horizon with at most one failure per operation per route.

A single plan is necessary because 1) the contractual arrangements with the carriers require a single plan that can be used for cost estimation for the whole horizon; and 2) there is a consistent driver-producer relationship when the producer is always served via the same route operated by the same vehicle. The second point leads to a familiar environment for the producer and the driver and avoids potential incompatibilities

between the vehicles and the producer's facilities.

We control the desired service quality over a given horizon by setting a *service reliability threshold* (SRT), indicating the minimum percentage of days over the horizon T that the planned routes should be executable with no failures. The magnitude of the SRT has a major impact on the design of the plan. Clearly, if SRT = 100%, no failure occurs in any period of the horizon. However, this strategy is not cost-efficient, because it often requires many vehicles.

Let Ξ be the set of all periods in a given horizon T. We associate with each period $\xi \in \Xi$ a weight W_ξ, representing the share of period ξ in horizon T. It is calculated by dividing the length of period ξ by the length of horizon T. We also associate with each period ξ a production coefficient, P_ξ, which is defined to be the ratio of the production level in period ξ to a chosen *reference production level* P_{ref}. The choice of the reference production level is discussed in detail in Dayarian et al. [45].

The model is defined on a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, where $\mathcal{V}$ and $\mathcal{A}$ are the node and arc sets, respectively. The node set contains the depots, producers, and plants; $\mathcal{V} = \mathcal{D} \cup \mathcal{N} \cup \mathcal{P}$. The arc set $\mathcal{A} \subset \mathcal{V} \times \mathcal{V}$ defines feasible movements between different locations in $\mathcal{V}$. For each pair of locations $n_i, n_j \in \mathcal{V}$, $n_i \neq n_j$, there exists an arc $(i, j) \in \mathcal{A}$. Each arc $(i, j) \in \mathcal{A}$ has an associated nonnegative travel cost that is proportional to the length of the arc $dist_{ij}$. An unlimited fleet of vehicles $\mathcal{K}$, with identical capacity Q, is available at each depot. However, employing vehicle $k \in \mathcal{K}$ incurs a fixed cost of c_k.

In each period, each producer $n_j \in \mathcal{N}$ produces a limited product quantity on a daily basis. The supply levels in period $\xi \in \Xi$ are given by a vector in which the jth parameter, denoted o_j^ξ, is the supply (offer) of producer j. Moreover, the supply of each producer n_j in the reference period is given by o_j^{ref}. Therefore, the supply of producer n_j in period ξ is

$$o_j^\xi = P_\xi . o_j^{ref} \qquad (j \in \mathcal{N}, \xi \in \Xi), \tag{5.1}$$

where P_ξ represents the production in period ξ. Each plant $p \in \mathcal{P}$ receives, on a daily basis, the collected product. The demand of each plant p in period ξ is given by D_p^ξ. The routes are designed to have no failures in the reference periods and at most one failure in the other periods. In other words, for each route r, the following inequalities hold:

$$\sum_{j \in r} o_j^\xi \leq 2Q, \qquad s \in \mathcal{S} \tag{5.2}$$

and

$$\sum_{j \in r} o_j^{ref} \leq Q. \tag{5.3}$$

The cost of the solution has three components: 1) the fixed vehicle costs; 2) the first-stage routing costs, which are the costs of the planned routes; and 3) the second-stage

routing costs, which are the expected recourse costs in different periods of the horizon.

5.3 Literature review

In this section, we review metaheuristic methods for VRPs with a similar structure to our problem.

Our problem setting has some special features:

1. The need to satisfy the plant demands; our problem can be seen as a many-to-one pickup and delivery problem (PDP).
2. The need to account for the production variations, while planning over a horizon.

Lahrichi et al. [101], investigating the same dairy application, considered a variant of the VRP with features similar to those of our problem. They used a generalized version of unified tabu search [35]. They simultaneously considered the plant deliveries, different vehicle capacities, different numbers of vehicles at each depot, and multiple depots and periods. Dayarian et al. [44] proposed a branch-and-price algorithm for a variant of the DTPQ in which a time window is associated with each producer, and the production levels over the horizon are assumed to be fixed.

The VRP that is the most similar to our problem is the multi-period or periodic MDVRP. In most studies of the multi-period vehicle routing problem (MPVRP), customers request a service that could be done over a multi-period horizon [see 153, 4, 158, 5]. The classical MPVRP is closely related to the periodic vehicle routing problem (PVRP) in which the customers specify a service frequency and allowable combinations of visit days. A complete survey of the PVRP and its extensions can be found in Francis et al. [59]. The best-known algorithms for the PVRP are those of Cordeau et al. [34], Hemmelmayr et al. [80], Vidal et al. [155], and Rahimi-Vahed et al. [126]. In our problem, all the producers need to be served every period on a daily basis. Moreover, the definition of the periods is based on the seasonal variations.

A single plan for a horizon of several periods has been investigated in the context of telecommunication network design [98, 67]. However, apart from the work of Dayarian et al. [45], we are not aware of any previous study of the VRP with the multi-period configuration considered in this paper. Dayarian et al. [45] used a branch-and-price approach to solve the problem that we investigate. However, their algorithm is able to solve instances with only up to twenty producers.

There are certain similarities between our problem and the consistent vehicle routing problem (ConVRP) introduced by Groër et al. [77]. In the ConVRP, customers with known demands receive service either once or with a predefined frequency over a multiple-day horizon. Frequent customers must receive consistent service, which is

defined as visits from the same driver at approximately the same time throughout the planning horizon [148].

Complete surveys of metaheuristics for the VRP can be found in Gendreau et al. [68] and Vidal et al. [156]. They include neighborhood searches [65, 35, 136, 24], population-based methods such as evolutionary and genetic algorithms [19, 25, 155], hybrid metaheuristics [61, 18, 85] and parallel and cooperative metaheuristics [39, 40, 100]. Of the neighborhood search methods, the large neighborhood search (LNS) algorithms [142] have proven to be successful for several classes of the VRP. The ALNS [134, 121], an extension of the LNS, is also related to the ruin-and-recreate approach of Schrimpf [138]. Recently, ALNS has provided good solutions for a wide variety of vehicle routing problems; see for instance Ropke and Pisinger [134], Pisinger and Ropke [122], Azi et al. [7], and Pepin et al. [118].

The MPVRP, as considered in this paper, has to date received limited attention. Based on the success of the ALNS, we propose an ALNS for our problem. This algorithm is outlined in the next section.

5.4 Classical ALNS for the VRP

The ALNS algorithm is an iterative process, in which at every iteration part of the current solution is destroyed using a destruction heuristic and then reconstructed using a construction heuristic in the hope of finding a better solution. The destruction heuristic usually disconnects a number $q \in [q_{min}, q_{max}]$ of the nodes from their current routes and places them into the *unassigned node pool*, Φ. Note that q_{min} and q_{max} are parameters whose values are to be tuned. The construction heuristic then inserts the nodes from Φ into the routes of the solution. The main components of the ALNS [134, 121] are:

Adaptive search engine: At each iteration of the ALNS, one independently selects a destruction and a construction heuristic via a biased random mechanism, referred to as the roulette-wheel. It favors the heuristics that have been successful according to certain criteria in recent iterations. The adaptive layer of the ALNS procedure controls the functionality of the roulette-wheel. One associates with each heuristic a weight that is incremented during the search based on a scoring mechanism that measures the performance of the heuristic. The probability of selecting a given heuristic is proportional to the ratio of its weight to the sum of the weights of the other heuristics.

Adaptive weight adjustment: One divides the search into a number of fixed-length segments of consecutive iterations. In the first segment, all the heuristics have the same weight. At the end of each segment, the weights used to select the heuristics are updated.

Acceptance and stopping criteria: The new solution obtained via the destruction-

construction procedure is usually accepted or rejected based on some criterion. This criterion is usually defined by the search paradigm applied at the master level, e.g., simulated annealing (SA) [see 93]. The new solution s' replaces the current solution s if $f(s') < f(s)$, where $f(s)$ represents the value of solution s. In SA, with $\Delta f = f(s') - f(s)$, solution s' is accepted with probability

$$\exp(\frac{-\Delta f}{T}), \tag{5.4}$$

where $T > 0$ is the temperature parameter. The temperature is initialized to T^{init} and at the end of each iteration it is lowered by a cooling rate $c \in (0, 1)$: $T \leftarrow c.T$. The probability of accepting worse solutions reduces as T decreases. This allows the algorithm to progressively find better local optima. The stopping criterion is based either on a predetermined number of iterations or a predefined final temperature T^{fin}.

5.4.1 Selected destruction/construction heuristics

Several destruction and construction heuristics have been proposed, and some can be adapted to the VRP context. We focus on the destruction heuristics outlined below.

Random Removal: This heuristic [134] randomly selects q nodes, removes them from their current position, and places them into Φ. The random nature of this heuristic diversifies the search.

Worst Removal: This heuristic, initially proposed by Rousseau et al. [136] and later used by Ropke and Pisinger [134], removes the q worst placed nodes and places them into Φ.

Route Removal: This heuristic removes a randomly selected route and places the corresponding nodes in Φ.

Cluster Removal: This heuristic [121] removes a cluster of nodes from a route, based on their geographical region. It randomly selects a route from the current solution. It then applies the well-known Kruskal algorithm to find a minimum spanning tree for the nodes of this route, based on the arc length. When two forests have been generated, one of them is randomly chosen and its nodes are removed and placed in Φ.

Smart Removal: This heuristic [136] randomly selects a pivot node and removes portions of different routes around the pivot, based on a reference distance and a proximity measure.

We consider the following construction heuristics:

Sequential Insertion: This heuristic inserts the nodes from Φ in order. Each node is placed in the position that incurs the minimal local cost.

Best-First Insertion: This heuristic inserts each node in the cheapest position. At each step it selects the node with the lowest insertion cost.

Regret Insertion: This heuristic [134], orders the nodes in Φ by decreasing regret values. The regret value is the cost difference between the best insertion position and the second best. More generally, the k-regret heuristic defines the regret value with respect to the k best routes.

5.5 Proposed solution framework

Our algorithm is based on a general ALNS but incorporates a number of intensification and diversification strategies that improve its performance; an outline is presented in Algorithm 3.

Similarly to the classical ALNS, the search is divided into several segments, each including a series of consecutive iterations. However, we dynamically adjust the length of each segment based on a criterion that will be described in Section 5.5.4. At each iteration, we explore the neighborhood of the current solution, generating potentially φ new solutions. We obtain the new solutions by applying a randomly selected destruction-construction operator to the current solution. At the end of each iteration, we apply an acceptance criterion to the best solution among the φ solutions found. If the solution satisfies the criterion, it replaces the current solution. Our acceptance criterion, which is based on a probabilistic threshold inspired by SA, is discussed in Section 5.5.4.

At the end of each segment, we apply a series of local search (LS) operators to the best solution found in the segment. If this gives an improvement, we update the current solution. To help the algorithm escape from local minima, we implement a diversity management mechanism; see Section 5.5.6.

We also propose the use of an enhanced central memory. It stores both high-quality solutions and a set of diverse solutions. We design several new destruction heuristics that use information extracted from the central memory; see Section 5.5.7. Moreover, we design new operators for our specific problem setting. The main components of our algorithm are described below.

5.5.1 Search space

It is well known in the metaheuristic literature that allowing the search into infeasible regions may lead to good solutions. We therefore permit infeasible solutions in which the plant demands are not completely satisfied. We evaluate the moves and solutions using a penalty function $f(s) = C(s)+\eta D^{-}(s)$, where $C(s)$ is the total operating cost of the solution (i.e., fixed, routing, and recourse costs) and $D^{-}(s)$ is the unsatisfied plant demand. The parameter η is initially set to 1. After each block of $Iter^{adj}$ iterations, we multiply η by 2 if the number of infeasible solutions in the last $Iter^{his}$ iterations is

greater than δ_{max}, and we divide it by 2 if the number of such solutions is less than δ_{min}.

This penalty function is similar to that used in Taburoute [65] and the unified tabu search [35]. Our penalty strategy favors removal from routes serving plants with an oversupply and insertion into routes serving plants with an undersupply.

Our penalty function adds a penalty ρ to the local cost of removal or insertion in a given position, where

$$\rho = \eta D^{-}(s). \tag{5.5}$$

5.5.2 Destruction-construction operators

The new solutions are obtained by applying an operator $opr \in \Omega$, to the current solution, where Ω is the set of all operators. In the classical ALNS algorithm, the destruction and construction heuristics are selected independently, but we form *opr* by coupling a pair of heuristics. The main advantage is that we can weight the performance of each (destruction-construction) pair. We use two main types of heuristics:

1. Generic heuristics: A generic destruction heuristic can be coupled with any generic construction heuristic. The heuristics presented in Sections 5.4, 5.5.7, and 5.5.8 are all considered generic heuristics.
2. Specialized heuristics: For each specialized destruction heuristic, we develop a specialized construction heuristic. The resulting operator has a specific goal such as the generation of solutions with a certain level of diversity. We present our specialized heuristics in Section 5.5.9.

5.5.3 Central memory

We consider a central memory, denoted Ψ, with a limited number of solutions. We extract from it different types of information for use in destruction heuristics. There is clearly a trade-off between search quality on the one hand and computational efficiency and memory requirements on the other. We use the extracted information to determine the relatedness between different nodes of the graph with respect to different criteria. We design a destruction heuristic based on each criterion (see Section 5.5.7). The central memory contains three lists of solutions:

- **Best Feasible Solutions (Ψ_{FS}):** A list of the β_1 best feasible solutions generated so far.
- **Best Infeasible Solutions (Ψ_{NFS}):** A list of the β_2 best infeasible solutions generated so far.

- Diverse Solutions (Ψ_{DIV}): A list of β_3 solutions, selected according to the quality-diversity criterion discussed in Section 5.5.6.

5.5.4 Acceptance criterion

Our acceptance criterion is inspired by SA, but we do not perform the cooling procedure at the end of every iteration. We perform the procedure when no global best feasible solution has been found in the last δ iterations. This can be seen as a *dynamic repetition schedule* that dynamically defines the number of iterations executed at a given temperature.

We divide the search into several segments. The length of each segment corresponds to the repetition schedule for a given temperature and therefore has a minimum length of δ iterations. If a new global best feasible solution is found in the current segment, the length of the segment is extended.

5.5.5 Adaptive mechanism

At every iteration of the ALNS, we choose the operator to apply to the current solution via a roulette-wheel mechanism. Each operator *opr* is assigned a weight ω_{opr} according to its performance history. Given Ω, the operator set, the probability of selecting *opr* is $\omega_{opr} / \sum_{k \in \Omega} \omega_k$.

To evaluate the performance of the operators, we implement an adaptive weight adjustment procedure. After each block of γ segments, referred to as a mega-segment, we update the operator weights based on their long- and short-term performance history. The short-term history covers the last mega-segment, and the long-term history covers the entire search. Each operator is also assigned a score, which is reset to zero at the end of each mega-segment. Initially, all the weights are set to one and all the scores to zero. At each iteration, we update the scores. We do this by adding a bonus factor $\sigma_i, i \in \{1, \ldots, 4\}$, where $\sigma_i \leq \sigma_{i+1}, i \in \{1, 2, 3\}$, to the current score as follows:

I. We add σ_4 if a new global best feasible solution has been found.

II. We add σ_3 if the new solution improves the current solution but not the global best feasible solution.

III. We add σ_2 if the new solution satisfies the acceptance criterion and is inserted into Ψ_{FS}.

IV. We add σ_1 if the new solution satisifes the acceptance criterion but is not inserted into Ψ_{FS}.

In all other cases, the bonus factor is zero. Moreover, suppose that π_{opr} is the total score of *opr* obtained from ν_{opr} applications of *opr* in the last mega-segment. We

control the influence of the short- and long-term history using a parameter $\alpha \in [0, 1]$, called the reaction factor, through the formula

$$\omega_{opr,\iota+1} = \omega_{opr,\iota}(1 - \alpha) + \alpha \frac{\pi_{opr}}{\nu_{opr}}. \tag{5.6}$$

Here $\omega_{opr,\iota}$ represents the weight of operator *opr* in mega-segment ι. A value of α close to zero increases the impact of the long-term history; a value close to one increases the impact of the short-term history.

5.5.6 Diversity management

The diversity of the search is governed by the two mechanisms discussed below.

Diverse Solutions (Ψ_{DIV})

The decisions taken in some destruction heuristics use the information extracted from the central memory Ψ. Recall that this central memory is divided into lists of best feasible solutions, best infeasible solutions, and diverse solutions. Several strategies have been proposed for determining a set of diverse solutions [155, 126].

We consider a new utility function that evaluates the solution based on $g(s) = C(s) - \upsilon DIV(s)$, where $C(s)$ is the quality and $DIV(s)$ is the diversity measure:

$$DIV(s) = \sum_{s' \in \Psi_{FS} \cup \Psi_{NFS}} \chi(s, s'). \tag{5.7}$$

Parameter υ is self-adjusting: if during the last $nbSegm^{DIV}$ segments no improved solution has been found, υ is doubled.

In Equation (5.7), $\chi(s, s')$ is the distance between solutions s and s':

$$\chi(s, s') = \sum_{i \in \mathcal{N}} (\mathbf{1}(succ^{s}_{n_i} \neq succ^{s'}_{n_i}) + \mathbf{2}(n^{s}_{d_i} \neq n^{s'}_{d_i}) + \mathbf{2}(n^{s}_{p_i} \neq n^{s'}_{p_i})). \tag{5.8}$$

Here $succ^{s}_{n_i}$, $n^{s}_{d_i}$, and $n^{s}_{p_i}$ are the successor, depot, and plant of node n_i in solution s.

We compare two solutions based on their node sequencing and their assignments to depots and plants. The weight of each sequencing difference is set to one, and the weight of each assignment difference is set to two. We first attempt to insert every new solution into Ψ_{FS} or Ψ_{NFS}. When a solution cannot be inserted or a solution currently in one of these lists is replaced, we must decide whether or not to add it to Ψ_{DIV}. If the number of solutions in Ψ_{DIV} is less than β_3, we add the new solution. Otherwise, we consider the current members of Ψ_{DIV} and the incoming solution, and we retain the β_3 best solutions as measured by $g(s)$ values.

Diversity segment

If after $nbSegm^{DIV}$ segments, no new best global solution is found, we devote a complete segment to the generation of diverse solutions. In this *diversity segment*, at each iteration, we randomly select one operator from the *specialized operators* (Section 5.5.9). Similarly to the regular segments, at each iteration, we generate φ solutions. We take the most diverse one according to $DIV(s)$ that is within a radius of r of the best-known solution and apply an acceptance criterion. In this criterion, the new solution s' replaces the current solution s if $g(s') < g(s)$.

5.5.7 Generic destruction heuristics

We now describe the generic destruction heuristics. Some are new, while others are adapted from existing heuristics proposed by Pisinger and Ropke [121], which primarily differ in the way that the relatedness are weighted. We use the six heuristics below.

Solution-Cost-Based Related Removal

The solution-cost-based related removal heuristic, based on the historical node-pair removal [121], associates with each arc $(u, v) \in A$ a weight $f^*(u, v)$. This weight indicates the value of the best-known solution that contains arc (u, v). Whenever a new solution is inserted in the central memory, we update the $f^*(u, v)$ value of all the arcs (u, v) in the solution.

Following a call to this heuristic, we perform a worst removal procedure in which the weight $f^*(u, v)$ replaces the cost of each arc $(u, v) \in A$. We repeat this process until q nodes have been removed and placed in Φ.

Route-Cost-Based Related Removal

The route-cost-based related removal heuristic, which is similar to the heuristic above, associates with each arc $(u, v) \in A$ a weight $r^*(u, v)$, indicating the value of the minimal-cost route found so far that contains arc (u, v). We perform a worst removal based on the $r^*(u, v)$ weights.

Paired-Related Removal

This heuristic investigates adjacent producer nodes. We give each arc (i, j) a weight $\varpi_{(i,j)}$, initially set to 0. The heuristic starts by adding a weight h_s to the weights of all the arcs used in the solutions of the central memory. When an arc (i, j) is used by solution s, we add the weight h_s to both (i, j) and (j, i). We compute h_s

Algorithm 3 ALNS

```
1: s ← InitialSolution;
2: Initialize the weights π;
3: Set the temperature T;
4: iter ← 0;
5: segmentIter ← 0;
6: seg ← 0;
7: s_seg ← s;
8: repeat
9:     repeat
10:        s_iter ← s;
11:        q_iter ← Number of nodes to be removed;
12:        Opr_iter ← Select an operator;
13:        s' ← Opr_iter(s, q_iter);
14:        if f(s') < f(s_iter) then
15:            s_iter ← s';
16:        end if
17:    until iter/φ == 0
18:    if f(s_iter) < f(s*) and s_iter feasible then
19:        s* ← s_iter;
20:        s_seg ← s_iter;
21:        segmentIter ← 0;
22:    else
23:        if ACCEPT(s_iter, s) then
24:            s ← s_iter;
25:        end if
26:    end if
27:    if f(s_iter) < f(s_seg) then
28:        s_seg ← s_iter;
29:    end if
30:    Update the score of opr;
31:    if segmentIter == δ then
32:        s' ← LOCAL SEARCH(s_seg);
33:        if f(s') < f(s*) then
34:            s* ← s';
35:            segmentIter ← 0;
36:        end if
37:    else
38:        if f(s) < f(s) then
39:            s ← s';
40:        end if
41:        T ← c.T;
42:        s_seg ← s;
43:        seg ← seg + 1;
44:    end if
45:    if seg/γ == 0 then
46:        Update the weights;
47:    end if
48:    iter ← iter + 1;
49:    segmentIter ← segmentIter + 1;
50: until Stopping Criterion
51: return s*
```

via $h_s = List.size() - pos_{inList}(s)$, where $List$ represents the list to which solution s belongs, $List.size()$ is the length of that list, and $pos_{inList}(s)$ is the position of solution s in that list. This procedure favors the solutions at the start of the lists. When a new solution is inserted into any of the lists, we update the weights h_s. We use the arc weights $\varpi_{(i,j)}$ to identify the q producer nodes that seem to be related to each other. An initial node n_i is randomly selected, removed, and placed in Φ. Then, while $|\Phi| < q$, we randomly select a node n_j from Φ and identify the node n_k in Φ that is the most closely related to node n_j (it has the highest $\varpi_{(j,k)}$). We then remove the node n_k and place it in Φ.

Route-Related Removal

This heuristic, similarly to the previous heuristic, adds a weight h_s to all pairs of nodes served by the same route in solution s. We assign weights as for the previous heuristic. We remove nodes from their current position following a similar procedure to that for the previous heuristic.

Depot-Producer-Related Removal

This heuristic attempts to identify the nodes that may be mis-assigned to a depot. Each depot-node pair (n_d, n_i), for $d \in \mathcal{D}$ and $i \in \mathcal{N}$, is assigned a weight. The weight increases by h_s if, in solution s, producer i is assigned to a route departing from depot d. We calculate the value of h_s as for the paired-related removal heuristic. We select a node to remove via the following steps:

Step 1: We sort the producer-depot assignments in the current solution s according to the historical pair weights obtained as described above in $List_{i,d}(s)$.

Step 2: Starting from the producer-depot pair with the lowest weight, we remove nodes from their current position with probability

$$Pr_{n_i,n_{d_i}}(s) = \frac{rank(n_i)}{List_{i,d}(s).size()} \tag{5.9}$$

where $rank(n_i)$ is the position of the pair (n_{d_i}, n_i) in $List_{i,d}(s)$. Moreover, $List_{i,d}(s).size()$ is the length of the node-depot list, which is the number of producer nodes. Accordingly, we remove the node with the lowest weight from its current position with probability 1.

Step 3: If the list is traversed to the end, but the number of removed nodes is less than q, we update the length of the list to $List_{i,d}(s).size() - |\Phi|$ and make the corresponding updates to the pair ranking. We then return to **Step 2.**

Plant-Producer-Related Removal

This heuristic follows the three steps above. It attempts to remove producer nodes based on the node-plant pair weights calculated from the historical information.

5.5.8 Generic construction heuristics

After the destruction heuristic, the nodes that have been removed and placed in Φ are considered for reinsertion into routes.

Sequential Insertion with Plant Satisfaction

This heuristic is identical to sequential insertion except that insertions are not penalized with the parameter ρ. To overcome possible infeasibilities, we use the following two-phase insertion heuristic. In the first phase we insert nodes only for routes serving plants with an unsatisfied demand. When all the plant demands are met we move to the second phase. The remaining nodes may now be inserted in any route, as in normal sequential insertion. This procedure does not necessarily guarantee the feasibility of the new solution, because it depends on the infeasibility level of the original solution and the nodes in Φ.

Minimum-Loss Insertion

This heuristic is based on the regret insertion heuristic but does not use ρ. It inserts nodes into the routes while attempting to maintain the feasibility of the solution at the minimal cost. This heuristic is based on the regret associated with the insertion of a node in a route serving a plant with unsatisfied demand rather than in the best possible route. Clearly, the best candidate is a node for which the best possible position is in a route serving a plant with unsatisfied demand. The best insertion candidate is determined using the following criterion:

$$n_i := arg \min_{n_i \in \Phi} (\min_{r \in R_s^{DP}} (\Delta f_{r+n_i}(s)) - \min_{r \in R_s} (\Delta f_{r+n_i}(s))), \tag{5.10}$$

where R_s is the set of routes for solution s, and R_s^{DP} is the set of routes serving plants with unsatisfied demand. If all the plant demands are met, the insertion order of the remaining nodes in Φ is defined as for the regret insertion operator.

5.5.9 Specialized operators

We also design specialized operators for our problem setting. Each consists of a pair of destruction and construction heuristics that work together; they are not coupled with

any other heuristics. The destruction and construction heuristics cooperate to achieve diverse solutions (recall that these operators are only used in the diversity segments). We use the three operators below.

Depot-Exchange Operator

This operator changes the depot of a subset of the nodes to enhance the diversification.

Depot-Exchange Removal

This heuristic selects the nodes to be removed via the following steps:

Step 1: Randomly select a pair of depots, n_{d_1} and n_{d_2}.

Step 2: Sort the nodes n_i, $i \in \mathcal{N}$ in $List_{reg}(n_{d_1}, n_{d_2})$ according to the regret of assigning them to n_{d_1} or n_{d_2} using the following formula:

$$regret(n_i)_{d_1,d_2} = |dist_{d_1,n_i} - distd_2, n_i|. \tag{5.11}$$

Step 3: Starting from the node with the lowest regret value, remove nodes from their current positions with probability

$$Pr_{n_i} = \frac{rank(n_i)}{List_{reg}(n_{d_1}, n_{d_2}).size()}, \tag{5.12}$$

where $rank(n_i)$ is the position of node n_i in $List_{reg}(n_{d_1}, n_{d_2})$, so the position of the node with the smallest regret value is $List_{reg}(n_{d_1}, n_{d_2}).size()$.

Step 4: If the $List_{reg}(n_{d_1}, n_{d_2})$ is completely traversed but still $|\Phi| < q$, replace $List_{reg}(n_{d_1}, n_{d_2}).size()$ by $List_{reg}(n_{d_1}, n_{d_2}).size() - |\Phi|$, update $rank(n_i)$, and go to **Step 3**.

Depot-Exchange Insertion

Following a call to the above heuristic, we reinsert nodes from Φ into routes using this heuristic. It is based on the regret insertion heuristic, while nodes are preassigned to the depots. This preassignment uses the depots n_{d_1} and n_{d_2} selected for the removal heuristic. Each node $n_i \in \Phi$ is assigned to n_{d_1} with probability

$$Pr_{n_i,d_1} = 1 - \frac{dist_{d_1,n_i}}{dist_{d_1,n_i} + dist_{d_2,n_i}}, \tag{5.13}$$

and to n_{d_2} with probability $1 - Pr_{n_i,d_1}$.

Plant-Exchange Operator

Similarly to the depot-exchange operator, this operator changes a subset of the producer-plant assignments.

Plant-Exchange Removal

We randomly select two plants, n_{p_1} and n_{p_2}. We then sort the nodes based on the regret value $regret(n_i)_{p_1,p_2} = |dist_{p_1,n_i} - dist_{p_2,n_i}|$. Starting from the node with the lowest regret value, we remove nodes from their current positions with probability

$$Pr_{n_i} = \frac{rank(n_i)}{List_{reg}(n_{p_1}, n_{p_2}).size()}. \tag{5.14}$$

Plant-Exchange Insertion

This is a regret insertion heuristic that restricts the insertion of each node $n_i \in \Phi$ to a preassigned producer-plant pair. The preassignments use the plants n_{p_1} and n_{p_2} selected for the removal heuristic. The probability function is

$$Pr_{n_i,p_j} = 1 - \frac{dist_{p_j,n_i}}{dist_{p_1,n_i} + dist_{p_2,n_i}}, \qquad j \in \{1, 2\}. \tag{5.15}$$

Tabu-Based Operator

This operator pairs the most random removal and one of the most successful insertion heuristics, i.e., the random removal heuristic and the regret insertion heuristic. To diversify the search, it attempts to avoid generating the same solutions by prohibiting the reassignment of removed nodes to their previous routes. The removal heuristic records a list of the routes to which the removed nodes were previously assigned. The regret insertion heuristic is as described in Section 5.4 except that it avoids reinserting a node into its previous route. The removal heuristic has a high level of diversity, and the insertion heuristic is designed to insert removed nodes into the best routes (provided they are different from the previous routes).

5.5.10 Local search

At the end of each segment, LS procedures are performed on the best solution found during the segment. Our LS procedures are inspired by the education phase of a genetic algorithm proposed by Vidal et al. [155]. The procedures are restricted to the feasible region. We build each node's neighborhood using a granularity threshold ϑ, initially

proposed by Toth and Vigo [152], which is computed as follows:

$$\vartheta = \lambda \frac{Z(s)}{nbArc(s)}, \tag{5.16}$$

where $Z(s)$ and $nbArc(s)$ are the sum of the arc costs and the number of arcs used in solution s, and λ is a suitable sparsification factor. In our implementation, $Z(s)$ and $nbArc(s)$ are limited to the arcs between producer nodes; the recourse costs and the corresponding arcs are omitted. The value $\frac{Z(s)}{nbArc(s)}$ is the average length of the arcs between the producer nodes in solution s. The neighbour set of each node n_i contains all nodes n_j such that $dist_{ij} \leq \vartheta$.

Suppose that n_u, assigned to route r_u, is a neighbor of n_v, assigned to route r_v. Moreover, suppose that n_x and n_y are immediate successors of n_u and n_v in r_u and r_v, respectively. For every node n_u and all of its neighbors n_v, we perform the LS operators in a random order. When a better solution is found, the new solution replaces the current solution. The LS stops when no operator generates an improved solution. The LS operators are as follows:

Insertion 1: Remove n_u and reinsert it as the successor of n_v.

Insertion 2: Remove n_u and n_x; reinsert n_u after n_v and n_x after n_u.

Insertion 3: Remove n_u and n_x; reinsert n_x after n_v and n_u after n_x.

Swap 1: Swap the positions of n_u and n_v.

Swap 2: Swap the position of the pair (n_u, n_x) with n_v.

Swap 3: Swap the position of (n_u, n_x) with (n_v, n_y).

2-opt: If $r_u = r_v$, replace (n_u, n_x) and (n_v, n_y) with (n_u, n_v) and (n_x, n_y).

2-opt* 1: If $r_u \neq r_v$, replace (n_u, n_x) and (n_v, n_y) with (n_u, n_v) and (n_x, n_y).

2-opt* 2: If $r_u \neq r_v$, replace (n_u, n_x) and (n_v, n_y) with (n_u, n_y) and (n_x, n_v).

5.6 Bounds on the multi-period solution

To evaluate the performance of our algorithm, we compute lower and upper bounds on the objective function value. This calculation is based on the set partitioning formulation of the problem [45]. Let the single-period problem that considers only the production levels in the reference period be P^{ref}, with optimal solution x^{ref}. Let P^{mp} be the multi-period problem, with optimal solution x^*.

The route cost has three components: 1) fixed vehicle costs, 2) first-stage routing costs, and 3) second-stage routing costs (recourse costs). These components are denoted $c_f(x)$, $c(x)$, and $\mathcal{F}(x)$, respectively.

Table 5.1: Bounding multi-period solution notation

Notation	Description
P^{mp}	the multi-period problem.
P^{ref}	the single-period problem corresponding to the reference period.
x^*	optimal solution of the multi-period problem.
x^{ref}	optimal solution of P^{ref}.
$c_f(x)$	fixed vehicle costs of solution x.
$c(x)$	first-stage routing costs of solution x.
$\mathcal{F}(x)$	recourse costs of solution x.
p_r	the plant to which route r is assigned.
t_r^ξ	is 1 if a failure occurs on route r in period ξ.

For any feasible solution x of P^{mp}, the following inequality provides an upper bound on the value of the multi-period solution:

$$c_f(x^*) + c(x^*) + \mathcal{F}(x^*) \leq c_f(x) + c(x) + \mathcal{F}(x). \tag{5.17}$$

Moreover, since the fixed vehicle costs are significantly large compared to the total routing costs, the number of vehicles used in the multi-period solution is the minimum number of vehicles needed during the reference period, so the fixed vehicle costs are the same:

$$c_f(x^*) = c_f(x^{ref}). \tag{5.18}$$

Since x^* is also a feasible solution to P^{ref}, we have

$$c(x^{ref}) \leq c(x^*). \tag{5.19}$$

We combine (5.18) and (5.19) to obtain a lower bound on the value of the multi-period solution:

$$c_f(x^{ref}) + c(x^{ref}) \leq c_f(x^*) + c(x^*) + \mathcal{F}(x^*). \tag{5.20}$$

We also consider a lower bound on the value of $\mathcal{F}(x^*)$. Let $F(r, \xi)$ be the recourse cost in period $\xi \in \Xi$ for route $r \in \mathcal{R}_s$, where $\mathcal{R}_s$ is the set of routes in solution s. We have

$$\mathcal{F}(x) = \sum_{\xi \in \Xi} \sum_{r \in \mathcal{R}_s} F(r, \xi). \tag{5.21}$$

Let the set of producer nodes visited by route r be $\mathcal{N}_r$, the plant to which r is assigned be p_r, and the set of all routes serving plant $p \in \mathcal{P}$ be $\mathcal{R}_s^p \subseteq \mathcal{R}_s$. Then

$$F(r, \xi) \geq \quad 2 \min_{i \in \mathcal{N}_r} dist_{i,p_r}.t_r^\xi \tag{5.22}$$

$$\Rightarrow \mathcal{F}(x^*) \geq \quad 2 \sum_{r \in \mathcal{R}_s} t_r^\xi \min_{i \in \mathcal{N}_r} dist_{i,p_r} \tag{5.23}$$

$$= \quad 2 \sum_{p \in \mathcal{P}} \sum_{r \in \mathcal{R}_s^p} t_r^\xi \min_{i \in \mathcal{N}_r} dist_{i,p_r}, \tag{5.24}$$

where $F(r, \xi)$ is the recourse cost on route r in period ξ, and t_r^{ξ} is a binary parameter, which is equal to 1 if a failure occurs on route r in period ξ and 0, otherwise.

The minimal failure cost for a given instance can then be computed by first determining the minimum number of vehicles needed to serve the plants and producers. We then assign the producers to vehicles (routes) while attempting to minimize the total failure cost. To do this, we assign failure points to the routes so that the total failure cost is minimized. We perform this two-step procedure by solving the bin-packing models discussed below.

5.6.1 Minimum number of vehicles

We first present the model that allows us to determine the minimum number of vehicles to cover the plant demands. Table 5.2 gives the notation, and the constraints are as follows:

1. Each producer is assigned to one vehicle and each vehicle to one plant;
2. The vehicle capacities are respected;
3. The plant demands are satisfied.

Table 5.2: Bin-packing notation for minimum number of vehicles

Notation	Description
x_{ikp}	1 if producer i is assigned to vehicle k and plant p.
y_{kp}	1 if vehicle k serves plant p.
o_i	supply of producer $i \in \mathcal{N}$.
D_p	demand of plant $p \in \mathcal{P}$.

The formulation is

$$\min \quad \sum_{k \in \mathcal{K}} \sum_{p \in \mathcal{P}} y_{kp} \tag{5.25}$$

subject to

$$\sum_{k \in \mathcal{K}} \sum_{p \in \mathcal{P}} x_{ikp} = 1 \quad (i \in \mathcal{N}); \tag{5.26}$$

$$\sum_{i \in \mathcal{N}} \sum_{k \in \mathcal{K}} o_i x_{ikp} \geq D_p \quad (p \in \mathcal{P}); \tag{5.27}$$

$$\sum_{i \in \mathcal{N}} o_i x_{ikp} \leq Q \quad (p \in \mathcal{P}, k \in \mathcal{K}); \tag{5.28}$$

$$x_{ikp} \leq y_{kp} \quad (i \in \mathcal{N}, p \in \mathcal{P}, k \in \mathcal{K}); \tag{5.29}$$

$$x_{ikp}, y_{kp} \in 0, 1 \quad (i \in \mathcal{N}, p \in \mathcal{P}, k \in \mathcal{K}), \tag{5.30}$$

where the objective function minimizes the number of vehicles. Constraint (5.26) ensures that all the producers are assigned to exactly one route. Constraint (5.27) ensures

that the plant demands are satisfied, and Constraint (5.28) ensures that the vehicle capacities are respected.

5.6.2 Minimum failure cost

Given the minimum number of vehicles, we can compute a lower bound on the total failure cost of P^{mp} based on inequality (5.24). Let the minimum number of vehicles be K^*. We assign nodes to the restricted vehicle set $\mathcal{K}^*$, assuming that for a given route r, all the failures in different periods occur on the node that is closest to p_r. We assign the nodes by solving an extension of the first bin-packing formulation that minimizes the failure cost.

Table 5.3: Bin-packing notation for minimum failure cost

Notation	Description
$\mathcal{K}^*$	set of K^* identical vehicles.
t_k^ξ	1 if a failure in period ξ is assigned to vehicle k.
u_{ikp}^ξ	1 if a failure in period ξ is assigned to producer i on vehicle k, serving plant p.
l_{kp}	quantity delivered to plant p by vehicle k.

Table 5.3 gives the notation, and the formulation is

$$Z = \min \quad \sum_{\xi \in \mathcal{S}} W_\xi \sum_{p \in \mathcal{P}} \sum_{i \in \mathcal{N}} 2.d_{i,p} u_{ikp}^\xi \tag{5.31}$$

subject to

$$l_{kp} = \sum_{i \in \mathcal{N}} o_i x_{ikp} \quad (p \in \mathcal{P}, k \in \mathcal{K}^*); \tag{5.32}$$

$$l_{kp} \leq Q y_{kp} \quad (p \in \mathcal{P}, k \in \mathcal{K}^*); \tag{5.33}$$

$$\sum_{p \in \mathcal{P}} y_{kp} = 1 \quad (k \in \mathcal{K}^*); \tag{5.34}$$

$$\sum_{k \in \mathcal{K}^*} l_{kp} \geq D_p \quad (p \in \mathcal{P}); \tag{5.35}$$

$$\sum_{k \in \mathcal{K}^*} \sum_{p \in \mathcal{P}} x_{ikp} = 1 \quad (i \in \mathcal{N}); \tag{5.36}$$

$$x_{ikp} \leq y_{kp} \quad (i \in \mathcal{N}, p \in \mathcal{P}, k \in \mathcal{K}^*); \tag{5.37}$$

$$Pt(\xi) \sum_{p \in \mathcal{P}} l_{kp} \leq Q(1 + t_k^\xi) \quad (\xi \in \mathcal{S}, k \in \mathcal{K}^*); \tag{5.38}$$

$$\sum_{p \in \mathcal{P}} \sum_{i \in \mathcal{N}} u_{ikp}^\xi = t_k^\xi \quad (\xi \in \mathcal{S}, k \in \mathcal{K}^*); \tag{5.39}$$

$$u_{ikp}^\xi \leq x_{ikp} \quad (\xi \in \mathcal{S}, i \in \mathcal{N}, p \in \mathcal{P}, k \in \mathcal{K}^*); \tag{5.40}$$

$$y_{kp} \leq y_{k-1p} + y_{k-1p-1} \quad (p \in \mathcal{P}, k \in \mathcal{K}^*); \tag{5.41}$$

$$y_{11} = 1; \tag{5.42}$$

$$x_{ikp}, y_{kp}, t_k^\xi, u_{ikp}^\xi \in \{0, 1\} \quad (\xi \in \mathcal{S}, i \in \mathcal{N}, p \in \mathcal{P}, k \in \mathcal{K}^*). \tag{5.43}$$

Constraints (5.32) and (5.33) ensure that the vehicle capacities are satisfied. Constraint (5.34) ensures that each vehicle is assigned to a single plant. Constraint (5.35) ensures that the plant demands are satisfied, and Constraint (5.36) ensures that each producer is assigned to a single vehicle. Constraint (5.37) ensures that producers are assigned only to open routes. For each period ξ, Constraints (5.38)–(5.40) determine the number and location of failures on each vehicle k. Constraints (5.41) and (5.42) break the possible symmetry due to the set of identical vehicles. The objective function, Z, represents a lower bound on the total failure cost. We assume that, for a given route, all the failures in different periods occur in the node that is closest to the assigned plant. The bound can be tightened if we acknowledge that not all periods have failures at the same node. Proposition 1 provides a condition determining when two periods both encounter failure at the same node.

Proposition 1. *Two periods ξ_1 and ξ_2 both encounter a failure at node n_j if the following inequality holds:*

$$\frac{Q}{P_2}(1 - \frac{P_2}{P_1}) \leq o_j. \tag{5.44}$$

Proof. Assume that $P_1 \geq P_2$ and that in period ξ_1 the quantity collected prior to node n_j is Q. The quantity collected in period ξ_2 will then be $P_2.\frac{Q}{P_1}$. Moreover, ξ_2 has a failure at node n_j if $P_2.\frac{Q}{P_1} + P_2.o_j \geq Q$. □

Including this condition in the model (5.31)–(5.43) may lead to an increase in the value of Z by assigning certain failure points to nodes that are farther from the plant. This occurs when two different periods cannot both encounter failure on the closest node to the plant.

5.7 Computational experiments

We now describe our computational experiments. In Section 5.7.1, we introduce the set of test problems. We calibrate the parameter values via extensive sensitivity analysis; the results of these tests are presented in Section 5.7.2. We also study the impact of different components of the algorithm based on a series of tests, which are presented in Section 5.7.3. Finally, the computational results for the test problems are presented in Section 5.7.4.

5.7.1 Test problems

We consider the test problems proposed by Dayarian et al. [45] as well as extensions of them. The extensions increase the size of the instances. Dayarian et al. [45] generated instances with 15 or 20 producers, 2 or 3 depots, and 2 or 3 plants. Each instance was solved with 4 or 5 periods, to represent the multi-periodic aspect of the problem. For

each case with 4 or 5 periods, 5 different scenarios $\{T1, \ldots, T5\}$ were explored, differing in terms of the distribution of the period weights and the SRT level. The details of the instances considered in this paper are presented in Table 5.4.

Table 5.4: Specifications of test problems

Number of producers	Number of depots	Number of plants
15	2, 3	2, 3
20	2, 3	2, 3
40	2, 3	2, 3
100	2, 3, 6	2, 3, 6
200	3, 6	3, 6

The production levels and period weights are the same as in Dayarian et al. [45]; Table 5.5 gives the production levels and weights.

Table 5.5: Weight and production-level distribution of the periods

# periods	Type 1		Type 2		Type 3		Type 4		Type 5	
	P_s	$W_\xi\%$	P_ξ	$W_\xi\%$	P_ξ	$W_\xi\%$	P_ξ	$W_\xi\%$	P_ξ	$W_\xi\%$
	1.00	60	1.00	50	1.00	40	1.00	30	1.00	20
4	1.30	20	1.30	25	1.20	35	1.10	30	1.10	40
	1.50	10	1.50	15	1.35	20	1.20	25	1.30	30
	1.70	10	1.70	10	1.50	15	1.40	15	1.70	10
	P_ξ	$W_\xi\%$	P_ξ	$W_\xi\%$	P_ξ	$W_\xi\%$	P_ξ	$W_\xi\%$	P_ξ	$W_\xi\%$
	1.00	60	1.00	50	1.00	40	1.00	30	1.00	20
5	1.30	15	1.30	20	1.20	25	1.10	25	1.10	35
	1.50	15	1.50	15	1.35	20	1.20	20	1.20	25
	1.70	5	1.70	10	1.50	10	1.40	15	1.40	15
	1.90	5	1.90	5	1.65	5	1.70	10	1.70	5

We ran our ALNS algorithm for each of the above test problems and investigated its performance in terms of solution quality and computational efficiency. The algorithm was coded in C++ and the tests were run on computers with a 2.67 GHz processor and 24 GB of RAM.

5.7.2 Parameter settings and sensitivity analysis

Similarly to most metaheuristics, changing the values of the parameters may affect the performance (but not the correctness) of the algorithm.

We tune the parameters via a blackbox optimizer to be described later. One drawback of this optimizer is that as the number of parameters increases, the accuracy of the algorithm decreases considerably. We therefore apply a two-phase procedure, based on extensive preliminary tests. It divides the parameters into two subsets. The first subset

contains the less sensitive parameters, and the second subset includes the parameters with a greater impact on the performance of the algorithm. We tune the parameters in the first subset by trial-and-error; Table 5.6 gives the resulting values.

Table 5.6: Parameter values found by trial and error

	Parameter	Value
$[q_{min}, q_{max}]$	Bounds on number of nodes removed q	$[\min(5, 0.05\|\mathcal{N}\|), \min(20, 0.4\|\mathcal{N}\|)]$
$Iter^{adj}$	Number of iterations after which η is updated	20
$Iter^{his}$	History used to update η	100
δmin and δmax	Bounds on number of infeasible solutions used to update η	30 and 45
β_1, β_2, β_3	Lengths of lists in central memory	20, 20, 10
λ	Sparsification factor in granularity threshold	1
$\sigma_1, \sigma_2, \sigma_3, \sigma_4$	Bonus factors for adaptive weight adjustment	1, 1, 1, 2

We set the initial temperature to $T^{init} = \frac{0.05C(s_0)}{|\mathcal{N}|\ln(0.5)}$, where $C(s_0)$ is the value of the initial solution. By Eq. (5.4), setting the initial temperature to $\frac{0.05C(s_0)}{\ln(0.5)}$ allows us to accept solutions that are 5% different from the current solution with a probability of 50%. Our preliminary tests showed that dividing this value by the number of producers improved the results; similar results were reported by Pisinger and Ropke [121]. We set the final temperature to $T^{fin} = T^{init}.c^{25000}$, allowing a minimum of 25000 iterations.

We tune the parameters in the second subset by first determining a range for each parameter based on extensive preliminary tests. We then find the best value for each parameter using the Opal algorithm [6]. Opal takes an algorithm and a parameter vector as input, and it outputs parameter values based on a user-defined performance measure. Opal models the problem as a blackbox optimization which is then solved by a state-of-the-art direct search solver; see Audet et al. [6].

To define a performance measure for Opal, we selected a restricted set of training instances. This set included instances ranging from 20 to 200 producer nodes, with 2 to 6 depots and plants. For a given vector of parameters, we ran each instance five times and recorded the average objective function value. The performance measure is defined to be the geometric mean of the average values of the training instances. Table 5.7 gives the values found for the second subset of parameters.

Table 5.7: Parameter values found using Opal

	Parameter	Range	Value
δ	Default segment length	[50, 150]	70
φ	Inner loop length	[3, 7]	6
γ	Number of segments to update operator weights	[1, 4]	2
α	Impact of long-term/short-term memory in weight update	[0, 1]	0.25
c	Cooling rate for SA	[0.9980, 0.9998]	0.9987
r	Acceptance radius gap in diversity segments	[0.01, 0.07]	0.05
$nbSegm^{DIV}$	Call diversity segment after observing no improvement in this number of segments	[25, 100]	45

5.7.3 Evaluating the contributions of the heuristics

Table 5.8 provides statistics on the removal and insertion heuristics. We ran each instance five times while excluding one heuristic and keeping the others. For each instance, we recorded the average result over the five runs. The values in Table 5.8 indicate the degradation in the geometric mean of the values obtained for all the instances in the training set. We use the geometric mean because the training set includes problems of different sizes with varying objective values. With the geometric mean the smaller instances are not dominated by the larger ones.

The plant-producer-related removal is the most efficient removal heuristic, followed by the route removal and smart removal heuristics. Minimum-loss insertion is the most useful insertion heuristic, followed by the regret insertion heuristic. We do not include the specialized operators in this evaluation because their main goal is to create diversity in the search. However, we have studied the impact of excluding the diversity segment. Our tests on the training set show that the solutions found without the diversity segment are on average 0.01% better. However, in some cases, particularly for smaller instances, the diversity segment helps us to escape from local optima. We have also evaluated the LS operators for the training set. The solutions found without these operators are on average 0.16% worse.

Table 5.8: Evaluation of contribution of each heuristic

Heuristic	Solution degradation without this heuristic (%)
Random Removal	-0.03
Worst Removal	0.00
Route Removal	0.05
Cluster Removal	0.02
Smart Removal	0.05
Solution-Cost-Based Related Removal	0.02
Route-Cost-Based Related Removal	0.04
Paired-Related Removal	-0.03
Route-Related Removal	0.02
Depot-Producer-Related Removal	0.04
Plant-Producer-Related Removal	0.07
Sequential Insertion	-0.01
Sequential Insertion with Plant Satisfaction	0.03
Best-First Insertion	0.02
Regret Insertion	0.04
Minimum-Loss Insertion	0.07

5.7.4 Computational results

Table 5.9 presents the results of applying our algorithm to the instances described in Section 5.7.1. In this table,

ALNS best is the average of the best solutions found;

ALNS avg. is the mean value of the average of the solutions found over the five runs;

% dev. is the average of standard deviation over the five runs;

T (s) is the average computational time.

Detailed results for each instance are given in Tables 5.10–5.16 in the Appendix. The standard deviations reported in Table 5.9 are based on the routing costs; the fixed vehicle costs have been removed. Tables 5.10–5.16 report the deviations based on both the total cost and the routing costs.

Table 5.9: Results for instances of different sizes

Instance size	ALNS best	ALNS avg.	% dev.	T (s)
15	5074.80	5074.80	0.00	5
20	5935.33	5935.68	0.05	7
40	11551.42	11552.10	0.04	26
100	31951.60	31976.71	0.48	129
200	53601.86	53654.35	0.51	235

For the smaller instances, optimal solutions reported by Dayarian et al. [45]. In Tables 5.10 and 5.11, these solutions are given in column BKS DCGR. For the larger instances, we generate lower and upper bounds as described in Section 5.6. The lower bound has two parts: 1) the value of the optimal solution for the VRP for the reference period, and 2) a lower bound on the total recourse cost, obtained by solving the bin-packing formulations in Section 5.6. We used Cplex 12.2 to solve these problems. We compute the upper bound by evaluating the cost of the VRP for the reference period, based on the objective function of the multi-period problem. We adapt the algorithm proposed by Dayarian et al. [44] for a similar problem to solve the VRP for the reference period. This algorithm can solve problems with up to 50 producers; we do not report bounds for larger problems.

Table 5.10 gives the results for the instances with 15 producers. For the instances with 2 or 3 depots and plants and 4 or 5 periods, our algorithm was able to find the optimal solutions with a standard deviation of zero. The computational time is about 1/80th of that required by the branch-and-price algorithm of Dayarian et al. [45].

Table 5.11 gives the results for the instances with 20 producers for which the optimal solutions are available. These instances have 2 or 3 depots and 3 plants. Table 5.12 gives the results for the instances with 20 producers for which the optimal solutions are unknown. For 18 of the instances in Table 5.11, every run of the algorithm found the optimal solution. For 19 of the instances in Table 5.12, our solution lies between the computed bounds. We also calculate the value of $\frac{LB}{ALNS_{best}}$. For Table 5.11, $ALNS_{best}$ is the optimal value for each instance; the average value of this ratio is 0.991. For Table 5.12, the average value is 0.989. This comparative factor between instances in Tables 5.11 and 5.12 indicates the quality of the solutions obtained for instances with 20 producers, for which the optimal solutions are available.

For the instances with 40 producers, all the solutions found lie between the computed bounds. The computational time is less than 2% of the time needed to solve the single-period VRP using the branch-and-price algorithm. The results for the instances with 100 and 200 producers show that larger problems are more difficult. Increasing the number of plants has a greater impact than increasing the number of depots, on both the computational time and the deviation from the best solution.

5.8 Conclusions

We have investigated the design of tactical plans for a transportation problem inspired by real-world milk collection in Quebec. To take the seasonal variations into account, we modeled the problem as a multi-period VRP. We developed an ALNS algorithm incorporating several heuristics for this VRP.

We tested the algorithm on a large set of instances of different sizes. The results for the smaller instances were compared with the existing exact solutions in the literature. For the larger instances, where optimal solutions were not available, we computed lower and upper bounds on the value of the solution.

Future research will include more attributes and constraints such as soft time windows on the collection, restrictions on the route length, and heterogeneous fleets of vehicles. We also plan to consider the situation where a vehicle may perform several deliveries to more than one plant per day. It would also be interesting to take into account the daily variations in the production levels. This transforms the problem into a VRP with stochastic demands.

Table 5.10: Results for instances with 15 producers

	Instance	BKS DCGR	T (s)	ALNS best	ALNS avg. over 5	% dev total cost	% dev routing cost	T (s)
	pr-15-2D2P4S-T1	4353.62	19	**4353.62**	**4353.62**	0	0	3
2 depots	pr-15-2D2P4S-T2	4395.48	26	**4052.82**	**4052.82**	0	0	4
2 plants	pr-15-2D2P4S-T3	4478.78	24	**5930.40**	**5930.40**	0	0	6
4 periods	pr-15-2D2P4S-T4	4434.82	13	**4395.48**	**4395.48**	0	0	4
	pr-15-2D2P4S-T5	4472.04	7	**4090.51**	**4090.51**	0	0	4
	pr-15-2D2P5S-T1	4358.84	22	**4478.78**	**4478.78**	0	0	3
2 depots	pr-15-2D2P5S-T2	4403.64	30	**4148.54**	**4148.54**	0	0	4
2 plants	pr-15-2D2P5S-T3	4439.31	16	**5959.89**	**5959.89**	0	0	6
5 periods	pr-15-2D2P5S-T4	4449.81	9	**4434.82**	**4434.82**	0	0	3
	pr-15-2D2P5S-T5	4476.56	7	**4115.57**	**4115.57**	0	0	4
	pr-15-2D3P4S-T1	5855.70	1422	**5894.16**	**5894.16**	0	0	5
2 depots	pr-15-2D3P4S-T2	5860.58	911	**4472.04**	**4472.04**	0	0	3
3 plants	pr-15-2D3P4S-T3	5831.72	964	**4158.00**	**4158.00**	0	0	4
4 periods	pr-15-2D3P4S-T4	5821.90	998	**5837.27**	**5837.27**	0	0	5
	pr-15-2D3P4S-T5	5843.12	947	**4358.84**	**4358.84**	0	0	4
	pr-15-2D3P5S-T1	5871.89	1103	**5898.82**	**5898.82**	0	0	5
2 depots	pr-15-2D3P5S-T2	5886.37	928	**4055.18**	**4055.18**	0	0	4
3 plants	pr-15-2D3P5S-T3	5843.29	972	**4403.64**	**4403.64**	0	0	4
5 periods	pr-15-2D3P5S-T4	5843.12	980	**4098.60**	**4098.60**	0	0	4
	pr-15-2D3P5S-T5	5832.51	874	**5945.35**	**5945.35**	0	0	6
	pr-15-3D2P4S-T1	4052.82	58	**4439.31**	**4439.31**	0	0	3
3 depots	pr-15-3D2P4S-T2	4090.51	23	**4118.45**	**4118.45**	0	0	4
2 plants	pr-15-3D2P4S-T3	4148.54	34	**5985.97**	**5985.97**	0	0	7
4 periods	pr-15-3D2P4S-T4	4115.57	18	**4132.68**	**4132.68**	0	0	4
	pr-15-3D2P4S-T5	4158.00	34	**4449.81**	**4449.81**	0	0	3
	pr-15-3D2P5S-T1	4055.18	30	**5900.75**	**5900.75**	0	0	5
3 depots	pr-15-3D2P5S-T2	4098.60	27	**4476.56**	**4476.56**	0	0	3
2 plants	pr-15-3D2P5S-T3	4118.45	28	**5896.57**	**5896.57**	0	0	6
5 periods	pr-15-3D2P5S-T4	4132.68	24	**4152.31**	**4152.31**	0	0	4
	pr-15-3D2P5S-T5	4152.31	17	**5866.92**	**5866.92**	0	0	6
	pr-15-3D3P4S-T1	5930.40	759	**5843.12**	**5843.12**	0	0	6
3 depots	pr-15-3D3P4S-T2	5959.89	838	**5871.89**	**5871.89**	0	0	6
3 plants	pr-15-3D3P4S-T3	5894.16	600	**5843.12**	**5843.12**	0	0	6
4 periods	pr-15-3D3P4S-T4	5837.27	447	**5855.70**	**5855.70**	0	0	6
	pr-15-3D3P4S-T5	5898.82	764	**5886.37**	**5886.37**	0	0	6
	pr-15-3D3P5S-T1	5945.35	717	**5821.90**	**5821.90**	0	0	6
3 depots	pr-15-3D3P5S-T2	5985.97	835	**5860.58**	**5860.58**	0	0	6
3 plants	pr-15-3D3P5S-T3	5900.75	615	**5843.29**	**5843.29**	0	0	6
5 periods	pr-15-3D3P5S-T4	5896.57	561	**5831.72**	**5831.72**	0	0	6
	pr-15-3D3P5S-T5	5866.92	473	**5832.51**	**5832.51**	0	0	6
	Avg.	5074.80	429	**5074.80**	**5074.80**	0	0	5

Table 5.11: Results for instances with 20 producers with available optimal solutions

	Instance	LB	BKS DCGR	LB/BKS	T (s)	ALNS best over 5	ALNS avg. over 5	% dev total cost	% dev routing cost	T (s)
	pr-20-2D3P4S-T1	5810.84	5873.92	0.989	3934	**5873.92**	**5873.92**	0.00	0.00	6
2 depots	pr-20-2D3P4S-T2	5827.34	5907.21	0.986	7338	**5907.21**	**5907.21**	0.00	0.00	6
3 plants	pr-20-2D3P4S-T3	5851.1	5890.07	0.993	4431	**5890.07**	**5890.07**	0.00	0.00	7
4 periods	pr-20-2D3P4S-T4	5790.35	5807.1	0.997	4240	**5807.1**	**5807.1**	0.00	0.00	7
	pr-20-2D3P4S-T5	5789.32	5861.86	0.988	3617	**5861.86**	**5861.86**	0.00	0.00	6
	pr-20-2D3P5S-T1	5827.04	5883.98	0.990	3371	**5883.98**	**5883.98**	0.00	0.00	7
2 depots	pr-20-2D3P5S-T2	5848.16	5919.85	0.988	3422	**5919.85**	**5919.85**	0.00	0.00	7
3 plants	pr-20-2D3P5S-T3	5854.21	5888.36	0.994	4814	**5888.36**	**5888.36**	0.00	0.00	7
5 periods	pr-20-2D3P5S-T4	5826.85	5856.77	0.995	2239	**5856.77**	**5856.77**	0.00	0.00	7
	pr-20-2D3P5S-T5	5808.6	5833.37	0.996	2166	**5833.37**	**5833.37**	0.00	0.00	7
	pr-20-3D3P4S-T1	5951.3	6013.02	0.990	13090	**6013.02**	**6013.02**	0.00	0.00	8
3 depots	pr-20-3D3P4S-T2	5974.3	6043.07	0.989	9930	**6043.07**	**6043.07**	0.00	0.00	8
3 plants	pr-20-3D3P4S-T3	5950.7	6026.6	0.987	10404	**6026.6**	**6026.6**	0.00	0.00	8
4 periods	pr-20-3D3P4S-T4	5898.75	5948.7	0.992	6706	**5948.7**	**5948.7**	0.00	0.00	7
	pr-20-3D3P4S-T5	5917.5	5980.32	0.989	7072	**5980.32**	5981.44	0.03	0.12	8
	pr-20-3D3P5S-T1	5979.4	6032.42	0.991	13749	**6032.42**	**6032.42**	0.00	0.00	8
3 depots	pr-20-3D3P5S-T2	6002.4	6067.63	0.989	14925	**6067.63**	**6067.63**	0.00	0.00	8
3 plants	pr-20-3D3P5S-T3	5962.9	6037.43	0.988	9448	**6037.43**	6038.69	0.03	0.11	8
5 periods	pr-20-3D3P5S-T4	5950.56	6016.16	0.989	7633	**6016.16**	**6016.16**	0.00	0.00	8
	pr-20-3D3P5S-T5	5924.85	5982.43	0.990	5735	**5982.43**	**5982.43**	0.00	0.00	8
	Avg.	5887.324	5943.514	0.991	6913	5943.5135	5943.63	0.00	0.01	7

Table 5.12: Results for instances with 20 producers without available optimal solutions

	Instance	LB	UB	ALNS best over 5	LB/ALNS best	ALNS avg. over 5	% dev total cost	% dev routing cost	T (s)
	pr-20-2D2P4S-T1	6162.22	6301	6237.18	0.988	6237.18	0.00	0.00	7
2 depots	pr-20-2D2P4S-T2	6186.47	6366.29	6266.92	0.987	6266.92	0.00	0.00	7
2 plants	pr-20-2D2P4S-T3	6158.79	6499.43	6226.73	0.989	6226.73	0.00	0.00	7
4 periods	pr-20-2D2P4S-T4	6121.98	6481.62	6178.21	0.991	6178.21	0.00	0.00	6
	pr-20-2D2P4S-T5	6185.52	6539.06	6246.02	0.990	6246.02	0.00	0.00	7
	pr-20-2D2P5S-T1	6182.71	6302.52	6259.28	0.988	6259.28	0.00	0.00	7
2 depots	pr-20-2D2P5S-T2	6220.79	6367.63	6301.07	0.987	6301.07	0.00	0.00	8
2 plants	pr-20-2D2P5S-T3	6160.79	6431.58	6218.3	0.991	6218.3	0.00	0.00	7
5 periods	pr-20-2D2P5S-T4	6172.34	6484.31	6258.85	0.986	6258.85	0.00	0.00	7
	pr-20-2D2P5S-T5	6158.74	6543.94	6230.57	0.988	6230.57	0.00	0.00	7
	pr-20-3D2P4S-T1	5552.08	5602.42	5588.46	0.993	5588.46	0.00	0.00	6
3 depots	pr-20-3D2P4S-T2	5578.15	5640.67	5604.2	0.995	5604.2	0.00	0.00	6
2 plants	pr-20-3D2P4S-T3	5542.82	5639.71	5627.18	0.985	5627.86	0.03	0.13	6
4 periods	pr-20-3D2P4S-T4	5520.98	5597.4	5597.4	0.986	5600.46	0.12	0.62	6
	pr-20-3D2P4S-T5	5550.18	5623.64	5623.64	0.987	5623.64	0.00	0.00	7
	pr-20-3D2P5S-T1	5553.98	5614.93	5597.81	0.992	5597.81	0.00	0.00	7
3 depots	pr-20-3D2P5S-T2	5565.85	5639.79	5616.21	0.991	5616.21	0.00	0.00	6
2 plants	pr-20-3D2P5S-T3	5548.67	5626.75	5620.96	0.987	5620.96	0.00	0.00	7
5 periods	pr-20-3D2P5S-T4	5543.6	5622.43	5622.43	0.986	5624.03	0.04	0.18	7
	pr-20-3D2P5S-T5	5532.29	5621.31	5621.31	0.984	5627.81	0.15	0.75	7
	Avg.			5927.14	0.989	5927.73	0.02	0.08	7

Table 5.13: Results for instances with 40 producers

	Instance	Bounds on opt. sol.	T (s)	ALNS best	ALNS avg. over 5	% dev total cost	% dev routing cost	T (s)
	pr-40-2D2P4S-T1	[12229.2, 12405.5]	5441	12389.5	12389.5	0.00	0.00	19
2 depots	pr-40-2D2P4S-T2	[12336, 12558.3]	5337	12535.1	12535.1	0.00	0.00	20
2 plants	pr-40-2D2P4S-T3	[12556.2, 12780.2]	5342	12752.5	12754.4	0.02	0.08	19
4 periods	pr-40-2D2P4S-T4	[12569.8, 12700.3]	5386	12679.6	12679.6	0.00	0.00	19
	pr-40-2D2P4S-T5	[12700.4, 12881.6]	5416	12856.6	12860.2	0.04	0.15	21
	pr-40-2D2P5S-T1	[12241.8, 12415.4]	5344	12398.7	12398.7	0.00	0.00	22
2 depots	pr-40-2D2P5S-T2	[12359.1, 12574.4]	5374	12555	12555	0.00	0.00	21
2 plants	pr-40-2D2P5S-T3	[12463.5, 12652]	5358	12632.8	12632.8	0.00	0.00	22
5 periods	pr-40-2D2P5S-T4	[12609.8, 12742.5]	5344	12730.4	12730.4	0.00	0.00	22
	pr-40-2D2P5S-T5	[12693.3, 12844.1]	5352	12825.6	12825.6	0.00	0.00	22
	pr-40-2D3P4S-T1	[11826.5, 12019.5]	1540	11956.1	11959.6	0.04	0.15	27
2 depots	pr-40-2D3P4S-T2	[11893.3, 12135.6]	1536	12039.1	12040	0.02	0.06	26
3 plants	pr-40-2D3P4S-T3	[11933.6, 12188.6]	1540	12093.4	12093.4	0.00	0.00	26
4 periods	pr-40-2D3P4S-T4	[11838.9, 12033.8]	1532	11965.4	11965.4	0.00	0.00	26
	pr-40-2D3P4S-T5	[11902.4, 12191.2]	1540	12072.3	12073.6	0.02	0.07	28
	pr-40-2D3P5S-T1	[11846.3, 12040.4]	1525	11984.5	11986.5	0.02	0.08	29
2 depots	pr-40-2D3P5S-T2	[11916.4, 12157.7]	1535	12074.6	12075.9	0.02	0.07	28
3 plants	pr-40-2D3P5S-T3	[11899.8, 12120.9]	1725	12045.8	12045.8	0.00	0.00	28
5 periods	pr-40-2D3P5S-T4	[11911.8, 12110.6]	1515	12068.7	12068.7	0.00	0.00	29
	pr-40-2D3P5S-T5	[11895.3, 12128.5]	1534	12046.8	12046.8	0.00	0.00	28
	pr-40-3D2P4S-T1	[9681.72, 9862.4]	640	9794.28	9794.28	0.00	0.00	27
3 depots	pr-40-3D2P4S-T2	[9725.53, 9955.19]	640	9860.69	9860.69	0.00	0.00	27
2 plants	pr-40-3D2P4S-T3	[9770.77, 10051]	634	9916.4	9917.29	0.02	0.08	26
4 periods	pr-40-3D2P4S-T4	[9655.76, 9937.56]	650	9763.42	9768.03	0.07	0.29	26
	pr-40-3D2P4S-T5	[9726.14, 10051.9]	648	9824.06	9824.06	0.00	0.00	25
	pr-40-3D2P5S-T1	[9688.24, 9873.74]	641	9812.15	9812.64	0.01	0.05	29
3 depots	pr-40-3D2P5S-T2	[9748.89, 9974.3]	648	9888.96	9888.96	0.00	0.00	30
2 plants	pr-40-3D2P5S-T3	[9749.21, 9990.35]	635	9892.71	9893.51	0.01	0.04	29
5 periods	pr-40-3D2P5S-T4	[9728.32, 9990.4]	638	9863.89	9863.89	0.00	0.00	30
	pr-40-3D2P5S-T5	[9704.04, 10017.8]	644	9829.02	9830.13	0.02	0.11	30
	pr-40-3D3P4S-T1	[11525.3, 11697.1]	229	11642.8	11642.8	0.00	0.00	24
3 depots	pr-40-3D3P4S-T2	[11569.3, 11788.6]	233	11709.7	11709.7	0.00	0.00	24
3 plants	pr-40-3D3P4S-T3	[11618, 11833.3]	228	11718.2	11718.2	0.00	0.00	25
4 periods	pr-40-3D3P4S-T4	[11525.5, 11675.7]	233	11624.6	11629.1	0.06	0.27	26
	pr-40-3D3P4S-T5	[11570.5, 11807.1]	228	11727.6	11727.8	0.00	0.02	27
	pr-40-3D3P5S-T1	[11530.4, 11701.8]	233	11654	11654	0.00	0.00	25
3 depots	pr-40-3D3P5S-T2	[11592.2, 11802.1]	221	11736	11736	0.00	0.00	28
3 plants	pr-40-3D3P5S-T3	[11595, 11779.8]	229	11688.2	11688.2	0.00	0.00	27
5 periods	pr-40-3D3P5S-T4	[11593.8, 11742.4]	252	11718.6	11718.6	0.00	0.00	28
	pr-40-3D3P5S-T5	[11576.9, 11743.8]	231	11688.9	11688.9	0.00	0.00	30
	Avg.	[11412.5, 11623.9]	1949	11551.417	11552.1	0.01	0.04	26

Table 5.14: Results for instances with 100 producers (1)

	Instance	ALNS best	ALNS avg. over 5	% dev total cost	% dev routing cost	T (s)
	pr-100-2D2P4S-T1	29519.2	29532.1	0.05	0.21	72
2 depots	pr-100-2D2P4S-T2	29831.6	29838.2	0.03	0.14	71
2 plants	pr-100-2D2P4S-T3	30193.8	30204.8	0.05	0.18	73
4 periods	pr-100-2D2P4S-T4	29968.2	29988.2	0.09	0.35	74
	pr-100-2D2P4S-T5	30251.3	30268	0.07	0.28	77
	pr-100-2D2P5S-T1	29580.4	29591.2	0.04	0.18	79
2 depots	pr-100-2D2P5S-T2	29892.1	29910.4	0.07	0.29	78
2 plants	pr-100-2D2P5S-T3	29965	29988.7	0.09	0.37	77
5 periods	pr-100-2D2P5S-T4	30100.9	30119	0.07	0.29	83
	pr-100-2D2P5S-T5	30228.9	30248.8	0.08	0.33	85
	pr-100-2D3P4S-T1	26407.4	26416.5	0.04	0.22	57
2 depots	pr-100-2D3P4S-T2	26585.5	26609.1	0.10	0.48	56
3 plants	pr-100-2D3P4S-T3	26830.6	26857.3	0.12	0.56	60
4 periods	pr-100-2D3P4S-T4	26666.9	26710.9	0.19	0.92	60
	pr-100-2D3P4S-T5	26925.1	26939.5	0.07	0.32	57
	pr-100-2D3P5S-T1	26415.1	26430.8	0.07	0.36	61
2 depots	pr-100-2D3P5S-T2	26626.5	26648.8	0.09	0.45	63
3 plants	pr-100-2D3P5S-T3	26671.8	26691.4	0.09	0.43	61
5 periods	pr-100-2D3P5S-T4	26786	26832.1	0.22	1.00	63
	pr-100-2D3P5S-T5	26859.8	26920	0.27	1.26	66
	pr-100-2D6P4S-T1	26940.4	26964.9	0.10	0.47	98
2 depots	pr-100-2D6P4S-T2	27148.6	27179.1	0.14	0.60	99
6 plants	pr-100-2D6P4S-T3	27418.9	27462.9	0.19	0.81	113
4 periods	pr-100-2D6P4S-T4	27164.5	27178.7	0.08	0.35	119
	pr-100-2D6P4S-T5	27413.6	27464.6	0.25	1.05	114
	pr-100-2D6P5S-T1	26946.5	26980.7	0.15	0.67	114
2 depots	pr-100-2D6P5S-T2	27171.6	27218.3	0.20	0.87	116
6 plants	pr-100-2D6P5S-T3	27225.8	27248.9	0.11	0.47	121
5 periods	pr-100-2D6P5S-T4	27338.8	27347.6	0.04	0.18	130
	pr-100-2D6P5S-T5	27430.4	27451.9	0.10	0.44	131
	pr-100-3D2P4S-T1	23774.1	23791.6	0.11	0.43	89
3 depots	pr-100-3D2P4S-T2	24038.8	24049.3	0.05	0.20	86
2 plants	pr-100-3D2P4S-T3	24269.8	24296.2	0.14	0.52	92
4 periods	pr-100-3D2P4S-T4	24070.5	24084.5	0.08	0.33	83
	pr-100-3D2P4S-T5	24289.4	24300.6	0.06	0.24	85
	pr-100-3D2P5S-T1	23808.4	23811	0.02	0.07	86
3 depots	pr-100-3D2P5S-T2	24062.1	24073.7	0.06	0.22	97
2 plants	pr-100-3D2P5S-T3	24110.6	24127.9	0.10	0.38	97
5 periods	pr-100-3D2P5S-T4	24204.8	24233.4	0.14	0.53	106
	pr-100-3D2P5S-T5	24311	24315.2	0.03	0.11	107
	pr-100-6D2P4S-T1	26283.4	26289.4	0.03	0.15	95
6 depots	pr-100-6D2P4S-T2	26482.9	26487.5	0.02	0.10	94
2 plants	pr-100-6D2P4S-T3	26721.2	26724.1	0.01	0.06	110
4 periods	pr-100-6D2P4S-T4	26557	26572	0.07	0.34	89
	pr-100-6D2P4S-T5	26790.2	26812.8	0.10	0.45	116
	pr-100-6D2P5S-T1	26319.1	26324.4	0.03	0.12	100
6 depots	pr-100-6D2P5S-T2	26533.7	26541.4	0.03	0.17	121
2 plants	pr-100-6D2P5S-T3	26592	26598.1	0.03	0.13	116
5 periods	pr-100-6D2P5S-T4	26710.7	26729.7	0.08	0.39	122
	pr-100-6D2P5S-T5	26793.7	26801.2	0.04	0.16	113
	Avg.	33630.72	33655.19	0.11	0.49	113

Table 5.15: Results for instances with 100 producers (2)

	Instance	ALNS best	ALNS avg. over 5	% dev total cost	% dev routing cost	T (s)
	pr-100-3D3P4S-T1	27704.8	27740	0.14	0.59	101
3 depots	pr-100-3D3P4S-T2	27904.7	27927.7	0.11	0.45	102
3 plants	pr-100-3D3P4S-T3	28143.9	28163.3	0.09	0.35	101
4 periods	pr-100-3D3P4S-T4	27803.8	27852.6	0.20	0.82	107
	pr-100-3D3P4S-T5	28037.4	28081.3	0.18	0.70	110
	pr-100-3D3P5S-T1	27768.7	27779.1	0.05	0.19	107
3 depots	pr-100-3D3P5S-T2	27990.1	28015.9	0.10	0.42	107
3 plants	pr-100-3D3P5S-T3	28006.1	28023.4	0.07	0.28	111
5 periods	pr-100-3D3P5S-T4	28038	28067.5	0.14	0.54	119
	pr-100-3D3P5S-T5	28067.9	28076.4	0.04	0.15	121
	pr-100-3D6P4S-T1	33482.8	33489.6	0.03	0.14	134
3 depots	pr-100-3D6P4S-T2	33605.1	33652.9	0.16	0.81	136
6 plants	pr-100-3D6P4S-T3	33501.3	33534.9	0.11	0.59	148
4 periods	pr-100-3D6P4S-T4	33185.2	33195.4	0.04	0.22	150
	pr-100-3D6P4S-T5	33413.7	33435.1	0.08	0.42	157
	pr-100-3D6P5S-T1	33531.2	33560.7	0.10	0.52	139
3 depots	pr-100-3D6P5S-T2	33751.2	33760.5	0.04	0.19	141
6 plants	pr-100-3D6P5S-T3	33512.3	33540.2	0.09	0.48	154
5 periods	pr-100-3D6P5S-T4	33500.2	33519.6	0.07	0.35	163
	pr-100-3D6P5S-T5	33345.4	33362.3	0.06	0.30	162
	pr-100-6D3P4S-T1	26829.5	26838.3	0.04	0.18	109
6 depots	pr-100-6D3P4S-T2	27056.5	27069	0.05	0.24	112
3 plants	pr-100-6D3P4S-T3	27256.4	27289.8	0.14	0.60	115
4 periods	pr-100-6D3P4S-T4	26980.3	26994.1	0.06	0.27	122
	pr-100-6D3P4S-T5	27225.3	27239.1	0.07	0.30	125
	pr-100-6D3P5S-T1	26852.4	26858.2	0.03	0.13	114
6 depots	pr-100-6D3P5S-T2	27089.7	27110.9	0.09	0.40	117
3 plants	pr-100-6D3P5S-T3	27140.4	27152.4	0.05	0.23	120
5 periods	pr-100-6D3P5S-T4	27147.4	27177.6	0.13	0.57	129
	pr-100-6D3P5S-T5	27176.4	27190.5	0.06	0.28	132
	pr-100-6D6P4S-T1	30673	30705.2	0.15	0.68	210
6 depots	pr-100-6D6P4S-T2	30878.8	30919.7	0.16	0.70	208
6 plants	pr-100-6D6P4S-T3	31076	31109.8	0.15	0.66	218
4 periods	pr-100-6D6P4S-T4	30795.9	30824.8	0.12	0.53	226
	pr-100-6D6P4S-T5	31072.9	31099.1	0.10	0.46	231
	pr-100-6D6P5S-T1	30729	30754.4	0.10	0.47	215
6 depots	pr-100-6D6P5S-T2	30929.2	30974.6	0.19	0.83	216
6 plants	pr-100-6D6P5S-T3	30995.4	31027.9	0.13	0.58	223
5 periods	pr-100-6D6P5S-T4	30964.4	31026.3	0.24	1.07	233
	pr-100-6D6P5S-T5	30943.6	31002.7	0.27	1.19	240
	Avg.	29852.7	29878.6	0.11	0.47	150

Table 5.16: Results for instances with 200 producers

	Instance	ALNS best	ALNS avg. over 5	% dev total cost	% dev routing cost	T (s)
	pr-200-3D3P4S-T1	53888	53915.7	0.06	0.26	167
3 depots	pr-200-3D3P4S-T2	54490.3	54514.3	0.06	0.22	165
3 plants	pr-200-3D3P4S-T3	55283.6	55326.7	0.10	0.36	172
4 periods	pr-200-3D3P4S-T4	54907	54985.3	0.17	0.63	186
	pr-200-3D3P4S-T5	55642	55682.7	0.09	0.33	192
	pr-200-3D3P5S-T1	53968.6	53995.9	0.06	0.24	174
3 depots	pr-200-3D3P5S-T2	54525.8	54564.7	0.08	0.33	176
3 plants	pr-200-3D3P5S-T3	54817.5	54860.6	0.09	0.35	182
5 periods	pr-200-3D3P5S-T4	55176.1	55210.7	0.08	0.30	184
	pr-200-3D3P5S-T5	55519.3	55551.3	0.08	0.29	195
	pr-200-3D6P4S-T1	49392.1	49440.8	0.11	0.54	207
3 depots	pr-200-3D6P4S-T2	49871.4	49977.6	0.26	1.20	202
6 plants	pr-200-3D6P4S-T3	50415.7	50505.7	0.21	0.95	194
4 periods	pr-200-3D6P4S-T4	50163.1	50193.6	0.08	0.36	202
	pr-200-3D6P4S-T5	50753.3	50767.6	0.04	0.17	204
	pr-200-3D6P5S-T1	49435.4	49509.5	0.17	0.82	217
3 depots	pr-200-3D6P5S-T2	49913.3	50014.5	0.23	1.06	215
6 plants	pr-200-3D6P5S-T3	50124.4	50178.7	0.12	0.56	213
5 periods	pr-200-3D6P5S-T4	50382.9	50439.2	0.13	0.56	212
	pr-200-3D6P5S-T5	50668.5	50700.9	0.08	0.35	218
	pr-200-6D3P4S-T1	50071.9	50118	0.10	0.47	189
6 depots	pr-200-6D3P4S-T2	50576.1	50606.9	0.08	0.33	194
3 plants	pr-200-6D3P4S-T3	51270.7	51318.6	0.11	0.46	200
4 periods	pr-200-6D3P4S-T4	50913.8	50987.5	0.16	0.69	216
	pr-200-6D3P4S-T5	51526.7	51605.9	0.18	0.75	211
	pr-200-6D3P5S-T1	50113	50152.8	0.10	0.46	188
6 depots	pr-200-6D3P5S-T2	50644.7	50688.8	0.11	0.46	198
3 plants	pr-200-6D3P5S-T3	50942.2	50967.5	0.07	0.29	202
5 periods	pr-200-6D3P5S-T4	51235.6	51311.9	0.18	0.74	217
	pr-200-6D3P5S-T5	51426.1	51537	0.26	1.06	224
	pr-200-6D6P4S-T1	57968.5	58008.4	0.08	0.37	318
6 depots	pr-200-6D6P4S-T2	58522.4	58565.8	0.09	0.38	329
6 plants	pr-200-6D6P4S-T3	58977.3	59060.3	0.16	0.69	351
4 periods	pr-200-6D6P4S-T4	58463.3	58536	0.15	0.65	359
	pr-200-6D6P4S-T5	59006.3	59039.8	0.07	0.29	369
	pr-200-6D6P5S-T1	58006.9	58058.9	0.11	0.49	331
6 depots	pr-200-6D6P5S-T2	58597.2	58654.8	0.12	0.50	330
6 plants	pr-200-6D6P5S-T3	58701.8	58742.1	0.09	0.36	344
5 periods	pr-200-6D6P5S-T4	58880.1	58907.4	0.05	0.23	363
	pr-200-6D6P5S-T5	58891.3	58969.3	0.17	0.72	379
	Avg.	53601.855	53654.3425	0.12	0.51	235

Chapter 6

Conclusion

The VRP plays a vital role in the planning of many real-life collection/distribution systems such as garbage collection, mail delivery, and task sequencing. From a theoretical point of view, it also has drawn interest from many researchers during the past decades as a well-known combinatorial NP-hard problem. Due to significant economic benefits that can be achieved by optimizing logistics and transportation problems in practice, increased attention has been focused on various routing problems. Real-life problems often incorporate several attributes, constraints and characteristics and, consequently, get through the category of multi-attribute problems. Contrary to the classical VRP that focuses on the simplified models considering a small subset of existing attributes, the research on the multi-attribute extensions of the VRP attempts to simultaneously consider several complicating attributes encountered in real-life problems.

In this dissertation, we addressed a tactical planning problem arising in the milk collection-distribution activity in Quebec. This problem involves some special practical issues that are rarely considered by other authors in the field.

We first drew the sketch of a new class of problems, which is encountered in collection-distribution of perishable products such as milk, cheese, poultry and eggs. The characteristics and attributes of this class of problems are mainly inspired by the milk collection-distribution activity in Quebec. This problem deals with designing routing plans, where each route consists of two main phases. The first phase is expressed by a series of visits to a set of geographically dispersed producers in order to collect their product. The second phase consists of one or more visits to a set of processing plants to deliver the collected quantity of product.

The first paper focuses on a deterministic variant of the problem based on simple configuration of routes. It is characterized with an unlimited fleet of heterogeneous vehicles, departing from different depots to visit a set of producers. This problem involves constraints such as vehicle capacity, producer time windows and the necessity of satisfying plants' demands by delivering them adequate quantities of products. The goal of the considered problem is to minimize the total cost of a plan over the plan-

ning horizon. The cost of a plan consists of two main components: fixed and variable costs. The fixed costs are incurred when a vehicle is employed, while the variable costs depend of the total traveled distance. We presented a mathematical model for this problem taking the form of a set covering formulation. Our solution methodology was based on the branch-and-price framework. This approach deals with a master problem and a set of subproblems. The master problem is the linear relaxation of the proposed set covering. The subproblems, on the other hand, take the form of an elementary shortest path problem with resource constraints. In this particular problem, the resource constraints are associated with the vehicle capacity and producers' time windows. Solving the subproblems represents the bottleneck of the whole procedure. Therefore, we proposed a bilevel subproblem solving procedure, which is based on a dynamic-programming label-correcting procedure. The first level consists of a heuristic label-correcting procedure, in which an aggressive dominance rule is applied. Accordingly, a large portion of negative-reduced-cost columns is generated in a fraction of computational effort. However, the optimality is not guaranteed, since there may exist undiscovered negative-reduced-cost columns. Therefore, the second level consists of an exact label-correcting procedure, assuring the generation of all the remaining negative reduced-cost columns. We employed several cutting-edge strategies to accelerate the exact label-correcting procedure, and their performances are compared. Most of these strategies are based on partially relaxing elementarity condition and tightening it in the course of the procedure. Moreover, a new branching strategy, based on the structure of the problem, was proposed. This bi-level branching strategy first attempts to assign the producers to the plants through the branching decision. When all the producer-plant assignment variables are integer, the new branches are derived based on the flow variables. This branching strategy has been compared to another bi-level branching scheme; first branching on time windows and also, on flow variables. To evaluate the performance of this branch-and-price, a large set of randomly generated instances were considered. Different accelerating strategies, as well as, different branching strategies were compared on instances with up to fifty producers, three depots and three plants. The results show the high performance of the algorithm in terms of solution quality and computational efficiency.

In the second part of this dissertation, we addressed a new variant of the same problem class. In this variant, the assumptions regarding fixed production levels over a horizon are removed. Accordingly, two types of potential variations in terms of producers' production levels have been introduced; daily and seasonal variations. The seasonal variations occur periodically, following different seasonal characteristics, while the daily variations may take place on a random basis. In this work, we focused on inclusion of the seasonal variations in the routing plan design. In other words, the main goal of this problem is to design a unique routing plan over a horizon, while the existing seasonal variations, in terms of producers' production levels, are accounted for. The problem is modeled as a multi-period vehicle routing problem, in which each period represents a production season. A new set covering formulation was proposed. In this multi-period model, each period represents a given production level for each producer. The structure of this multi-period problem is very similar to the idea of an *a priori* optimization for a vehicle routing problem with stochastic demands. The solution

methodology proposed in this paper is based on the branch-and-price algorithm. Similar to the case of the first paper, the most significant, challenging, and time-consuming task was solving the subproblems. In this problem, one subproblem exists for each plant, which takes the form of a multi-period elementary shortest path problem with resource constraints. We introduced the non-monotonic resource consumption phenomena in the case of the MPESPPRC, which increases the difficulty of the dominance rules. To deal with this non-monotonicity phenomenon, we have proposed a bucket-based label storing strategy. A series of numerical tests was run to investigate the performance of the proposed algorithm. This algorithm is able to solve instances with twenty producers, three depots, three plants, and five periods.

To be able to address larger-scale problems, which are closer to real-life instances, in the third part of this thesis, a metaheuristic approach was proposed for the multi-period variant of the problem. This approach, based on the adaptive large neighbourhood search framework, includes several new destruction/construction operators, which are designed with insights from the structure of the problem. Moreover, some generic operators from the literature are adapted to the problem and are included in the algorithm. Also, in order to increase the efficiency of the search, a new diversity management procedure is proposed. The evaluation of the algorithm's performance is based on the following two criteria: 1) in the case of smaller instances, the solutions were compared to their optimal values obtained in the second paper, and 2) for the medium sized instances, a set of lower and upper bounds are computed. The lower bound is based on two components: 1) routing cost; a lower bound on the routing cost is obtained by solving a single variant of the problem, and 2) recourse cost; a lower bound on the recourse cost is obtained based on a bin-packing formulation. Results for instances with up to 200 producers, six depots, six plants and 5 periods are reported.

6.1 Future works and perspectives

We conclude this chapter by highlighting some research perspectives. Future works can be categorized in the three following ways:

1. As mentioned in Chapter 1, different route patterns are recognized in the DTP. All our problem descriptions in this dissertation were based on the simple route configuration. A research extension can address the single and multi-period variants of the problem, where more sophisticated route configurations are considered.

2. In the second and third papers of this thesis, we attempted to include the existing seasonal variations in the planning design. However, potential daily variations were neglected. Considering the potential daily fluctuations, in terms of producers' production level change the problem into a stochastic setting.

3. In the multi-period variant of the problem, no time window is considered for the producers. We plan to consider a new version of the problem, in which associated with each producer, there is a soft time window. The time windows of

the producers, being visited after a failure, may be violated, incurring a penalty. This penalty may be independent from the tardiness of the visit to the producer or as a function of temporal violation.

Bibliography

[1] 2010. URL `http://www.dairyinfo.gc.ca/`.

[2] 2013. URL `http://www.lait.org`.

[3] A. Ak and A. L. Erera. A paired-vehicle recourse strategy for the vehicle-routing problem with stochastic demands. *Transportation Science*, 41:222–237, 2007.

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

[5] T. Athanasopoulos. *The Multi-Period Vehicle Routing Problem and Its Applications*. PhD thesis, Department of Financial & Management Engineering, University of the Aegean, Chios, Greece, 2011.

[6] C. Audet, K.-C. Dang, and D. Orban. Optimization of algorithms with OPAL. Technical report, GERAD, 2012.

[7] N. Azi, M. Gendreau, and J.-Y. Potvin. An adaptive large neighborhood search for a vehicle routing problem with multiple trips. Technical Report CIRRELT-2010-08, CIRRELT, 2010.

[8] A. Bachem, W. Hochstättler, and M. Malich. The simulated trading heuristic for solving vehicle routing problems. *Discrete Applied Mathematics*, 65(1-3):47–72, 1996.

[9] P. Badeau, F. Guertin, M. Gendreau, J.-Y. Potvin, and E. Taillard. A parallel tabu search heuristic for the vehicle routing problem with time windows. *Transportation Research Part C: Emerging Technologies*, 5(2):109 – 122, 1997.

[10] B. M. Baker and M.A. Ayechew. A genetic algorithm for the vehicle routing problem. *Computers & Operations Research*, 30(5):787 – 800, 2003.

[11] R. Baldacci and A. Mingozzi. A unified exact method for solving different classes of vehicle routing problems. *Mathematical Programming*, 120(2):347–380, 2009.

[12] R. Baldacci, E. Bartolini, A. Mingozzi, and A. Valletta. An Exact Algorithm for the Period Routing Problem. *Operations Research*, 59:228–241, 2011.

[13] R. Baldacci, A. Mingozzi, and R. Roberti. New route relaxation and pricing strategies for the vehicle routing problem. *Operations Research*, 59:1269–1283, 2011.

[14] R. Baldacci, A. Mingozzi, and R. Roberti. New state-space relaxations for solving the traveling salesman problem with time windows. *INFORMS Journal on Computing*, 2011.

[15] M. L. Balinski and R. E. Quandt. On an Integer Program for a Delivery Problem. *Operations Research*, 12:300–304, 1964.

[16] J. F. Bard, G. Kontoravdis, and G. Yu. A Branch-and-Cut Procedure for the Vehicle Routing Problem with Time Windows. *Transportation Science*, 36:250–269, 2002.

[17] E. J. Beltrami and L. D. Bodin. Networks and vehicle routing for municipal waste collection. *Networks*, 4(1):65–94, 1974.

[18] R. Bent and P. Van Hentenryck. A two-stage hybrid local search for the vehicle routing problem with time windows. *Transportation Science*, 38(4):515–530, 2004.

[19] J. Berger, M. Barkaoui, and O. Bräysy. A route-directed hybrid genetic approach for the vehicle routing problem with time windows. *INFOR*, 41:179–194, 2003.

[20] A. A. Bertossi, P. Carraresi, and G. Gallo. On some matching problems arising in vehicle scheduling models. *Networks*, 17:271–281, 1987.

[21] D. J. Bertsimas, P. Jaillet, and A. R. Odoni. A priori optimization. *Operations Research*, 38:1019–1033, 1990.

[22] A. Bettinelli, A. Ceselli, and G. Righini. A branch-and-cut-and-price algorithm for the multi-depot heterogeneous vehicle routing problem with time windows. *Tramsportation Research Part C: Emerging Technologies*, 19(5):723–740, 2011.

[23] N. Boland, J. Dethridge, and I. Dumitrescu. Accelerated label setting algorithms for the elementary resource constrained shortest path problem. *Operations Research Letters*, pages 58–68, 2006.

[24] O. Bräysy and W. Dullaert. A Fast Evolutionary Metaheuristic for the Vehicle Routing Problem with Time Windows. *International Journal on Artificial Intelligence Tools*, 12:153–172, 2003.

[25] O. Bräysy and M. Gendreau. Vehicle routing problem with time windows, part ii: Metaheuristics. *Transportation Science*, 39(1):119–139, 2005.

[26] O. Bräysy and M. Gendreau. Vehicle routing problem with time windows, part i: Route construction and local search algorithms. *Transportation Science*, 39(1): 104â118, 2005.

[27] O. Bräysy and M. Gendreau. Vehicle routing problem with time windows, part ii: Metaheuristics. *Transportation Science*, 39(1):119–139, 2005.

[28] P. Chen, H. k. Huang, and X.-Y. Dong. Iterated variable neighborhood descent algorithm for the capacitated vehicle routing problem. *Expert Systems with Applications*, 37(2):1620–1627, 2010.

[29] C. H. Christiansen and J. Lysgaard. A branch-and-price algorithm for the capacitated vehicle routing problem with stochastic demands. *Operations Research Letters*, 35:773–781, 2007.

[30] N. Christofides, A. Mingozzi, and P. Toth. Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Mathematical Programming*, 20:255–282, 1981.

[31] N. Christofides, A. Mingozzi, and P. Toth. State-space relaxation procedures for the computation of bounds to routing problems. *Networks*, 11(2):145–164, 1981.

[32] G. Clarke and J. W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4):568–581, 1964.

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

[34] J.-F. Cordeau, M. Gendreau, and G. Laporte. A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks*, 30(2):105–119, 1997.

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

[36] J. F. Cordeau, G. Laporte, and A. Mercier. Improved tabu search algorithm for the handling of route duration constraints in vehicle routing problems with time windows. *The Journal of the Operational Research Society*, 55(5):542â546, 2004.

[37] J.-F. Cordeau, G. Laporte, M. W. P. Savelsbergh, and D. Vigo. Vehicle routing. In C. Barnhart and G. Laporte, editors, *Transportation*, volume 14 of *Handbooks in Operations Research and Management Science*, pages 367–428. Elsevier, 2007.

[38] T. G. Crainic and M. Toulouse. Parallel strategies for meta-heuristics. In F. Glover and G. A. Kochenberger, editors, *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research & Management Science*, pages 475–513. Springer US, 2003.

[39] T. G. Crainic and M. Toulouse. Explicit and emergent cooperation schemes for search algorithms. In V. Maniezzo, R. Battiti, and J.-P. Watson, editors, *Learning and Intelligent Optimization*, volume 5313 of *Lecture Notes in Computer Science*, pages 95–109. Springer Berlin Heidelberg, 2008.

[40] T. G. Crainic, G. C. Crisan, M. Gendreau, N. Lahrichi, and W. Rei. Multithread integrative cooperative optimization for rich combinatorial problems. In *Proceedings of the 2009 IEEE International Symposium on Parallel & Distributed Processing*, IPDPS '09, pages 1–8. IEEE Computer Society, 2009.

[41] E. Danna and C. Pape. Branch-and-price heuristics: A case study on the vehicle routing problem with time windows. In G. Desaulniers, J. Desrosiers, and M. M. Solomon, editors, *Column Generation*, pages 99–129. Springer, 2005.

[42] G. B. Dantzig and J. H. Ramser. The truck dispatching problem. *Management Science*, 6(1), 1959.

[43] G. B. Dantzig and P. Wolfe. Decomposition Principle for Linear Programs. *Operations Research*, 8(1):101–111, 1960.

[44] I. Dayarian, T.G. Crainic, M. Gendreau, and W. Rei. A column generation approach for a multi-attribute vehicle routing problem. Technical Report CIRRELT-2013-57, Montreal, CIRRELT, 2013.

[45] I. Dayarian, T.G. Crainic, M. Gendreau, and W. Rei. A branch-and-price approach for a multi-period vehicle routing problem. Technical Report CIRRELT-2013-60, Montreal, CIRRELT, 2013.

[46] E. V. Denardo and B. L. Fox. Shortest-route methods: 1. Reaching, pruning, and buckets. *Operations Research*, 27:161–186, 1979.

[47] G. Desaulniers, J. Desrosiers, I. loachim, M. M. Solomon, F. Soumis, and D. Villeneuve. A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. In TeodorGabriel Crainic and Gilbert Laporte, editors, *Fleet Management and Logistics*, pages 57–93. Springer US, 1998.

[48] G. Desaulniers, F. Lessard, and A. Hadjar. Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. *Transportation Science*, 42(3):387–404, August 2008.

[49] M. Desrochers, J. Desrosiers, and M. Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40: 342–354, March 1992.

[50] J. Desrosiers, F. Soumis, and M. Desrochers. Routing with time windows by column generation. *Networks*, 14(4):545–565, 1984.

[51] J. Desrosiers, Y. Dumas, M. M. Solomon, and F. Soumis. Time Constrained Routing and Scheduling. In *Handbooks in Operations Research and Management Science 8: Network Routing, M.O. Ball, T.L. Magnanti, C.L. Monma, G.L. Nemhauser (eds), 35–139, Elsevier Science Publishers*, 1995.

[52] R. G. Dondo and J. Cerdá. A hybrid local improvement algorithm for large-scale multi-depot vehicle routing problems with time windows. *Computers & Chemical Engineering*, 33:513–530, 2009.

[53] M. Dror. Note on the Complexity of the Shortest Path Models for Column Generation in VRPTW. *Operations Research*, 42:977–978, 1994.

[54] M. Dror and P. Trudeau. Stochastic vehicle routing with modified savings algorithm. *European Journal of Operational Research*, (23), 1986.

[55] D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44(3):216–229, 2004.

[56] M. L. Fisher. Optimal solution of vehicle routing problems using minimum k-trees. *Operations Research*, 42:626–642, 1994.

[57] M. L. Fisher, K. O. Jornsten, and O. B. G. Madsen. Vehicle Routing with Time Windows: Two Optimization Algorithms. *Operations Research*, 45:488–492, 1997.

[58] K. Fleszar, I. H. Osman, and K. S. Hindi. A variable neighbourhood search algorithm for the open vehicle routing problem. *European Journal of Operational Research*, 195(3):803–809, 2009.

[59] P. M. Francis, K. R. Smilowitz, and M. Tzur. The period vehicle routing problem and its extensions. In B. Golden, S. Raghavan, and E. Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, volume 43 of *Operations Research/Computer Science Interfaces*, pages 73–102. Springer US, 2008.

[60] C. Gauvin, G. Desaulniers, and M. Gendreau. A branch-cut-and-price algorithm for the vehicle routing problem with stochastic demands. Technical report, GERAD, 2012.

[61] H. Gehring and J. Homberger. A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows. In K. Miettinen, Makela M, and J. Toivanen (eds.), editors, *Proceedings of EUROGEN99–Short Course on Evolutionary Algorithms in Engineering and Computer Science*, pages 57–64, 1999.

[62] S. Gélinas, M. Desrochers, J. Desrosiers, and M. M. Solomon. A new branching strategy for time constrained routing problems with application to backhauling. *Annals of Operations Research*, 61:91–109, 1995. ISSN 0254-5330.

[63] M. Gendreau and C. D. Tarantilis. Solving large-scale vehicle routing problems with time windows: The state-of-the-art. Technical report, CIRRELT, Montreal, 2010.

[64] M. Gendreau, A. Hertz, and G. Laporte. New insertion and postoptimization procedures for the traveling salesman problem. *Operations Research*, 40(6):1086–1094, November 1992.

[65] M. Gendreau, A. Hertz, and G. Laporte. A tabu search heuristic for the vehicle routing problem. *Management Science*, 40(10):1276–1290, 1994.

[66] M. Gendreau, G. Laporte, and R. Séguin. A tabu search heuristic for the vehicle routing problem with stochastic demands and customers. *Operations Research*, 44(3):469–477, 1996.

[67] M. Gendreau, Y. Potvin J, A. Smires, and P. Soriano. Multi-period capacity expansion for a local access telecommunications network. *European Journal of Operational Research*, 172:1051–1066, 2006.

[68] M. Gendreau, J. Y. Potvin, O. Bräysy, G. Hasle, and A. Løkketangen. Metaheuristics for the vehicle routing problem and its extensions: A categorized bibliography. In B. Golden, S. Raghavan, and E. Wasil, editors, *The Vehicle Routing Problem - Latest Advances and New Challenges*. Springer Verlag, Heidelberg, 2008.

[69] F. Glover. Heuristics for integer programming using surrogate constraints. *Decision Sciences*, 8(1):156–166, 1977.

[70] F. Glover. Tabu search - part I. *INFORMS Journal on Computing*, 1(3):190–206, 1989.

[71] F. Glover. Tabu search - part II. *INFORMS Journal on Computing*, 2(1):4–32, 1990.

[72] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, Norwell, MA, USA, 1997. ISBN 079239965X.

[73] F. Glover and C. McMillan. The general employee scheduling problem. an integration of {MS} and {AI}. *Computers & Operations Research*, 13(5):563 – 573, 1986.

[74] B. Golden, S. Raghavan, and E. A. Wasil. *The vehicle routing problem : latest advances and new challenges*. Operations research/Computer science interfaces series, 43. Springer, 2008.

[75] B. L. Golden, A. A. Assad, and E. A. Wasil. Routing Vehicles in the Real World: Applications in the Solid Waste ,Beverage, Food, Dairy, and Newspaper Industries. In Paolo Toth and Daniele Vigo, editors, *The Vehicle Routing Problem*. Society for Industrial and Applied Mathematics, 2002.

[76] Bruce L. Golden, EdwardA. Wasil, JamesP. Kelly, and I-Ming Chao. The impact of metaheuristics on solving the vehicle routing problem: Algorithms, problem sets, and computational results. In G. Crainic, T. G.and Laporte, editor, *Fleet Management and Logistics*, pages 33–56. Springer US, 1998.

[77] C. Groër, B. Golden, and E. Wasil. The consistent vehicle routing problem. *Manufacturing & Service Operations Management*, 11(4):630–643, 2009.

[78] P. Hansen, N. Mladenovic, and José A. Moreno-Pérez. Variable neighbourhood search: methods and applications. *Annals of Operations Research*, 175(1):367–407, 2010.

[79] R. F. Hartl, G. Hasle, and G. K. Janssens. Special issue on rich vehicle routing problems. *Central European Journal of Operations Research*, 14(2):103–104, 2006.

[80] V. C. Hemmelmayr, K. F. Doerner, and R. F. Hartl. A variable neighborhood search heuristic for periodic routing problems. *European Journal of Operational Research*, 195(3):791–802, 2009.

[81] S. C. Ho and M. Gendreau. Path relinking for the vehicle routing problem. *Journal of Heuristics*, 12(1-2):55–72, 2006.

[82] J. H. Holland. Outline for a logical theory of adaptive systems. *Journal of the ACM*, 9(3):297–314, 1962.

[83] J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, 1975.

[84] J. H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, Cambridge, MA, USA, 1992. ISBN 0262082136.

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

[86] S. Irnich and G. Desaulniers. *Shortest Path Problems with Resource Constraints*, chapter 2, pages 33–65. GERAD 25th Anniversary Series. Springer, 2005.

[87] S. Irnich and D. Villeneuve. The shortest path problem with resource constraints and k-cycle elimination for $k >= 3$. *INFORMS Journal on Computing*, 18(3): 391–406, 2006.

[88] O. Jabali, W. Rei, M. Gendreau, and G. Laporte. New valid inequalities for the multi-vehicle routing problem with stochastic demands. Technical report, CIRRELT-2012-58, Montreal, 2012.

[89] P. Jaillet. A priori solution of a traveling salesman problem in which a random subset of the customers are visited. *Operations Research*, 36:929–936, 1988.

[90] A. Juan, J. Faulin, S. Grasman, D. Riera, J. Marull, and C. Mendez. Using safety stocks and simulation to solve the vehicle routing problem with stochastic demands. *Transportation Research Part C - Emerging Technologies*, 19:751–765, 2011.

[91] B. Kallehauge, J. Larsen, and O. B.G. Madsen. Lagrangian duality applied to the vehicle routing problem with time windows. *Computers & Operations Research*, 33(5):1464 – 1487, 2006.

[92] J. N. Kennedy and R. C. Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948. IEEE, 1995.

[93] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[94] N. Kohl and O. B. G. Madsen. An optimization algorithm for the vehicle routing problem with time windows based on lagrangian relaxation. *Operations Research*, 45(3):395–406, 1997.

[95] N. Kohl, J. Desrosiers, O.B. G. Madsen, . M. Solomon, and F. Soumis. 2-path cuts for the vehicle routing problem with time windows. *Transportation Science*, pages 101–116, 1999.

[96] A. W. J. Kolen, A. H. G. Rinnooy Kan, and H. W. J. M. Trienekens. Vehicle Routing with Time Windows. *Operations Research*, 35:266–273, 1987.

[97] G. Kontoravdis and J. F. Bard. A GRASP for the Vehicle Routing Problem with Time Windows. *Informs Journal on Computing*, 7:10–23, 1995.

[98] R. Kouassi, M. Gendreau, J.-Y. Potvin, and P. Soriano. Heuristics for multi-period capacity expansion in local telecommunications networks. *Journal of Heuristics*, 15(4):381–402, 2009.

[99] J. Kytöjoki, T. Nuortio, O. Bräysy, and M. Gendreau. An efficient variable neighborhood search heuristic for very large scale vehicle routing problems. *Computers & Operations Research*, 34(9):2743–2757, 2007.

[100] N. Lahrichi, T. G. Crainic, M. Gendreau, W. Rei, G. C. Crişan, and T. Vidal. An integrative cooperative search framework for multi-decision-attribute combinatorial optimization. Technical report, CIRRELT, Montreal, 2012.

[101] N. Lahrichi, T.G. Crainic, M. Gendreau, W. Rei, and L-M. Rousseau. Strategic analysis of the dairy transportation problem. Technical Report CIRRELT-2012-80, Montreal, Montreal, Forthcoming in *Journal of Operational Research Society*, CIRRELT, 2012.

[102] G. Laporte. Fifty years of vehicle routing. *Transportation Science*, 43(4):408–416, 2009.

[103] G Laporte and F. V. Louveaux. The integer l-shaped method for stochastic integer programs with complete recourse. *Operations Research Letters*, 13:133–142, 1993.

[104] G. Laporte, Y. Nobert, and D. Arpin. Optimal solutions to capacitated multi-depot vehicle routing problems. *Congress*, (44):283–292, 1984.

[105] G. Laporte, Y. Nobert, and S. Taillefer. Solving a Family of Multi-Depot Vehicle Routing and Location-Routing Problems. *Transportation Science*, 22:161–172, 1988.

[106] H. C. W. Lau, T. M. Chan, W. T. Tsui, and W. K. Pang. Application of Genetic Algorithms to Solve the Multidepot Vehicle Routing Problem. *IEEE Transactions Automation Science and Engineering*, 7:383–392, 2010.

[107] J. K. Lenstra and A. H. G. Rinnooy Kan. Complexity of vehicle routing and scheduling problems. *Networks*, 11:221–227, 1981.

[108] J. T. Linderoth and M. W. P. Savelsbergh. A computational study of branch and bound search strategies for mixed integer programming. *INFORMS Journal on Computing*, 1999.

[109] M. E. Lübbecke and J. Desrosiers. Selected topics in column generation. *Operations Research*, 53:1007–1023, November 2005. ISSN 0030-364X.

[110] Desrochers M. An algorithm for the shortest path problem with resource constraints. Technical Report G-88-27, GERAD, 1988.

[111] R. Martinelli. *Exact Algorithms for Arc and Node Routing Problems*. PhD thesis, Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, Brazil, 2012.

[112] A. Mingozzi. The multi-depot periodic vehicle routing problem. In J.-D. Zucker and L. Saitta, editors, *Abstraction, Reformulation and Approximation*, volume 3607 of *Lecture Notes in Computer Science*, pages 901–901. Springer Berlin / Heidelberg, 2005. ISBN 978-3-540-27872-6.

[113] N. Mladenovic and P. Hansen. Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100, 1997.

[114] Y. Nagata and O. Bräysy. Edge assembly-based memetic algorithm for the capacitated vehicle routing problem. *Networks*, 54(4):205–215, 2009.

[115] Y. Nagata, O. Bräysy, and W. Dullaert. A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows. *Computers & Operations Research*, 37(4):724 – 737, 2010.

[116] B. Ombuki-Berman and F. T. Hanshar. Using genetic algorithms for multi-depot vehicle routing. In F. B. Pereira and J. Tavares, editors, *Bio-inspired Algorithms for the Vehicle Routing Problem*, volume 161 of *Studies in Computational Intelligence*, pages 77–99. Springer Berlin Heidelberg, 2009. ISBN 978-3-540-85151-6.

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

[118] A.-S. Pepin, G. Desaulniers, A. Hertz, and D. Huisman. A comparison of five heuristics for the multiple depot vehicle scheduling problem. *Journal of Scheduling*, 12(1):17–30, 2009.

[119] G. Pesant and M. Gendreau. *A View of Local Search in Constraint Programming*. Springer-Verlag, 1996.

[120] S. Pirkwieser and G. R. Raidl. Multilevel variable neighborhood search for periodic routing problems. In P. Cowling and P. Merz, editors, *Evolutionary Computation in Combinatorial Optimization*, volume 6022 of *Lecture Notes in Computer Science*, pages 226–238. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-12138-8.

[121] D. Pisinger and S. Ropke. A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34:2403–2435, 2007.

[122] D. Pisinger and S. Ropke. Large Neighborhood Search. In M. Gendreau and J.-Y. Potvin, editors, *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research & Management Science*, chapter 13, pages 399–419. Springer US, Boston, MA, 2010.

[123] J.-Y. Potvin. State-of-the art review - evolutionary algorithms for vehicle routing. *INFORMS Journal on Computing*, 21(4):518–548, 2009.

[124] J.-Y. Potvin, T. Kervahut, B. l. Garcia, and J.-M. Rousseau. The Vehicle Routing Problem with Time Windows Part I: Tabu Search. *Informs Journal on Computing*, 8:158–164, 1996.

[125] C. Prins. A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research*, 31(12):1985 – 2002, 2004.

[126] A. Rahimi-Vahed, Teodor Gabriel Crainic, Michel Gendreau, and Walter Rei. A path relinking algorithm for a multi-depot periodic vehicle routing problem. *Journal of Heuristics*, 19(3):497–524, 2013.

[127] I. Rechenberg. Cybernetic solution path of an experimental problem. Technical report, Royal Air Force Establishment, 1965.

[128] I. Rechenberg. *Evolutionsstrategie: optimierung technischer systeme nach prinzipien der biologischen evolution*. Frommann-Holzboog, 1973.

[129] C. Rego and C. Roucairol. A parallel tabu search algorithm using ejection chains for the vehicle routing problem. In I. H. Osman and J. P. Kelly, editors, *Meta-Heuristics*, pages 661–675. Springer US, 1996.

[130] J. Renaud, G. Laporte, and F. F. Boctor. A tabu search heuristic for the multi-depot vehicle routing problem. *Computers & Operations Research*, 23:229–235, 1996.

[131] M.G.C. Resende, C.C. Ribeiro, F. Glover, and R. Martí. Scatter search and path-relinking: Fundamentals, advances, and applications. In M. Gendreau and J.-Y. Potvin, editors, *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research & Management Science*, pages 87–107. Springer US, 2010.

[132] G. Righini and M. Salani. Dynamic programming algorithms for the elementary shortest path problem with resource constraints. *Electronic Notes in Discrete Mathematics*, 17:247 – 249, 2004.

[133] G. Righini and M. Salani. Decremental state space relaxation strategies and initialization heuristics for solving the orienteering problem with time windows with dynamic programming. *Computers & Operations Research*, 36(4):1191–1203, 2009.

[134] S. Ropke and D. Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Trasportation Science*, 40 (4):455–472, 2006.

[135] S. Ropke and D. Pisinger. A unified heuristic for a large class of vehicle routing problems with backhauls. *European Journal of Operational Research*, 171(3): 750–775, 2006.

[136] L. M. Rousseau, M. Gendreau, and G. Pesant. Using Constraint-Based Operators to Solve the Vehicle Routing Problem with Time Windows. *Journal of Heuristics*, 8:43–58, 2002.

[137] M.W.P. Savelsbergh. Local search in routing problems with time windows. *Annals of Operations Research*, 4(1):285–305, 1985.

[138] G. Schrimpf. Record breaking optimization results using the ruin and recreate principle. *Journal of Computational Physics*, 159(2):139–171, 2000.

[139] N. Secomandi and F. Margot. Reoptimization approaches for the vehicle-routing problem with stochastic demands. *Operations Research*, 57:214–230, 2009.

[140] R. Séguin. *Problèmes stochastiques de tournées de véhicules*. PhD thesis, Université de Montréal, Canada, Centre de Recherche sur le Transport, 1994.

[141] F. Semet and E. Taillard. Solving real-life vehicle routing problems efficiently using tabu search. *Annals of Operations Research*, 41:469–488, 1993.

[142] P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In M. Maher and J.-F. Puget, editors, *Principles and Practice of Constraint Programming – CP98*, volume 1520 of *Lecture Notes in Computer Science*, pages 417–431. Springer Berlin Heidelberg, 1998.

[143] Y. Shi and R. Eberhart. A modified particle swarm optimizer. In *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, pages 69–73, 1998.

[144] R. M. Van Slyke and R. Wets. *L*Shaped Linear Programs with Applications to Optimal Control and Stochastic Programming. *Siam Journal on Applied Mathematics*, 17, 1969.

[145] M. M. Solomon. On the worst-case performance of some heuristics for the vehicle routing and scheduling problem with time window constraints. *Networks*, 16(2): 161–174, 1986.

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

[147] El-Ghazali T. *Metaheuristics - From Design to Implementation*. Wiley, 2009. ISBN 978-0-470-27858-1.

[148] C. D. Tarantilis, F. Stavropoulou, and P. P. Repoussis. A template-based tabu search algorithm for the consistent vehicle routing problem. *Expert Systems with Applications*, 39(4):4233–4239, 2012.

[149] S. Thangiah and S. Salhi. Genetic clustering: An adaptive heuristic for the multi depot vehicle routing problem. *Applied Artificial Intelligence*, 15(4):361–383, 2001.

[150] F. A. Tillman. The multiple terminal delivery problem with probabilistic demands. *Transportation Science*, 3:192–204, 1969.

[151] P. Toth and D. Vigo. *The vehicle routing problem*, volume 9. Society for Industrial and Applied Mathematics, 2002.

[152] P. Toth and D. Vigo. The Granular Tabu Search and Its Application to the Vehicle-Routing Problem. *Informs Journal on Computing*, 15:333–346, 2003.

[153] F. Tricoire. *Optimization des tournees de vehicules et de personnels de maintenance: application a la distribution et au traitement des eaux*. PhD thesis, IRCCyN - Institut de Recherche en Communications et en Cybernétique de Nantes, Nantes, France, 2006.

[154] V. Černý. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45(1):41–51, 1985.

[155] T. Vidal, T. G. Crainic, M. Gendreau, N. Lahrichi, and W. Rei. A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. *Operations Research*, 60(3):611–624, 2012.

[156] T. Vidal, T. G. Crainic, M. Gendreau, and C. Prins. Heuristics for multi-attribute vehicle routing problems: A survey and synthesis. *European Journal of Operational Research*, 231(1):1–21, 2013.

[157] T. Vidal, T. G. Crainic, M. Gendreau, and C. Prins. A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers & Operations Research*, 40(1):475 – 489, 2013.

[158] M. Wen, J.-F. Cordeau, G. Laporte, and J. Larsen. The dynamic multi-period vehicle routing problem. *Computers & Operations Research*, 37(9):1615–1623, 2010.

[159] J. Xu and J. P. Kelly. A network flow-based tabu search heuristic for the vehicle routing problem. *Transportation Science*, 30(4):379–393, 1996.

[160] W. Yang, K. Mathur, and R. H. Ballou. Stochastic vehicle routing problem with restocking. *Transportation Science*, 34:99–112, 2000.