CIRRELT

Centre interuniversitaire de recherche
sur les réseaux d'entreprise, la logistique et le transport

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

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

Iman Dayarian
Teodor Gabriel Crainic
Michel Gendreau
Walter Rei

October 2013

CIRRELT-2013-60

Bureaux de Montréal :
Université de Montréal
C.P. 6128, succ. Centre-ville
Montréal (Québec)
Canada H3C 3J7
Téléphone : 514 343-7575
Télécopie : 514 343-7121

Bureaux de Québec :
Université Laval
2325, de la Terrasse, bureau 2642
Québec (Québec)
Canada G1V 0A6
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

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

Iman Dayarian[1,2,*], Teodor Gabriel Crainic[1,3], Michel Gendreau[1,4], Walter Rei[1,3]

[1] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT)

[2] Université de Montréal, Pavillon André-Aisenstadt, Department of Computer Science and Operations Research, P.O. Box 6128, Station Centre-Ville, Montréal, Canada H3C 3J7

[3] Department of Management and Technology, Université du Québec à Montréal, P.O. Box 8888, Station Centre-Ville, Montréal, Canada H3C 3P8

[4] Department of Mathematics and Industrial Engineering, École Polytechnique de Montréal, P.O. Box 6079, Station Centre-ville, Montréal, Canada H3C 3A7

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-attribute vehicle routing problem, heterogeneous fleet, multiple depots, branch-and-price.

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 (FRQNT) 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.

---

* Corresponding author: Iman.Dayarian@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2013

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 Dantzig and Ramser (1959). 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 (Toth and Vigo, 2002).

Several variants and extensions of the classical VRP have been introduced, and they are supported by a well-developed literature (Cordeau et al., 2007). 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 by a given vehicle. Moreover, in the dairy industry, because of contractual arrangements, a single plan is the basis of the negotiations between the stakeholders involved in each contract over a horizon.

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 formula-

tion 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 2, we describe the problem, and Section 3 presents the proposed two-stage formulation. The algorithm is presented in Section 4, and its components are described in Sections 5 to 7. The experimental results are reported in Section 8, and Section 9 provides concluding remarks.

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. (2012) and Dayarian et al. (2013). 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. 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. 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 also 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 in each execution of the plan.

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 additional travel costs are later reimbursed by the FPLQ.

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 (Lenstra and Kan, 1981).

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 (daily) 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 often 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 geographi-

cal 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 to 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 horizon length.

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

3. Model

In this section, we present a model for the problem introduced in Section 2. The formulation takes into account the routing characteristics and specifications and the variations in the supply over a given horizon T.

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 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 1(b) shows the normalized distribution of the periods.

Our model is based on the *a priori* optimization framework (Bertsimas et al., 1990; Jaillet, 1988), originally introduced for stochastic problems. This multi-stage framework assumes that a plan is designed in the first stage. New

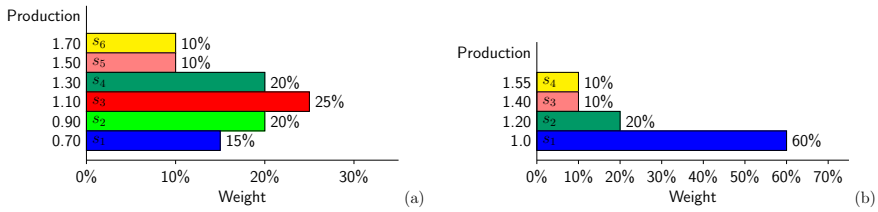

Figure 1: (a) Example of period distribution.
(b) Normalized distribution of periods based on reference period.

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 (Bertsimas et al., 1990). 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.

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

$$q_j^s = P_s . q_j^{ref}, \qquad j \in \mathcal{N} s \in \mathcal{S}. \tag{1}$$

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. 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:

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

and

$$\sum_{j \in r} q_j^{ref} \leq Q. \tag{3}$$

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. 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 delivered to plant p 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. Each route is performed using a vehicle $k \in \mathcal{K}$, where $\mathcal{K}$ represents an unlimited homogeneous fleet of vehicles. Associated with each vehicle 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 routing cost (first stage) 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:

$$\min \quad \sum_{r \in \mathcal{R}} (c_r + c_k) y_r + \mathcal{F}(x) \tag{4}$$

subject to

$$\sum_{r \in \mathcal{R}} a_{ir} y_r = 1 \quad (i \in \mathcal{N}); \tag{5}$$

$$\sum_{r \in \mathcal{R}} l_{pr}^{ref} y_r \geq D_p^{ref} \quad (p \in \mathcal{P}); \tag{6}$$

$$y_r \in \{1, 0\} \quad (r \in \mathcal{R}). \tag{7}$$

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)–(7). In fact, $\mathcal{F}(x)$ represents the value of the second-stage problem given a first-stage solution x. Constraint (5) ensures that each producer is visited exactly once by exactly one route, and constraint (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)

$$\mathcal{F}(x) = \sum_{s \in \mathcal{S}} Pr_s F(x, q^s) \tag{8}$$

where

$$F(x, q^s) = \min \quad \sum_{k \in \mathcal{K}} \sum_{i \in \mathcal{N}} 2c_{ip_k} w^s_{ik} \tag{9}$$

subject to

$$z^s_{ijk} \leq Q x_{ijk} \qquad (i, j \in \mathcal{V}, k \in \mathcal{K}, s \in \mathcal{S}), \tag{10}$$

$$w^s_{ik} \leq \sum_{j \in \mathcal{N}} x_{ijk} \qquad (i \in \mathcal{V}, k \in \mathcal{K}, s \in \mathcal{S}), \tag{11}$$

$$\sum_{j \in \mathcal{N} \cup \mathcal{U}} z^s_{ijk} = \sum_{j \in \mathcal{N} \cup \mathcal{D}} z^s_{jik} + q^s_i - Q w^s_{ik} \quad (i \in \mathcal{N}, k \in \mathcal{K}, s \in \mathcal{S}), \tag{12}$$

$$\sum_{j \in \mathcal{N}} z^s_{djk} = 0 \qquad (d \in \mathcal{D}, k \in \mathcal{K}, s \in \mathcal{S}), \tag{13}$$

$$z^s \geq 0 \qquad (s \in \mathcal{S}), \tag{14}$$

$$w^s_{ik} \in \{0, 1\} \qquad (i \in \mathcal{N}, k \in \mathcal{K}, s \in \mathcal{S}). \tag{15}$$

Constraint (10) shows that the flows are nonzero only on planned routes and do not exceed the vehicle capacity. Constraint (11) specifies that a failure on producer i on route k can occur only if i belongs to route k. Constraint (12) defines when a failure occurs on a given producer i. Constraint (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 adapted for stochastic problems with a similar structure.

4. Solution Approach

Since the cardinality of $\mathcal{R}$ is extremely large, the approach used to solve (4)–(7) is based on a column generation 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 Feillet et al., 2004; Irnich and Desaulniers, 2005). 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.

4.1. Literature review

In this section, we review research into different VRPs. This will allow us to take advantage of recent advances in other VRPs that are similar to our

model. The most closely related problem is the consistent vehicle routing problem (ConVRP) proposed by Groër et al. (2009). 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 (Tarantilis et al., 2012).

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 Tricoire, 2006; Angelelli et al., 2007; Wen et al., 2010; Athanasopoulos, 2011). 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. (2008). The best-known algorithms for the PVRP are those of Cordeau et al. (1997), Hemmelmayr et al. (2009), and Vidal et al. (2012).

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 single-period VRPs.

4.1.1. Vehicle routing problem with stochastic demands

The first study of the VRPSD was carried out by Tillman (1969) for the multi-depot problem; the solution method was based on the savings heuristic. Dror and Trudeau (1986) 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 (1994) and Gendreau et al. (1996) 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 (1993), which itself is an extension of the integer L-shaped method of Van Slyke and Wets (1969). More recently, Jabali et al. (2012) 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 (2007) 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. (2012) 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 (Yang et al., 2000; Secomandi and Margot, 2009; Ak and Erera, 2007; Juan et al., 2011).

In the next section, we describe the state of the art of solution methodologies for the single-period ESPPRC.

4.1.2. Elementary shortest path problem with resource constraints

Many recent advances in branch-and-price for the classical 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 difficulty to the shortest path problem with resource constraints (SPPRC), itself an NP-hard problem. The most promising methodologies for the 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. (2006) and Righini and Salani (2009) 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.

Based on the second strategy, Baldacci et al. (2011) introduced a new state-space relaxation, called ng-path relaxation, 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 (2012) 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 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 period-based formulation of the problem.

5. Master Problem

If we relax constraints (5), the set partitioning formulation is transformed into a set covering formulation. Moreover, a relaxation on constraints (7), provides a linear relaxation of the model. The master problem becomes

(MP)

$$\min \quad \sum_{r \in \mathcal{R}'} (c_r + c_k) y_r + \mathcal{F}(x) \tag{16}$$

subject to

$$\sum_{r\in\mathcal{R}'} a_{ir}y_r \geq 1 \quad (i \in \mathcal{N}); \tag{17}$$

$$\sum_{r\in\mathcal{R}'} l_{pr}^{ref}y_r \geq D_p^{ref} \quad (p \in \mathcal{P}); \tag{18}$$

$$0 \leq y_r \leq 1 \quad (r \in \mathcal{R}') \tag{19}$$

where constraint (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.

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 elementary shortest path problem with resource constraints (MPESPPRC). Consider the following dual variables:
λ_i: nonnegative dual variable of (17) for producer $i \in \mathcal{N}$;
μ_p: nonnegative dual variable of (18) for plant $p \in \mathcal{P}$.

In each subproblem, the objective function is given by

$$\min \quad Z = \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}}(c_{ij} - \lambda_i)x_{ij} - \sum_{p\in\mathcal{P}} l_{pr}^{ref}\mu_p + c_k + \mathcal{F}(x) \tag{20}$$

where Z represents the reduced cost of the generated column. Since the ESPPRC is an NP-hard problem in the strong sense (Irnich and Desaulniers, 2005), 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. (2013) 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 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.

6.1. Tabu Search

TS (see Glover and Laguna, 1997; Bräysy and Gendreau, 2005) 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 Desaulniers et al., 2008). 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, one possibility is to 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, a 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 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 (3), and then their expected costs and reduced costs, with respect to the different periods, are calculated. 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.

6.2. Exact dynamic programming

As mentioned in Section 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 $x \nleq m$ 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 Irnich and Desaulniers, 2005). We will now describe a special phenomenon that occurs when extending labels using the above extension functions in the case of the MPESPPRC.

6.2.1. 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 fo e.g. Feillet et al., 2004). 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 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: $L_1 = L_2$;

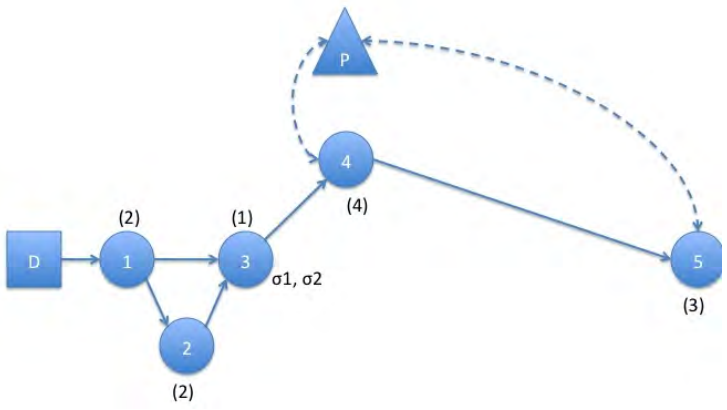

Figure 2: Example 1

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 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.

6.2.2. 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.

6.2.3. Matrix-based implementation

As mentioned in Section 6.2.1, 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 (1979), 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 6.2.1). 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.

6.2.4. 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. Elementarity is insured with DSSR but not with ng-route relaxation.

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

$$\Pi(r) = \{i_j \in \mathcal{V}_r | i_j \in \bigcap_{k=j+1}^{n} \mathcal{N}_{i_k}, j = 1, \ldots, n-1\} \cup \{i_n\}. \tag{21}$$

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).

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 single-period 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.

6.4. Skeleton of the multi-phase subproblem algorithm

Algorithm 1 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 1, 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 1 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
```

7. Strong Branching

The column generation does not guarantee the integrality of the solution. Therefore, as described in Section 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. (2013) 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 (1999):

$$score = \gamma \max(\Delta^R, \Delta^L) + (1 - \gamma) \min(\Delta^R, \Delta^L), \tag{22}$$

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

$$j = \arg\max\{\gamma \max(\Delta^R, \Delta^L) + (1 - \gamma) \min(\Delta^R, \Delta^L)\}. \tag{23}$$

8. Computational Results

To assess the performance of our algorithm, we carried out a computational study of test problems with different characteristics. Section 8.1 explains the characteristics of these problems, and Section 8.2 discusses the contribution of the different column generators. The results from the branch-and-price procedure are discussed in Section 8.3. In Section 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.

8.1. Benchmark

We generated the instances using the generator of Dayarian et al. (2013) with some modifications to adapt it to the structure of our problem. As shown in Table 1, Dayarian et al. (2013) 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 1: Four problem classes generated by Dayarian et al. (2013)

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 supply 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 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.

8.2. Linear relaxation

To evaluate the TS column generator, we ran a sample of the instances with and without it. Table 3 gives the value of the different parameters used; these values were obtained through a series of trial-and-error tests.

Table 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

Table 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

Table 3: Parameter values used in tabu search

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

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 Dayarian et al., 2013), where the HDP and EDP were more efficient in the absence of TS. This indicates the extra difficulty of the multi-period variant.

8.3. Integer solution

For each group of five instances of the same size, Tables 5 and 6 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 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

$$\text{optimality gap} = \frac{\text{(best upper bound - best lower bound)}}{\text{best lower bound}}. \qquad (24)$$

Table 5 gives the results for the instances with fifteen producers, and Table 6 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 6: 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.

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 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

$$c_f(x_{MPS}) + c(x_{MPS}) + \mathcal{F}(x_{MPS}) \leq c_f(x) + c(x) + \mathcal{F}(x). \tag{25}$$

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:

$$c(x_{RP}) \leq c(x_{MPS}). \tag{26}$$

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 (25)). We also have

$$c(x_{RP}) \leq c(x_{MPS}) \leq c(x_{WCS}), \tag{27}$$

$$\mathcal{F}(x_{WCS}) \leq \mathcal{F}(x_{MPS}) \leq \mathcal{F}(x_{RP}). \tag{28}$$

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 8.2 using WCS and RP. The routes obtained were then costed assuming different periods with different production levels and weights. Table 7 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 7:

$$gap_{(WCS)} = \frac{[c_f(x_{WCS}) + c(x_{WCS}) + \mathcal{F}(x_{WCS})] - [c_f(x_{MPS}) + c(x_{MPS}) + \mathcal{F}(x_{MPS})]}{c_f(x_{MPS}) + c(x_{MPS}) + \mathcal{F}(x_{MPS})}, \tag{29}$$

$$gap_{(RP)} = \frac{[c_f(x_{RP}) + c(x_{RP}) + \mathcal{F}(x_{RP})] - [c_f(x_{MPS}) + c(x_{MPS}) + \mathcal{F}(x_{MPS})]}{c_f(x_{MPS}) + c(x_{MPS}) + \mathcal{F}(x_{MPS})}. \tag{30}$$

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

$$\text{Recourse gap (RP)} = \frac{\mathcal{F}(x_{RP}) - \mathcal{F}(x_{MPS})}{\mathcal{F}(x_{MPS})}. \tag{31}$$

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.

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: 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 5: Computational results for instances with 15 producers

# producers	size comb.	# periods	period type	# opt. sol.	# int. sol.	CPU1	CPU2	opt. gap %
15	2D2P	4	T1	4	1	3643.8	54.8	18.5
			T2	4	1	3652.3	65.4	18.2
			T3	4	1	3651.4	64.3	15.0
			T4	4	1	3685.6	107	14.6
			T5	4	1	3681.2	101.5	15.3
		5	T1	4	1	3659.4	74.3	19.9
			T2	4	1	3662.8	78.5	18.3
			T3	4	1	3651.6	64.5	15.9
			T4	4	1	3669.1	86.4	16.1
			T5	4	1	3697.7	122.1	16.3
	2D3P	4	T1	5	0	1106.4	1106.4	0.0
			T2	5	0	964.1	964.1	0.0
			T3	5	0	837.8	837.8	0.0
			T4	5	0	804.1	804.1	0.0
			T5	5	0	768.6	768.6	0.0
		5	T1	5	0	1042.8	1042.8	0.0
			T2	5	0	1061.4	1061.4	0.0
			T3	5	0	883.5	883.5	0.0
			T4	5	0	833	833	0.0
			T5	5	0	755.4	755.4	0.0
	3D2P	4	T1	3	2	7291.1	151.9	24.0
			T2	3	2	7454.5	424.1	24.8
			T3	3	2	7273.7	122.9	22.5
			T4	3	2	7279.1	131.8	22.2
			T5	3	1	7286	143.4	24.2
		5	T1	3	2	7268.4	114.1	24.7
			T2	3	2	7295.8	159.7	24.1
			T3	3	2	7343.5	239.1	22.7
			T4	3	2	7272.1	120.2	26.0
			T5	3	2	7282.4	137.3	23.5
	3D3P	4	T1	5	0	1347	1347	0.0
			T2	5	0	1354.9	1354.9	0.0
			T3	5	0	1178.4	1178.4	0.0
			T4	5	0	1090.4	1090.4	0.0
			T5	5	0	1169.1	1169.1	0.0
		5	T1	5	0	1407.5	1407.5	0.0
			T2	5	0	1345.2	1345.2	0.0
			T3	5	0	1293.5	1293.5	0.0
			T4	5	0	1141.7	1141.7	0.0
			T5	5	0	1121.9	1121.9	0.0

Table 6: Computational results for instances with 20 producers

# producers	size comb.	# periods	period 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	4	1	10021.6	8027.1	3.3
			T2	4	1	11460.7	9825.9	4.2
			T3	4	0	10354.1	8442.7	∞
			T4	5	0	7588.5	7588.5	0.0
			T5	4	1	9846.4	7808	1.2
		5	T1	3	2	11450.2	7083.6	4.1
			T2	4	0	10679.6	8849.5	∞
			T3	4	1	8979.1	6723.9	4.6
			T4	4	1	7878.3	5347.9	2.0
			T5	5	0	6970.8	6970.8	0.0
	3D2P	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}	∞
	3D3P	4	T1	3	0	11379.9	6966.5	∞
			T2	3	0	10567.9	5613.2	∞
			T3	4	0	9434.3	7292.9	∞
			T4	4	0	9732	7665	∞
			T5	4	0	8519.2	6149	∞
		5	T1	3	1	12369.4	8615.6	7.2
			T2	3	0	11596.2	7327.1	∞
			T3	4	0	9889.4	7861.8	∞
			T4	4	0	10004.5	8005.6	∞
			T5	4	0	9295.1	7118.9	∞

Table 7: 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**

References

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.

E. Angelelli, M. Grazia Speranza, and M. W. P. Savelsbergh. Competitive analysis for dynamic multiperiod uncapacitated routing problems. *Netw.*, 49(4):308–317, 2007.

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.

R. Baldacci, A. Mingozzi, and R. Roberti. New route relaxation and pricing strategies for the vehicle routing problem. *Oper. Res.*, 59:1269–1283, 2011.

D. J. Bertsimas, P. Jaillet, and A. R. Odoni. A priori optimization. *Oper. Res.*, 38:1019–1033, 1990.

N. Boland, J. Dethridge, and I. Dumitrescu. Accelerated label setting algorithms for the elementary resource constrained shortest path problem. *Oper. Res. Lett.*, 34(1):58–68, 2006.

O. Bräysy and M. Gendreau. Vehicle routing problem with time windows, part II: Metaheuristics. *Transportation Science*, 39(1):119–139, 2005.

C. H. Christiansen and J. Lysgaard. A branch-and-price algorithm for the capacitated vehicle routing problem with stochastic demands. *Oper. Res. Lett.*, 35:773–781, 2007.

J.-F. Cordeau, M. Gendreau, and G. Laporte. A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Netw.*, 30(2):105–119, 1997.

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. 2007.

G. B. Dantzig and J. H. Ramser. The truck dispatching problem. *Management Science*, 6(1):80–91, 1959.

I. Dayarian, T.G. Crainic, M. Gendreau, and W. Rei. A column generation approach for a multi-attribute vehicle routing problem. Technical report, CIRRELT, Montreal, 2013.

E. V. Denardo and B. L. Fox. Shortest-route methods: 1. Reaching, pruning, and buckets. *Oper. Res.*, 27:161–186, 1979.

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, 2008.

M. Dror and P. Trudeau. Stochastic vehicle routing with modified savings algorithm. *Eur. J. Oper. Res.*, 23:228–235, 1986.

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. *Netw.*, 44(3):216–229, 2004.

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. 2008.

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.

M. Gendreau, G. Laporte, and R. Séguin. A tabu search heuristic for the vehicle routing problem with stochastic demands and customers. *Oper. Res.*, 44(3):469–477, 1996.

F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, Norwell, MA, USA, 1997. ISBN 079239965X.

C. Groër, B. Golden, and E. Wasil. The consistent vehicle routing problem. *Manufacturing & Service Operations Management*, 11(4):630–643, 2009.

V. C. Hemmelmayr, K. F. Doerner, and R. F. Hartl. A variable neighborhood search heuristic for periodic routing problems. *Eur. J. Oper. Res.*, 195(3): 791–802, 2009.

S. Irnich and G. Desaulniers. *Shortest Path Problems with Resource Constraints*, chapter 2, pages 33–65. GERAD 25th Anniversary Series. Springer, 2005.

O. Jabali, W. Rei, M. Gendreau, and G. Laporte. New valid inequalities for the multi-vehicle routing problem with stochastic demands. Technical report, CIRRELT, Montreal, 2012.

P. Jaillet. A priori solution of a traveling salesman problem in which a random subset of the customers are visited. *Oper. Res.*, 36:929–936, 1988.

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.

N. Lahrichi, T. G. Crainic, M. Gendreau, W. Rei, and L.-M. Rousseau. Strategic analysis of the dairy transportation problem. Technical report, CIRRELT, Montreal, 2012.

G. Laporte and F. V. Louveaux. The integer L-shaped method for stochastic integer programs with complete recourse. *Oper. Res. Lett.*, 13:133–142, 1993.

J. K. Lenstra and A. H. G. R. Kan. Complexity of vehicle routing and scheduling problems. *Netw.*, 11:221–227, 1981.

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.

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.

G. Righini and M. Salani. Decremental state space relaxation strategies and initialization heuristics for solving the orienteering problem with time windows with dynamic programming. *Comput. Oper. Res.*, 36(4):1191–1203, 2009.

N. Secomandi and F. Margot. Reoptimization approaches for the vehicle-routing problem with stochastic demands. *Oper. Res.*, 57:214–230, 2009.

R. Séguin. *Problèmes stochastiques de tournées de véhicules*. PhD thesis, Université de Montréal, Canada, Centre de Recherche sur le Transport, 1994.

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.

F. A. Tillman. The multiple terminal delivery problem with probabilistic demands. *Transportation Science*, 3:192–204, 1969.

P. Toth and D. Vigo. *The Vehicle Routing Problem*, volume 9. Society for Industrial and Applied Mathematics, 2002.

F. Tricoire. *Optimization des tournees de vehicules et de personnels de maintenance: Application a la distribution et au traitement des eaux*. PhD thesis, University of Nantes, Nantes, France, 2006.

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.

T. Vidal, T. G. Crainic, M. Gendreau, N. Lahrichi, and W. Rei. A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. *Oper. Res.*, 60(3):611–624, 2012.

M. Wen, J.-F. Cordeau, G. Laporte, and J. Larsen. The dynamic multi-period vehicle routing problem. *Comput. Oper. Res.*, 37(9):1615–1623, 2010.

W. Yang, K. Mathur, and R. H. Ballou. Stochastic vehicle routing problem with restocking. *Transportation Science*, 34:99–112, 2000.