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 Column Generation Approach for a Multi-Attribute Vehicle Routing Problem

Iman Dayarian
Teodor Gabriel Crainic
Michel Gendreau
Walter Rei

October 2013

CIRRELT-2013-57

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

Université Laval · UQÀM · HEC Montréal · École Polytechnique Montréal · Université de Montréal

A Column Generation Approach for a Multi-Attribute 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 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.

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

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) (see Hartl et al., 2006). 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 deliveries 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 2, we describe in detail the problem class and its different variants. In Section 3, we give a brief literature review to better position the present study. In Section 4, we choose a special case of the problem class and present the set partitioning model. In Section 5, we present the proposed solution methodology, and experimental results are given in Section 6. Finally, Section 7 provides concluding remarks.

2. Problem Class

In this section, we introduce a new MAVRP variant inspired by the dairy problem in Quebec (see Lahrichi et al., 2012). 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.

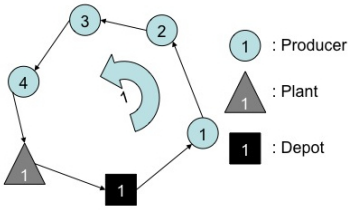

Figure 1: General configuration of simple route

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

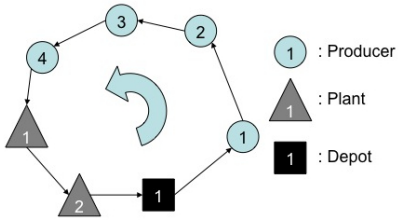

Figure 2: General configuration of multi-drop route

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.

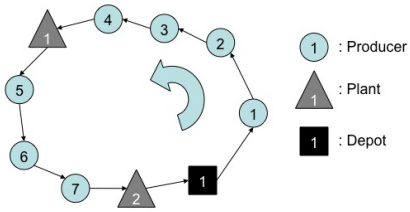

Figure 3: General configuration of interlaced multi-drop route

We consider simple routes, and Section 4 gives the details of this subclass of the problem.

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. (1992) 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 (1986) to optimality or near optimality. Kohl et al. (1999) improved the method by adding 2-path inequalities to the LP relaxation of the set partitioning formulation. Kohl and Madsen (1997) 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 (2006) 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. (2004) improved the extension of the Ford–Bellman algorithm proposed by Desrochers (1988). 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 (2004) 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. (2006) and Righini and Salani (2009) 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. (2006). Later, Desaulniers et al. (2008) 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. (2011a) 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. (2011b) 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 (2012) 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 (2009). 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. (2011) 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 prob-

lem considered 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.

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 5 we show how to adapt the approach for variant 2. We assume that the vehicles perform simple routes as described in Section 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 v_k, respectively. More precisely, v_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 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:

$$\min \quad \sum_{u\in\mathcal{U}} \sum_{d\in\mathcal{D}} \sum_{k\in\mathcal{K}_d} \sum_{p\in\mathcal{P}_{du}^k} (c_p + c_k) y_p \tag{1}$$

subject to

$$\sum_{u\in\mathcal{U}} \sum_{d\in\mathcal{D}} \sum_{k\in\mathcal{K}_d} \sum_{p\in\mathcal{P}_{du}^k} a_{ip} y_p = 1 \quad (i \in \mathcal{N}); \tag{2}$$

$$\sum_{d\in\mathcal{D}} \sum_{k\in\mathcal{K}_d} \sum_{p\in\mathcal{P}_{du}^k} g_p y_p \geq D_u \quad (u \in \mathcal{U}); \tag{3}$$

$$y_p \in \{1, 0\} \quad (d \in \mathcal{D}; k \in \mathcal{K}_d; u \in \mathcal{U}; p \in \mathcal{P}_{du}^k). \tag{4}$$

Constraint (2) ensures that each producer is visited exactly once by exactly one route, and constraint (3) guarantees that the plant demands are satisfied. In the following sections, we describe our algorithm in detail.

5. Solution method

In this section, we present our algorithm. In the path-based integer model (1)–(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

(2005)). 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 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 5.1–5.4, we describe the branch-and-price algorithm.

5.1. Master problem

A relaxation of constraints (2) converts the set partitioning model into a set covering model and 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)

$$\min \quad \sum_{u\in\mathcal{U}} \sum_{d\in\mathcal{D}} \sum_{k\in\mathcal{K}_d} \sum_{p\in\mathcal{P}'^{k}_{du}} (c_p + c_k) y_p \tag{5}$$

subject to

$$\sum_{u\in\mathcal{U}}\sum_{d\in\mathcal{D}}\sum_{k\in\mathcal{K}_d}\sum_{p\in\mathcal{P}'^{k}_{du}} a_{ip}y_p \geq 1 \quad (i\in\mathcal{N}); \tag{6}$$

$$\sum_{d\in\mathcal{D}}\sum_{k\in\mathcal{K}_d}\sum_{p\in\mathcal{P}'^{k}_{du}} g_p y_p \geq D_u \quad (u\in\mathcal{U}); \tag{7}$$

$$0 \leq y_p \leq 1 \quad (d\in\mathcal{D}; k\in\mathcal{K}_d; u\in\mathcal{U}; p\in\mathcal{P}'^{k}_{du}) \tag{8}$$

where constraint (6) ensures at least one visit to each producer, and constraint (7) guarantees that the plant demands are satisfied.

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 (1964). 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:

$$saving_{ij} = c_{0i} + c_{0j} - c_{ij} \tag{9}$$

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.

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 (5)–(8):
λ_i: nonnegative dual variable of (6) for producer $i \in \mathcal{N}$;
μ_u: nonnegative dual variable of (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 $0_d, d \in \mathcal{D}$, and services the producers of the set $\mathcal{C}_d$, is as follows:

$$\min \quad \sum_{i \in \mathcal{C}_d} \sum_{j \in \mathcal{C}_d} (v_k c_{ij} - \lambda_i) x_{ij} - \sum_{u \in \mathcal{U}} g_p \mu_u + c_k \tag{10}$$

subject to

$$\sum_{i \in \mathcal{C}_d} x_{ih} - \sum_{j \in \mathcal{C}_d} x_{hj} = 0 \quad (h \in \mathcal{C}_d); \tag{11}$$

$$\sum_{j \in \mathcal{C}_d} x_{0_d j} = 1; \tag{12}$$

$$f_u - \sum_{i \in \mathcal{C}_d} x_{iu} = 0 \quad (u \in \mathcal{U}); \tag{13}$$

$$\sum_{u \in \mathcal{U}} f_u = 1; \tag{14}$$

$$f_u - x_{u0'_d} = 0 \quad (u \in \mathcal{U}); \tag{15}$$

$$\sum_{i \in \mathcal{C}_d} \sum_{u \in \mathcal{U}} x_{ui} = 0; \tag{16}$$

$$x_{ij}(t_i + s_i + t_{ij} - t_j) \leq 0 \quad (i \in \mathcal{C}_d; j \in \mathcal{C}_d \cup \mathcal{U}); \tag{17}$$

$$e_i \leq t_i \leq l_i \quad (i \in \mathcal{C}_d \cup \mathcal{U}); \tag{18}$$

$$g_p \leq \sum_{i \in \mathcal{C}_d} q_i \sum_{j \in \mathcal{C}_d} x_{ij} \leq Q_k; \tag{19}$$

$$x_{ij} \in \{1, 0\} \quad (i, j \in \mathcal{C}_d); \tag{20}$$

$$f_u \in \{1, 0\} \quad (u \in \mathcal{U}). \tag{21}$$

Constraints 11–16 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 17 and 18 are time-window constraints. Constraint 19 is the capacity constraint, and Constraints 20 and 21 ensure integrality.

The subproblems above are variants of the ESPPRC and thus are NP-hard problems in the strong sense (Dror, 1994). 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, much research has focused on the computational efficiency of the labeling procedure for the ESPPRC subproblem. DSSR (see Righini and Salani, 2009; Boland et al., 2006) and the *ng*-route relaxation (see Baldacci et al., 2011a) 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 (see Martinelli, 2012) 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 represent 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 (ngR-DSSR)

The *ng*-route relaxation (Baldacci et al., 2011a), 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

$$\Pi(r) = \{i_j \in \mathcal{V}_{rd} | i_j \in \bigcap_{k=j+1}^{n} \mathcal{N}_{i_k}, j = 1, \ldots, n-1\} \cup \{i_n\}. \tag{22}$$

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 $\sigma_1 = (C_1, T_1, L_1, S_{ng_1}, \Pi_1)$ and $\sigma_2 = (C_2, T_2, L_2, S_{ng_2}, \Pi_2)$, 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}) $S_{ng_1} \leq S_{ng_2}$,

(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 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.

5.4. Branching strategy

As mentioned in Section 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 (see Gélinas et al., 1995).

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. (1995) for the VRPTW, splits the time window of a node into two new subin-

tervals; each branch corresponds to one of the subintervals. Some routes become infeasible following a split in a producer's time window. Gélinas et al. (1995) 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.

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.

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. (1997), 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 $z < exp(-h(min_d.\alpha + min_p(1-\alpha)))$ 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 4 shows an example of an instance with three depots, three outside plants, and 100 producers.

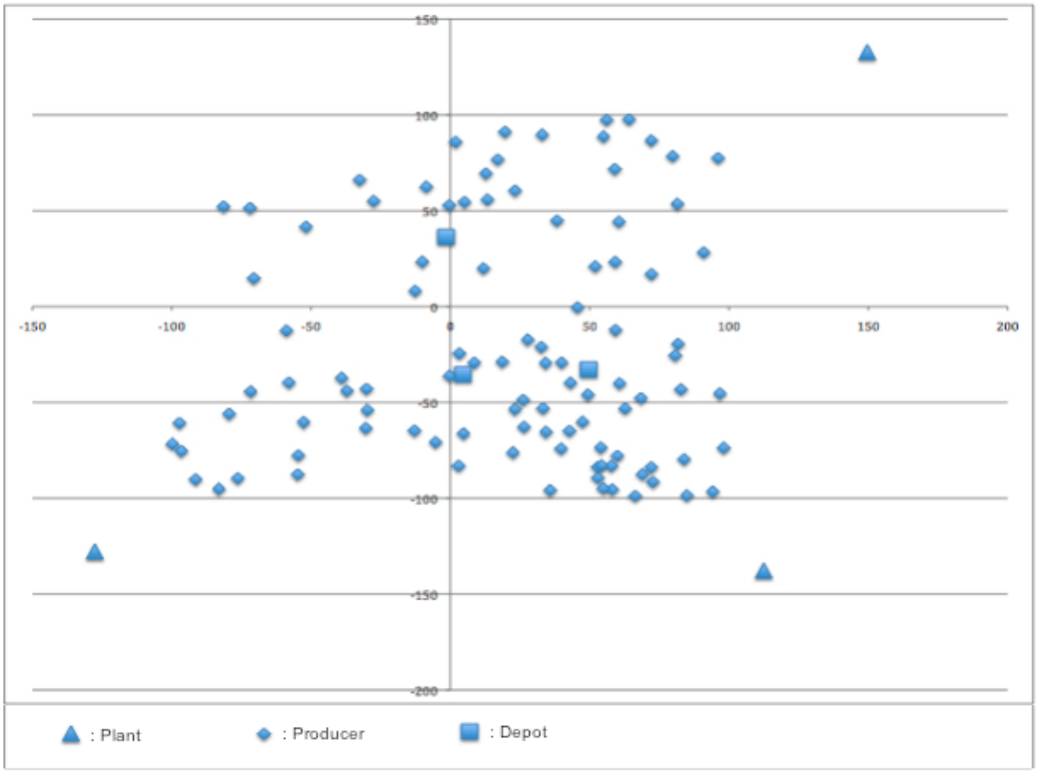

Figure 4: Producer locations in case with three outside plants

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 1 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 2. 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 1: Four problem classes

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

Table 2: Specifications of test problems

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

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 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.

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

Table 3: 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)

generate columns; we implemented a method based on TS. This approach, inspired by the procedure of Desaulniers et al. (2008), 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. (2008). However, our results for instances with 100 and 200 nodes show that TS is more efficient for larger instances and longer routes.

6.3. Branching and integer solution

As mentioned in Section 5, a branching scheme is often necessary. We now evaluate the performance of the two branching strategies introduced in Section 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 .8–.11 present the results for variant 1, and Tables .12–.15 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.

6.3.1. 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 4 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 5 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.

Using DSSR with BTW gives the best results. When BTW is used, DSSR almost always outperforms ngR-DSSR in terms of the percentage of solved problems. DSSR also has a smaller T1.

Table 4: 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	73%	28%	6664	3480	90%	5%	2889	1217
	40	45%	40%	11860	5725	76%	10%	5577	11748
	50	28%	41%	13290	2661	65%	8%	8267	4288
ng5	30	68%	33%	7024	1987	89%	5%	3008	1074
	40	44%	31%	11993	5570	74%	10%	5805	2031
	50	33%	18%	13129	8663	63%	14%	8257	3737
ng8	30	73%	26%	6713	3483	89%	5%	2904	1023
	40	44%	41%	11901	6097	76%	9%	5717	2474
	50	31%	39%	13291	3318	64%	10%	8216	4033

Table 5: 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	53%	40%	9363	1530	95%	2%	1590	786
	pr02	38%	30%	12711	6569	43%	20%	10881	2583
	pr03	57%	40%	8749	2952	97%	3%	1590	992
	pr04	45%	35%	11595	4770	73%	5%	8250	5309
ng5	pr01	48%	45%	9989	1158	95%	3%	1493	665
	pr02	30%	20%	13564	9600	48%	13%	10968	4209
	pr03	70%	22%	7181	2634	97%	3%	1695	1096
	pr04	43%	22%	12127	8234	60%	18%	8602	3154
ng8	pr01	55%	40%	9423	2039	95%	3%	1616	819
	pr02	45%	23%	12772	7989	48%	13%	10821	4177
	pr03	55%	42%	8753	2585	97%	3%	1525	926
	pr04	42%	37%	11593	4583	65%	12%	8487	4119

6.3.2. Variant 2: No preassignment

We solved the same instances using the algorithm adapted for variant 2. The performance of our algorithm depends on the producer-depot ratio, and

variant 2 is therefore more difficult to solve. We used the same two branching strategies. Tables .12–.15 report the results for pr01–pr04 with 30, 40, and 50 producers.

We again group the instances according to the number of producers. Tables 6 and 7 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 6 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.

Table 6: 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	66%	34%	6937	1465	85%	8%	4527	2610
	40	43%	38%	11670	4342	59%	9%	9358	5300
	50	33%	19%	13313	8566	45%	9%	11239	7785
ng5	30	68%	33%	7020	2047	88%	6%	3008	905
	40	43%	33%	11999	5021	71%	13%	5805	1307
	50	33%	18%	13126	8661	65%	11%	8256	4080
ng8	30	66%	34%	6945	1361	84%	6%	4564	2288
	40	44%	34%	11731	4638	58%	9%	9662	5099
	50	33%	16%	13575	8655	45%	9%	11059	7731

Table 7: 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	48%	43%	9971	1127	83%	15%	4679	2415
	pr02	28%	27%	13761	8244	33%	3%	13672	9118
	pr03	68%	25%	7155	1852	92%	5%	3462	2450
	pr04	43%	25%	11673	7942	43%	10%	11685	6945
ng5	pr01	48%	45%	9983	1149	85%	13%	4970	2958
	pr02	28%	23%	13813	8288	30%	2%	13694	8039
	pr03	68%	23%	7181	1885	90%	8%	3498	2196
	pr04	45%	20%	11999	8367	43%	10%	11657	7511
ng8	pr01	48%	45%	9983	1149	85%	13%	4970	2958
	pr02	28%	23%	13813	8288	30%	2%	13694	8039
	pr03	68%	23%	7181	1885	90%	8%	3498	2196
	pr04	45%	20%	12024	8216	43%	8%	11551	6966

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 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.

References

R. Baldacci and A. Mingozzi. A unified exact method for solving different classes of vehicle routing problems. *Math. Program.*, 120(2):347–380, 2009.

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

R. Baldacci, A. Mingozzi, and R. Roberti. New state-space relaxations for solving the traveling salesman problem with time windows. *INFORMS J. Comput.*, 2011b.

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.

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.

G. Clarke and J. W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Oper. Res.*, 12(4):568–581, 1964.

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.

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

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. Desrochers. An algorithm for the shortest path problem with resource constraints. Technical report, GERAD, 1988.

M. Desrochers, J. Desrosiers, and M. Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Oper. Res.*, 40:342–354, 1992.

M. Dror. Note on the complexity of the shortest path models for column generation in VRPTW. *Oper. Res.*, 42:977–978, 1994.

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.

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.

R. F. Hartl, G. Hasle, and G. K. Janssens, editors. *Special issue on rich vehicle routing problems*, volume 14. 2006.

S. Irnich and D. Villeneuve. The shortest path problem with resource constraints and k-cycle elimination for $k >= 3$. *INFORMS J. Comput.*, 18 (3):391–406, 2006.

N. Kohl and O. B. G. Madsen. An optimization algorithm for the vehicle routing problem with time windows based on Lagrangian relaxation. *Oper. Res.*, 45(3):395–406, 1997.

N. Kohl, J. Desrosiers, O. B. G. Madsen, M. M. Solomon, and F. Soumis. 2-path cuts for the vehicle routing problem with time windows. *Transportation Science*, 33:101–116, 1999.

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.

M. E. Lübbecke and J. Desrosiers. Selected topics in column generation. *Oper. Res.*, 53:1007–1023, 2005. ISSN 0030-364X.

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. Dynamic programming algorithms for the elementary shortest path problem with resource constraints. *Electronic Notes in Discrete Mathematics*, 17:247–249, 2004.

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.

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.

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

Table .8: Results for instances with inside plants and narrow time windows (pr01); variant 1

		DSSR						ng5						ng8					
Instance	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	3	2	7200.1	0.2	0.0	2.7	–	4	1	3601.3	1.7	0.1	9.8	–	3	2	7200.2	0.3	0.0	2.9	–
	BTW	5	0	4.6	4.6	3.6	–	–	5	0	4.9	4.9	3.6	–	–	5	0	4.9	4.9	3.6	–	–
pr01-30-2D3P	BPA	2	3	11901.7	2754.4	3.5	4.1	–	3	2	7205.4	9	0.2	5.1	–	2	3	12135.4	3338.4	3.5	4.8	–
	BTW	5	0	8.2	8.2	6.9	–	–	5	0	9	9	6.9	–	–	5	0	9	9	6.9	–	–
pr01-30-3D2P	BPA	5	0	38.5	38.5	1.9	–	–	3	2	7201.6	2.7	0.2	5.2	–	5	0	39	39	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	7200.5	4500.6	1.7	3.9	–	2	3	10801.5	3.9	0.3	8.0	–	4	1	7200.6	4500.8	1.7	4.0	–
	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	8883.7	2806.2	2.2	3.3	2.1	4	1	3605.2	6.5	0.1	11.3	–	3	2	9097.7	3162.8	2.2	5.1	–
	BTW	5	0	22.2	22.2	4.3	–	–	5	0	16.8	16.8	4.3	–	–	5	0	16.7	16.7	4.3	–	–
pr01-40-2D3P	BPA	4	1	7203	4503.7	0.5	4.8	–	1	3	14421.3	106.7	0.3	7.8	2.0	4	1	7203.7	4504.7	0.5	4.9	–
	BTW	5	0	4.9	4.9	1.5	–	–	5	0	5.5	5.5	1.5	–	–	5	0	5.5	5.5	1.5	–	–
pr01-40-3D2P	BPA	2	3	12247.3	3618.2	3.8	5.3	–	1	4	14400.7	3.5	0.1	5.7	–	2	3	12519.4	4298.5	3.8	5.6	–
	BTW	5	0	5	5	6.6	–	–	5	0	4	4	6.6	–	–	5	0	4	4	6.6	–	–
pr01-40-3D3P	BPA	2	3	10800.9	2.2	0.1	4.7	–	2	2	10835.6	89	0.3	6.3	2.4	2	3	10801	2.5	0.1	5.2	–
	BTW	5	0	13.1	13.1	4.4	–	–	5	0	14.2	14.2	4.4	–	–	5	0	14.3	14.3	4.4	–	–
pr01-50-2D2P	BPA	2	2	10803.7	9.3	0.0	3.1	1.9	2	2	10809.9	24.7	0.1	4.2	0.7	2	2	10804.8	12	0.0	2.8	7.0
	BTW	4	0	3602.9	3.6	1.8	–	7.6	4	0	3603	3.7	1.8	–	8.5	4	0	3603	3.8	1.8	–	8.0
pr01-50-2D3P	BPA	1	3	14402.3	11.5	0.1	6.5	0.9	3	2	11121.9	6536.6	0.7	5.9	–	1	3	14402.8	14.2	0.1	6.1	1.0
	BTW	4	1	8023.5	5529.3	6.6	3.9	–	4	1	6980.3	4225.4	6.6	3.5	–	4	1	8310.3	5887.9	6.6	3.5	–
pr01-50-3D2P	BPA	1	3	14400.1	0.6	4.2	5.0	8.8	1	4	14402.5	12.6	0.1	7.5	–	1	3	14400.1	0.7	4.2	3.8	3.2
	BTW	4	0	3760.5	200.6	5.6	–	15.8	4	1	3651.4	64.3	5.6	1.9	–	4	1	3797.2	246.5	5.6	5.0	–
pr01-50-3D3P	BPA	3	2	7268.6	114.3	0.1	2.6	–	3	1	11461.8	7102.9	0.9	6.2	0.4	4	1	7276.1	4595.2	0.7	3.8	–
	BTW	5	0	3637.2	3637.2	1.4	–	–	5	0	3630	3630	1.4	–	–	5	0	3630.2	3630.2	1.4	–	–

Table .9: Results for instances with inside plants and wide time windows (pr02); variant 1

		DSSR						ng5						ng8					
Instance	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	8455.8	2093	5.1	6.3	–	5	0	4527.5	4527.5	5.5	–	–	4	1	8440.3	6050.3	9.9	8.8	–
	BTW	2	2	10801.2	3.1	0.2	4.0	15.0	2	2	10801.2	2.9	0.2	4.7	11.2	2	2	10801.2	3	0.2	2.3	11.2
pr02-30-2D3P	BPA	3	2	10870.1	6116.9	5.1	7.0	–	2	3	11199.7	999.3	2.8	6.5	–	4	1	10890.9	9113.6	7.6	5.9	–
	BTW	4	1	7214	4517.6	6.0	3.3	–	5	0	7209.5	7209.5	7.9	–	–	4	1	7209.2	4511.6	6.1	3.0	–
pr02-30-3D2P	BPA	4	1	7698.3	5122.9	16.1	3.4	–	1	4	10316.6	9582.8	8.1	7.1	–	4	1	8173.4	5716.8	16.1	3.7	–
	BTW	3	1	7817.3	1028.8	10.6	5.9	14.8	3	0	7843.8	1073.1	10.6	–	16.8	3	0	7717.8	863.1	10.6	–	19.2
pr02-30-3D3P	BPA	4	1	7854.6	5318.3	9.9	7.0	–	3	2	10942.6	6237.6	3.6	12.2	–	4	1	7943.8	5429.8	9.9	3.8	–
	BTW	4	0	3891.4	364.2	6.4	–	25.2	4	0	3852.7	315.8	6.4	–	6.7	4	0	3838.2	297.8	6.4	–	14.2
pr02-40-2D2P	BPA	1	1	18000	18000	4.9	5.4	3.8	0	2	18000	18000	–	10.6	3.2	1	1	18000	18000	4.9	5.1	7.4
	BTW	1	1	14517.7	588.5	4.9	3.4	10.0	2	0	14577.8	9444.6	6.8	–	6.3	2	0	14687.6	9719.1	6.7	–	6.3
pr02-40-2D3P	BPA	2	2	10801.8	4.5	0.1	5.2	6.1	1	0	16221.1	9105.5	2.8	–	3.1	2	2	10802.2	5.6	0.1	5.5	5.9
	BTW	3	1	9835	4391.7	2.1	3.4	9.0	3	1	10804.1	6006.8	2.1	3.6	8.7	3	1	9367.9	3613.2	2.1	3.6	8.1
pr02-40-3D2P	BPA	1	4	14400.4	2	0.1	5.1	–	3	1	11429.4	7049	2.8	14.2	3.6	2	3	13342	6355.1	4.7	6.7	–
	BTW	2	3	10802.1	5.2	4.7	2.6	–	3	1	10804.2	6007	7.8	7.3	12.0	3	2	10801.2	6002	7.4	5.8	–
pr02-40-3D3P	BPA	2	1	13197.9	5994.8	8.0	5.9	4.1	2	0	12782.2	4955.4	2.2	–	3.4	2	1	14406.2	9015.6	8.0	6.0	3.6
	BTW	3	0	7516.1	526.9	7.6	–	14.2	3	0	7400.2	333.6	7.6	–	11.1	3	0	7399.3	332.1	7.6	–	12.1
pr02-50-2D2P	BPA	0	1	18000	18000	–	1.6	2.7	0	0	14400	18000	–	–	1.5	1	0	18000	18000	3.4	–	3.1
	BTW	1	0	14404.5	22.3	3.1	–	4.4	1	0	14402.8	13.9	3.1	–	4.2	1	0	14403	15.2	3.1	–	5.1
pr02-50-2D3P	BPA	0	1	18000	18000	–	6.7	3.6	0	0	18000	18000	–	–	1.6	0	1	18000	18000	–	7.0	2.3
	BTW	0	2	18000	18000	–	7.1	6.7	0	1	18000	18000	–	4.6	7.8	0	1	18000	18000	–	7.4	5.3
pr02-50-3D2P	BPA	2	1	10837.3	93.2	0.5	5.9	3.7	1	0	10949.3	746.3	1.1	–	1.7	2	1	10843.6	108.9	0.5	7.3	2.8
	BTW	2	1	11321.5	1303.9	0.5	1.5	7.5	2	1	11403.5	1508.7	0.5	1.5	7.3	3	0	11218.4	6697.3	3.3	–	9.5
pr02-50-3D3P	BPA	1	1	14415.4	77.1	0.1	7.9	2.2	0	0	18000	18000	–	–	1.2	1	1	14415.6	77.8	0.1	4.8	4.2
	BTW	1	0	14449.9	249.6	0.1	–	8.3	1	2	14517.6	588.1	0.1	7.0	5.9	1	1	14413.6	67.8	0.1	4.3	8.9

Table .10: Results for instances with outside plants and narrow time windows (pr03); variant 1

		DSSR						ng5						ng8					
Instance	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	65	65	2.5	–	–	5	0	0.7	0.7	0.1	–	–	5	0	67.6	67.6	2.5	–	–
	BTW	5	0	0.5	0.5	2.5	–	–	5	0	0.5	0.5	2.5	–	–	5	0	0.5	0.5	2.5	–	–
pr03-30-2D3P	BPA	0	5	18000	18000	–	3.3	–	3	2	7212.1	20.1	0.5	4.2	–	0	5	18000	18000	–	3.9	–
	BTW	5	0	3073.7	3073.7	10.0	–	–	5	0	3743.1	3743.1	10.0	–	–	5	0	3743.5	3743.5	10.0	–	–
pr03-30-3D2P	BPA	5	0	31.3	31.3	2.9	–	–	4	1	7203.1	4503.9	2.3	9.2	–	5	0	31	31	2.9	–	–
	BTW	5	0	0.7	0.7	2.9	–	–	5	0	0.7	0.7	2.9	–	–	5	0	0.7	0.7	2.9	–	–
pr03-30-3D3P	BPA	4	1	3600.8	1	0.2	6.7	–	5	0	7.1	7.1	0.4	–	–	4	1	3600.8	1	0.2	5.9	–
	BTW	5	0	5.8	5.8	3.2	–	–	5	0	6.9	6.9	3.2	–	–	5	0	6.6	6.6	3.2	–	–
pr03-40-2D2P	BPA	3	2	7642.2	736.9	1.9	3.0	–	4	1	7269.8	4587.3	2.0	4.4	–	3	2	7669.7	782.8	1.9	3.4	–
	BTW	4	1	3610.4	13	2.8	5.5	–	4	1	3611	13.8	2.8	5.6	–	4	1	3611	13.8	2.8	3.6	–
pr03-40-2D3P	BPA	3	2	10813.4	6022.4	1.8	4.1	–	3	0	7340.7	234.5	0.5	–	4.2	3	2	10815.2	6025.4	1.8	6.5	–
	BTW	5	0	440.7	440.7	4.0	–	–	5	0	446.4	446.4	4.1	–	–	5	0	436.4	436.4	4.0	–	–
pr03-40-3D2P	BPA	3	2	7200.1	0.2	0.0	9.1	–	2	3	14401.9	9004.8	4.1	5.9	–	3	2	7200.1	0.2	0.0	9.1	–
	BTW	5	0	0.8	0.8	4.6	–	–	5	0	0.8	0.8	4.6	–	–	5	0	0.8	0.8	4.6	–	–
pr03-40-3D3P	BPA	4	1	7204.4	4505.5	1.2	7.4	–	2	2	10812.8	32	0.4	3.7	2.4	3	2	7204.8	8	0.3	3.8	–
	BTW	5	0	251.8	251.8	2.7	–	–	5	0	256.4	256.4	2.7	–	–	5	0	254.6	254.6	2.7	–	–
pr03-50-2D2P	BPA	2	3	10890.7	1.8	0.1	5.1	–	4	1	9896	7870	2.0	2.4	–	3	2	10890.9	6001.5	2.1	10.1	–
	BTW	4	1	3666.4	83	3.4	4.0	–	4	1	3620.2	25.3	3.4	5.5	–	4	1	3620.2	25.3	3.4	5.4	–
pr03-50-2D3P	BPA	3	2	10832.5	6054.2	1.2	3.4	–	2	2	11046	615	1.0	3.2	1.5	2	3	10839.8	99.5	0.2	2.9	–
	BTW	5	0	1123.9	1123.9	2.5	–	–	5	0	1432.4	1432.4	2.5	–	–	5	0	1422	1422	2.5	–	–
pr03-50-3D2P	BPA	1	3	14400.1	0.3	0.0	1.6	1.5	4	0	7210	4512.5	0.6	–	1.1	1	3	14400.1	0.4	0.0	1.7	0.3
	BTW	5	0	5.5	5.5	3.8	–	–	5	0	5.8	5.8	3.8	–	–	5	0	6.2	6.2	3.8	–	–
pr03-50-3D3P	BPA	1	3	14401.3	6.7	0.1	3.1	2.8	4	1	3776	220.1	0.3	9.8	–	1	3	14401.5	7.7	0.1	3.4	2.9
	BTW	5	0	6900.8	6900.8	4.2	–	–	5	0	7214.6	7214.6	4.2	–	–	5	0	5201.3	5201.3	4.2	–	–

Table .11: Results for instances with outside plants and wide time windows (pr04); variant 1

		DSSR						ng5						ng8					
Instance	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	8414.3	6017.8	7.9	1.3	–	3	2	7401.7	336.1	3.9	6.4	–	3	2	8467.9	2113.1	5.7	3.1	–
	BTW	5	0	2763.1	2763.1	8.4	–	–	4	1	3683.9	104.9	10.0	19.9	–	5	0	1901.9	1901.9	8.4	–	–
pr04-30-2D3P	BPA	3	2	7223.2	38.7	0.9	12.1	–	4	1	3843.6	304.5	3.2	5.4	–	3	1	7217.8	29.7	0.9	9.8	20.8
	BTW	4	0	6256.3	3320.4	3.6	–	11.5	4	0	6944.6	4180.8	3.6	–	10.4	4	0	7205.6	4507	3.6	–	13.4
pr04-30-3D2P	BPA	5	0	54.9	54.9	1.3	–	–	4	1	7440	4800.1	4.2	6.9	–	5	0	53.4	53.4	1.3	–	–
	BTW	5	0	1.9	1.9	1.3	–	–	5	0	1.3	1.3	1.3	–	–	5	0	1.3	1.3	1.3	–	–
pr04-30-3D3P	BPA	4	1	8014.8	5518.5	5.1	1.4	–	3	2	7472	453.4	6.0	5.0	–	3	2	7949.2	1248.7	2.8	2.3	–
	BTW	5	0	4382.9	4382.9	5.5	–	–	4	1	4021.7	527.2	5.5	1.1	–	4	1	4019.5	524.4	5.5	1.0	–
pr04-40-2D2P	BPA	2	2	10958.4	396.1	1.4	76.1	10.5	3	2	8742.6	2571	2.7	10.3	–	2	2	10951.4	378.5	1.4	2.7	5.9
	BTW	2	2	10802.1	5.2	1.4	2.4	9.4	3	1	10802.2	6003.7	5.2	2.3	9.3	3	0	10802.2	6003.7	4.3	–	9.8
pr04-40-2D3P	BPA	0	2	18000	18000	–	7.0	6.0	3	0	9045.2	3075.4	2.9	–	4.4	0	2	18000	18000	–	6.8	2.9
	BTW	2	0	12693.1	4732.8	8.9	–	10.5	1	1	14425.9	129.5	8.7	2.7	6.6	1	1	14426	130	8.7	2.3	6.2
pr04-40-3D2P	BPA	2	3	18000	18000	8.9	4.6	–	1	3	18000	18000	2.1	8.0	3.5	1	4	18000	18000	4.2	4.5	–
	BTW	5	0	7725.1	7725.1	8.9	–	–	3	2	7736.7	894.5	6.4	3.0	–	4	1	7705.4	5131.8	8.3	2.7	–
pr04-40-3D3P	BPA	2	2	14402.2	9005.4	2.4	13.2	5.4	3	1	14582.4	12304	4.0	3.5	7.8	2	1	14402.2	9005.5	2.4	12.7	8.3
	BTW	4	0	10986.6	9233.3	7.7	–	22.1	2	1	11967.3	2918.4	6.2	2.0	10.9	3	1	11943.1	7905.2	5.7	1.9	19.7
pr04-50-2D2P	BPA	1	4	14402.6	13.1	0.1	3.9	–	0	1	18000	18000	–	3.7	3.1	1	4	14402.7	13.5	0.1	4.0	–
	BTW	4	0	7451.5	4814.4	4.7	–	5.7	4	1	7548.1	4935.1	4.6	5.8	–	4	0	7732.6	5165.8	4.7	–	5.9
pr04-50-2D3P	BPA	1	1	14403.2	16.1	0.2	6.6	2.9	1	0	18000	18000	2.6	–	2.4	1	2	14404.1	20.7	0.2	6.3	2.0
	BTW	2	1	14407.2	9018.1	3.8	5.2	4.2	1	1	14407.7	38.4	0.2	2.7	3.3	1	1	14407.7	38.5	0.2	2.7	3.4
pr04-50-3D2P	BPA	2	2	10859.4	148.6	1.4	3.2	5.8	1	0	14993.2	2965.8	1.8	–	3.0	3	1	10861.1	6101.8	2.0	4.1	6.0
	BTW	4	0	3678.7	98.3	3.0	–	10.9	4	0	3686.7	108.4	3.0	–	10.5	4	0	3697.7	122.1	3.0	–	12.8
pr04-50-3D3P	BPA	1	1	14405.2	26	0.4	4.2	4.5	0	0	18000	18000	–	–	2.5	1	1	14405.8	29.2	0.4	4.5	3.9
	BTW	2	0	17845.5	17613.9	4.9	–	9.6	1	2	18000	18000	0.9	2.1	13.1	1	2	18000	18000	0.6	2.5	12.1

Table .12: Results for instances with inside plants and narrow time windows (pr01); variant 2

		DSSR						ng5						ng5					
Instance	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	3601.2	1.5	0.1	8.9	–	4	1	3601.4	1.7	0.1	8.1	–	4	1	3601.4	1.7	0.1	9.4	–
	BTW	4	1	3601.8	2.2	0.1	4.3	–	5	0	4.8	4.8	3.6	–	–	4	1	3601.7	2.1	0.1	5.0	–
pr01-30-2D3P	BPA	3	2	7204.8	8	0.2	6.9	–	3	2	7205.4	9.1	0.2	6.3	–	3	2	7205.4	9	0.2	6.7	–
	BTW	5	0	714.6	714.6	4.8	–	–	5	0	9	9	6.9	–	–	5	0	2538.2	2538.2	4.8	–	–
pr01-30-3D2P	BPA	3	2	7201.3	2.1	0.2	4.8	–	3	2	7201.6	2.7	0.2	5.2	–	3	2	7201.7	2.8	0.2	5.1	–
	BTW	5	0	13.4	13.4	3.3	–	–	5	0	0.1	0.1	1.9	–	–	5	0	14.5	14.5	3.3	–	–
pr01-30-3D3P	BPA	2	3	10801.1	2.8	0.3	8.0	–	2	3	10801.5	3.7	0.3	7.9	–	2	3	10801.5	3.8	0.3	7.9	–
	BTW	5	0	95.9	95.9	6.0	–	–	5	0	0.5	0.5	2.7	–	–	5	0	133.8	133.8	6.0	–	–
pr01-40-2D2P	BPA	4	1	3604.2	5.2	0.1	11.3	–	4	1	3604.9	6.2	0.1	11.5	–	4	1	3604.4	5.5	0.1	11.5	–
	BTW	4	1	3602.2	2.8	0.1	8.6	–	5	0	16.8	16.8	4.3	–	–	4	1	3602.5	3.1	0.1	8.8	–
pr01-40-2D3P	BPA	1	2	14415.6	78	0.3	8.4	1.3	1	3	14421.4	106.9	0.3	7.8	1.9	1	3	14421.2	106.1	0.3	7.8	2.0
	BTW	4	1	7589	4986.3	6.5	4.3	–	5	0	5.4	5.4	1.5	–	–	4	1	7615.8	5019.7	6.4	3.9	–
pr01-40-3D2P	BPA	1	4	14400.9	4.3	0.1	5.8	–	1	4	14400.7	3.5	0.1	5.8	–	1	4	14400.7	3.5	0.1	5.8	–
	BTW	4	1	7438.3	4797.9	7.7	4.8	–	5	0	3.9	3.9	6.6	–	–	4	1	9992	7990	7.7	5.4	–
pr01-40-3D3P	BPA	2	2	10830.7	76.8	0.3	6.2	2.5	2	2	10835.6	89.1	0.3	6.1	2.2	2	2	10835.3	88.4	0.3	6.3	2.2
	BTW	5	0	2924.9	2924.9	4.9	–	–	5	0	14.4	14.4	4.4	–	–	5	0	2481.3	2481.3	4.9	–	–
pr01-50-2D2P	BPA	2	2	10807.3	18.4	0.1	3.8	0.7	2	2	10809.9	24.7	0.1	4.2	0.7	2	2	10809.1	22.9	0.1	4.3	0.7
	BTW	4	1	4156.8	696	2.6	3.3	–	4	0	3602.9	3.7	1.8	–	7.8	4	1	3901.2	376.5	2.6	3.1	–
pr01-50-2D3P	BPA	3	2	11050	6416.7	0.7	6.0	–	3	2	11107.7	6512.9	0.7	5.9	–	3	2	11116.9	6528.1	0.7	5.9	–
	BTW	5	0	2418.1	2418.1	2.9	–	–	4	1	7041.5	4301.9	6.6	3.6	–	5	0	3001.4	3001.4	2.9	–	–
pr01-50-3D2P	BPA	1	4	14401.8	9.2	0.1	7.5	–	1	4	14402.3	11.6	0.1	7.5	–	1	4	14402.4	12.1	0.1	7.5	–
	BTW	3	2	9185.1	3308.5	3.4	7.2	–	4	1	3652	65	5.6	1.7	–	3	2	8349.2	1915.3	3.4	6.1	–
pr01-50-3D3P	BPA	3	1	11337.1	6895.2	0.9	6.6	0.4	3	1	11406.3	7010.5	0.9	6.5	0.4	3	1	11399.7	6999.4	0.9	6.5	0.4
	BTW	2	2	14407.3	9018.3	1.1	5.3	0.9	5	0	3630.2	3630.2	1.4	–	–	3	1	14408.7	12014.5	3.2	4.3	0.5

Table .13: Results for instances with inside plants and wide time windows (pr02); variant 2

		DSSR						ng5						ng5					
Instance	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	4147.6	684.5	4.4	21.4	–	4	1	4514.3	1142.8	4.4	1.7	–	5	0	4156.7	4156.7	5.5	–	–
	BTW	4	1	4082.9	603.7	4.4	20.8	–	2	2	10801.2	3	0.2	5.6	11.2	4	0	4128.1	660.2	4.4	–	10.8
pr02-30-2D3P	BPA	2	3	11236.9	1092.3	2.8	4.5	–	2	3	11196.2	990.5	2.8	6.6	–	2	3	11199.1	997.8	2.8	5.9	–
	BTW	3	0	11457.2	7095.4	6.3	–	6.7	4	1	7209.5	4511.9	6.3	3.2	–	2	0	11629.4	2073.5	4.8	–	7.1
pr02-30-3D2P	BPA	1	4	15454.6	5273.2	8.1	6.9	–	1	4	10294.8	9473.8	8.1	7.2	–	1	4	15634.1	6170.4	8.1	7.0	–
	BTW	3	1	12071.6	8119.3	9.5	3.4	8.9	3	0	7843.5	1072.4	10.6	–	19.0	3	1	13152.9	9921.5	9.5	7.3	8.4
pr02-30-3D3P	BPA	3	2	10872.3	6120.6	3.6	11.9	–	3	2	10926.7	6211.1	3.6	12.5	–	2	3	10883.5	208.8	1.1	8.3	–
	BTW	3	0	13089.5	9815.8	3.5	–	9.1	4	0	3850.5	313.1	6.4	–	8.9	3	0	11787.1	7645.2	3.5	–	8.3
pr02-40-2D2P	BPA	0	3	18000	18000	–	11.4	3.0	0	2	18000	18000	–	10.8	3.0	0	3	18000	18000	–	10.2	3.2
	BTW	0	0	18000	18000	–	–	5.0	1	1	14579.2	896.1	5.1	2.9	6.9	0	0	18000	18000	–	–	4.8
pr02-40-2D3P	BPA	1	0	15747.9	6739.3	2.7	–	3.2	1	0	16233	9164.8	2.8	–	3.0	1	0	16213.8	9069.1	2.7	–	3.1
	BTW	1	0	14682.6	1413.2	2.5	–	4.7	3	1	10804.1	6006.8	2.1	3.5	7.9	1	0	14839.7	2198.5	2.5	–	4.7
pr02-40-3D2P	BPA	3	2	8890.9	2818.2	2.5	13.1	–	3	1	11382.1	6970.1	2.8	13.8	4.0	3	1	9151.1	3251.9	2.5	13.5	3.9
	BTW	3	0	7794.2	990.4	2.5	–	4.5	2	2	10804.1	10.3	5.1	5.0	12.8	3	0	7799.5	999.2	2.5	–	4.3
pr02-40-3D3P	BPA	2	0	12270.9	3677.3	2.2	–	3.8	2	0	12792.4	4980.9	2.2	–	3.3	2	0	11991.6	2979.1	2.2	–	3.2
	BTW	2	0	14429.4	9073.6	2.2	–	5.0	3	0	7401.8	336.3	7.6	–	15.5	1	0	14469.6	348.2	3.5	–	3.9
pr02-50-2D2P	BPA	0	1	18000	18000	–	1.8	1.9	0	0	14400	18000	–	–	1.5	0	0	18000	18000	–	–	2.1
	BTW	0	0	18000	18000	–	–	2.7	1	0	14402.7	13.4	3.1	–	3.8	0	0	18000	18000	–	–	2.2
pr02-50-2D3P	BPA	0	0	18000	18000	–	–	1.8	0	0	18000	18000	–	–	1.7	0	0	18000	18000	–	–	1.8
	BTW	0	0	18000	18000	–	–	2.9	0	1	18000	18000	–	7.8	6.6	0	0	18000	18000	–	–	2.9
pr02-50-3D2P	BPA	1	0	14505.1	525.4	1.0	–	2.3	1	0	10965.7	828.7	1.1	–	1.8	1	0	14523.7	618.5	1.0	–	2.6
	BTW	1	0	14460.3	301.3	1.0	–	3.3	3	0	11399.1	6998.5	3.4	–	7.4	1	0	14523.9	619.4	1.0	–	4.3
pr02-50-3D3P	BPA	0	0	18000	18000	–	–	1.5	0	0	18000	18000	–	–	1.2	0	0	18000	18000	–	–	1.4
	BTW	0	0	18000	18000	–	–	3.2	1	2	14515.4	577	0.1	6.9	7.8	0	0	18000	18000	–	–	3.3

Table .14: Results for instances with outside plants and narrow time windows (pr03); variant 2

		DSSR						ng5						ng5					
Instance	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	0.5	0.5	0.1	–	–	5	0	0.7	0.7	0.1	–	–	5	0	0.7	0.7	0.1	–	–
	BTW	5	0	0.3	0.3	0.1	–	–	5	0	0.5	0.5	2.5	–	–	5	0	0.4	0.4	0.1	–	–
pr03-30-2D3P	BPA	3	2	7209.8	16.4	0.5	4.4	–	4	1	7212	4515	2.3	7.7	–	3	2	7212	19.9	0.5	4.5	–
	BTW	5	0	653.6	653.6	4.1	–	–	5	0	3741	3741	10.0	–	–	5	0	701.6	701.6	4.1	–	–
pr03-30-3D2P	BPA	4	1	7202.6	4503.2	2.3	8.9	–	4	1	7203.1	4503.9	2.3	8.7	–	4	1	7203.1	4503.8	2.3	9.0	–
	BTW	5	0	305.9	305.9	4.5	–	–	5	0	0.7	0.7	2.9	–	–	5	0	354.7	354.7	4.5	–	–
pr03-30-3D3P	BPA	5	0	5.9	5.9	0.4	–	–	5	0	7.1	7.1	0.4	–	–	5	0	7.1	7.1	0.4	–	–
	BTW	5	0	46.3	46.3	0.4	–	–	5	0	6.8	6.8	3.2	–	–	5	0	37.8	37.8	0.4	–	–
pr03-40-2D2P	BPA	4	1	7267.4	4584.3	2.0	4.5	–	4	1	7270	4587.5	2.0	4.5	–	4	1	7269.2	4586.6	2.0	4.2	–
	BTW	4	1	3605.1	6.4	2.0	2.4	–	4	1	3611.2	14	2.8	5.7	–	5	0	3605.5	3605.5	3.5	–	–
pr03-40-2D3P	BPA	3	1	7305.2	175.4	0.5	3.5	5.0	3	0	7340.3	233.8	0.5	–	4.0	3	0	7336.4	227.3	0.5	–	3.9
	BTW	3	1	14567.5	12279.1	0.5	2.4	6.2	5	0	446.4	446.4	4.1	–	–	3	1	14638.7	12397.8	0.5	1.7	6.0
pr03-40-3D2P	BPA	1	4	14401.6	8.2	0.1	4.4	–	1	4	14401.9	9.3	0.1	4.5	–	1	4	14402.1	10.3	0.1	4.4	–
	BTW	4	1	7327.6	4659.5	5.1	1.1	–	5	0	0.8	0.8	4.6	–	–	4	1	7267.8	4584.7	5.1	1.7	–
pr03-40-3D3P	BPA	2	2	10810.1	25.2	0.4	3.4	2.6	2	2	10812.9	32.2	0.4	3.6	2.5	2	2	10813.4	33.5	0.4	3.7	2.4
	BTW	5	0	142.7	142.7	5.4	–	–	5	0	256	256	2.7	–	–	4	1	3629	36.3	3.4	2.1	–
pr03-50-2D2P	BPA	4	1	9625.3	7531.6	2.0	2.5	–	4	1	9900.9	7876.2	2.0	2.4	–	4	1	9894.3	7867.9	2.0	2.4	–
	BTW	5	0	61	61	2.4	–	–	4	1	3620.1	25.2	3.4	4.0	–	5	0	69.8	69.8	2.4	–	–
pr03-50-2D3P	BPA	2	2	11072.9	682.2	1.0	3.9	1.4	2	2	11046	615.1	1.0	3.8	1.6	2	2	11051.4	628.4	1.0	3.9	1.5
	BTW	5	0	7305.7	7305.7	3.8	–	–	5	0	1443.7	1443.7	2.5	–	–	5	0	4094.4	4094.4	3.7	–	–
pr03-50-3D2P	BPA	4	0	7206.9	4508.7	0.6	–	1.3	4	0	7210.1	4512.6	0.6	–	1.3	4	0	7210.3	4512.9	0.6	–	1.1
	BTW	4	0	3609.7	12.1	0.6	–	4.7	5	0	5.9	5.9	3.8	–	–	4	1	3609.3	11.6	0.6	2.6	–
pr03-50-3D3P	BPA	4	1	3747.5	184.4	0.3	9.6	–	4	1	3775.7	219.6	0.3	10.1	–	4	1	3776.3	220.4	0.3	9.1	–
	BTW	5	0	3921.5	3921.5	2.4	–	–	5	0	7135.3	7135.3	4.2	–	–	4	1	3961.8	452.3	0.3	3.4	–

Table .15: Results for instances with outside plants and wide time windows (pr04); variant 2

		DSSR						ng5						ng5					
Instance	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	7350	249.9	3.9	5.8	–	3	2	7399.7	332.8	3.9	6.4	–	3	2	7350.9	251.5	3.9	6.5	–
	BTW	4	0	6084.4	3105.5	4.5	–	10.5	4	1	3684.5	105.6	10.0	20.0	–	4	0	6278.1	3347.6	4.5	–	12.8
pr04-30-2D3P	BPA	4	1	3817.9	272.4	3.2	5.0	–	4	1	3844.7	305.8	3.2	5.4	–	4	1	3798.1	247.6	3.2	4.0	–
	BTW	5	0	4010.7	4010.7	4.6	–	–	4	0	6944.7	4180.8	3.6	–	10.3	5	0	3957.1	3957.1	4.7	–	–
pr04-30-3D2P	BPA	4	1	7442.8	4803.5	4.2	6.8	–	4	1	7441.5	4801.9	4.2	6.9	–	4	1	7434.5	4793.1	4.2	7.1	–
	BTW	4	1	7979.8	5474.8	4.3	5.1	–	5	0	1.3	1.3	1.3	–	–	4	1	6681	3851.3	4.2	3.0	–
pr04-30-3D3P	BPA	3	2	7441.9	403.2	5.8	5.3	–	3	2	7472.8	454.7	6.0	4.9	–	3	2	7436.2	393.7	5.8	4.9	–
	BTW	3	2	8223.4	1705.6	6.8	7.5	–	4	1	4022.3	527.9	5.5	1.1	–	3	2	8024	1373.4	6.8	13.8	–
pr04-40-2D2P	BPA	3	2	8846.9	1911.6	2.7	10.3	–	3	2	8788.2	2647	2.7	10.3	–	3	2	8422	2036.7	2.7	10.1	–
	BTW	3	0	9025.6	3042.6	2.7	–	4.9	2	2	10802.2	5.6	1.4	3.1	10.3	3	1	8624	2373.4	2.7	7.7	5.0
pr04-40-2D3P	BPA	3	1	9100.3	3167.1	2.8	10.3	4.1	3	0	9069.2	3115.4	2.9	–	4.4	3	0	9049.5	3082.4	2.8	–	4.3
	BTW	2	0	10983.8	459.6	1.3	–	5.4	2	0	14425.9	9064.8	9.1	–	7.1	2	0	11013.7	534.3	1.3	–	5.3
pr04-40-3D2P	BPA	1	4	18000	18000	2.1	8.1	–	1	3	18000	18000	2.1	8.1	3.5	1	4	18000	18000	2.1	7.9	–
	BTW	0	1	18000	18000	–	4.1	5.2	3	2	7738.3	898.9	6.4	2.8	–	0	0	18000	18000	–	–	5.0
pr04-40-3D3P	BPA	3	1	13323.5	10205.8	4.0	1.5	8.0	3	1	14638.1	12396.8	4.0	2.7	9.5	4	0	13780.3	12725.4	4.9	–	8.1
	BTW	3	0	9614.8	4024.6	3.9	–	7.6	2	1	11975.1	2937.8	6.2	2.1	11.0	3	0	9011.4	3019	3.9	–	6.7
pr04-50-2D2P	BPA	0	1	14400	18000	–	3.7	2.3	0	1	18000	18000	–	3.7	3.1	0	0	18000	18000	–	–	3.2
	BTW	0	0	18000	18000	–	–	3.4	4	1	7547.5	4934.4	4.6	6.4	–	0	0	18000	18000	–	–	4.0
pr04-50-2D3P	BPA	1	0	18000	18000	2.6	–	2.8	1	0	18000	18000	2.6	–	2.4	1	0	18000	18000	2.6	–	2.4
	BTW	1	1	14539.5	607.3	2.2	4.7	4.3	1	1	14407.6	38.2	0.2	1.8	3.3	1	0	14558.6	792.8	2.2	–	4.2
pr04-50-3D2P	BPA	1	0	14857.3	2286.5	1.8	–	3.1	1	0	14993.5	2967.7	1.8	–	2.8	1	0	15012.9	3064.5	1.8	–	3.0
	BTW	1	0	15763.8	6818.8	1.8	–	3.6	4	0	3687.4	109.3	3.0	–	15.3	1	0	16469	10345.1	1.8	–	3.3
pr04-50-3D3P	BPA	0	0	18000	18000	–	–	2.8	0	0	18000	18000	–	–	2.6	0	0	18000	18000	–	–	2.5
	BTW	0	1	18000	18000	–	1.3	3.1	2	1	18000	18000	4.5	2.8	12.0	0	1	18000	18000	–	2.5	3.3