Submitted to
manuscript (Please, provide the mansucript number!)

A branch-and-price algorithm for 2-period vehicle routing problems

Yves Crama
HEC-Management School, University of Liège, 4000 Liège, Belgium, y.crama@ulg.ac.be

Mahmood Rezaei
HEC-Management School, University of Liège, 4000 Liège, Belgium, m.rezaei@ulg.ac.be

Tom Van Woensel
School of Industrial Engineering, Eindhoven University of Technology, 5600MB Eindhoven, Netherlands, t.v.woensel@tue.nl

We consider a Vehicle Routing Problem (VRP) with deterministic orders in two periods from a set of stores. Orders in period 1(2) can be postponed(advanced) to the other period but any diversion from the initial orders incurs a penalty. From the perspective of a Logistics Service Provider (LSP), such diversions could be beneficial if savings in the routing costs outweigh the penalties. So could they be from a store's view, as the store can set a high enough penalty to compensate the diversion from its own optimal orders. In this paper, we introduce a new model where we seek a better solution for the LSP, compared to solving two independent VRPs with fixed orders, by allowing orders to be fully postponed or advanced. We apply a branch-and-price algorithm to solve this model to optimality. Many cutting-edge techniques are implemented to have an efficient branch-and-price algorithm, and two ideas to possibly improve the upper bound are tested. We draw algorithmic and managerial insights based on our test instances.

Key words: Branch-and-price; column generation; inventory routing; multi-period vehicle routing.
History:

---

1. Introduction

Consider a Logistics Service Provider (LSP) supplying units of a single product from a central warehouse to a multitude of geographically dispersed stores. The LSP has access to an unlimited supply of the product. Independently of other stores, each store places its orders for two periods, say for example, day $t+1$ and day $t+2$. An order placed for period $t+1$ can be completely postponed to period $t+2$ while a penalty is paid by the LSP to the store for such a postponement. In a similar way, complete advancement of an order from period $t+2$ to period $t+1$ could be acceptable for a store but the LSP has to pay a penalty for it. Whether postponement/advancement is allowed and (if so) its associated penalty are specified by the store. In fact, we may assume that advancement and postponement is always allowed for all stores but their associated penalties might be too costly, which deters the LSP to do it. The LSP's objective is to minimize the routing costs in two periods and the penalties for the orders which are fully postponed/advanced. Compared to solving two independent Vehicle Routing Problems (VRPs), solving this 2-period VRP is beneficial for both sides, i.e., the LSP and the stores. Obviously, the solution of the 2-period VRP is advantageous for the LSP, as it provides more flexibility to coordinate the routing costs of two periods and consequently to decrease their sum. So is it for the stores due to the fact that they can choose a high enough penalty to compensate the costs imposed to their inventory systems by postponing/advancing.

1.1. Motivation

Our 2-period VRP may be regarded as a chunk of a Dynamic Multi-Period Vehicle Routing Problem (DMPVRP) with a finite/infinite horizon. Consider the big picture presented in Figure 1 where each store receives stochastic demands from its own customers during period t.

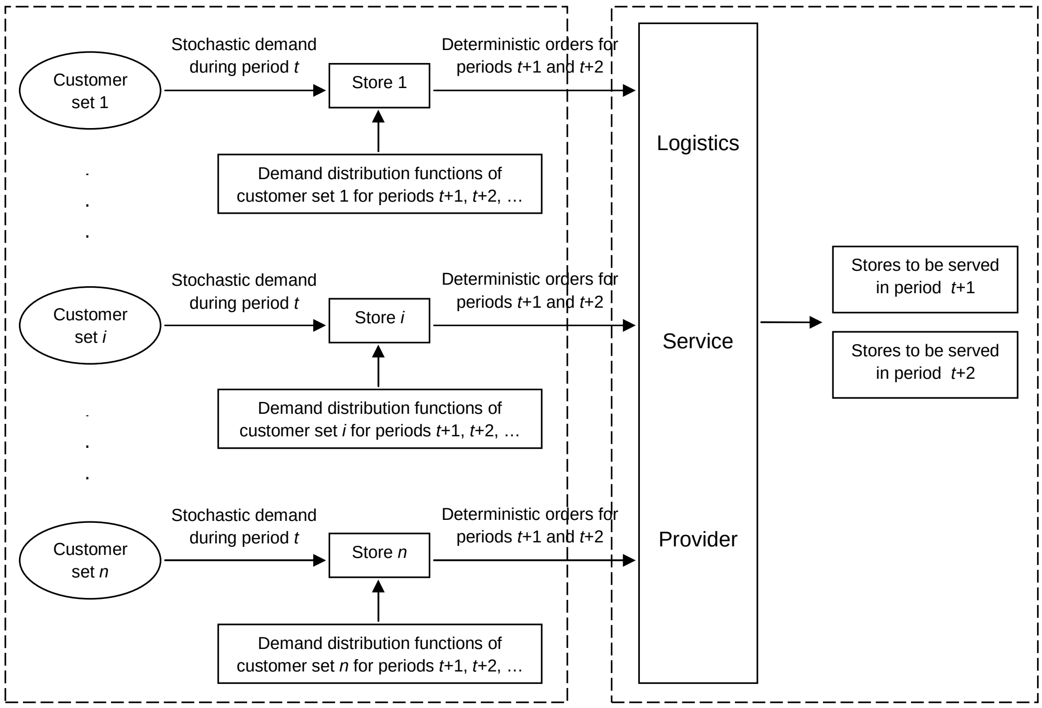

Figure 1 The big picture.

Every store i has its own inventory control system whereby at the end of period t it individually calculates the optimal orders to be placed for periods $t+1$ and $t+2$. Calculation of such deterministic orders could be based on the current inventory level in store i, the demand distribution functions of the final customers in periods $t+1$ onward, and other relevant parameters. But it does not explicitly take into account the global routing costs. The stores have long-term contracts with the LSP. The contracts bind the LSP to serve them for some pre-agreed annual payment. Although not relevant to our model, the LSP may once have calculated a fixed cost-to-serve for each store (Ozener et al. 2013) based on the average magnitude of the orders from that store, its distance from the warehouse, and how isolated it is. Then, it is able to apply it to the terms of the contract. In its day-to-day operations, the LSP focuses on only two periods. At the end of period t, the LSP receives the order sizes of the stores for periods $t+1$ and $t+2$, decides about each order whether it should be satisfied/postponed/advanced, executes its decision for period $t+1$, and waits until the end of period $t+1$ when orders for periods $t+2$ and $t+3$ are placed by the stores. In other words, a rolling horizon of two periods is considered where the decision for period $t+1$ is executed but the decision for period $t+2$ may undergo changes. One reason to consider such a short rolling horizon is that demands from the final customers to each store are stochastic. As a result, the orders for period $t+2$ placed at the end of period t are estimations. Therefore, each store prefers to observe its real inventory level at the end of period $t+1$ and update its initial order for period $t+2$. Moreover, the estimated orders are less reliable for farther periods, especially when stochasticity of the demands from the final customers is very high. From a store's point of view, advancement is tantamount to holding unnecessary inventory, and postponement potentially yields lost sales and low service levels. Therefore, advancement or postponement by only one period is justifiable when the holding costs are significant and the stores are committed to provide very high service levels for their customers. A typical application with the aforementioned characteristics is inventory control of fresh products in supermarkets, where the products rapidly lose their quality and the stores aim to provide a very high service level. As a result, advancement and postponement by more than one period is undesirable from the stores' point of view. Van Donselaar et al. (2006) conduct an analysis of two Dutch supermarket chains and report that the average delivery frequency of fresh products to each store is 1.2 days. This is consistent with the results of our interview with the supply chain manager of a Belgian supermarket chain, who confirmed that most of the stores are served every day or every other day.

By confining ourselves to two periods, the big picture presented in Figure 1 can be decomposed into n independent inventory control problems on the left side and a 2-period VRP on the right side. This paper is only dedicated to model and solve the latter problem. Hence, we consider the deterministic orders placed by each store for periods $t+1$ and $t+2$, and build our model from the LSP's perspective. For the sake of simplicity in our notations, we denote periods $t+1$ and $t+2$ by 1 and 2. Without loss of generality, we assume that all the stores place an order every period or every other period. Therefore, by considering two periods, each store has a positive order for period 1, or period 2, or both. Although our 2-period model focuses on one aspect of the broader DMPVRP, it can be exploited to solve the DMPVRP over a rolling horizon.

1.2. Additional discussion

In order to put our problem in a more formal framework, consider a 2-period VRP where deterministic orders of stores are known for two periods. Unlimited supply of the product exists in a warehouse (depot) where the LSP loads a number of vehicles in each period in order to deliver it to the stores. We also assume that the LSP has access to an unlimited number of homogeneous capacitated vehicles. If the LSP has to satisfy each order in its associated period, then our 2-period VRP simplifies into two independent VRPs. But, as we stated before, orders for period 1 (period 2) could be postponed (advanced), and the LSP may benefit from a decrease in the routing costs in two periods by postponing and/or advancing a set of orders. A crucial feature of our model is that the magnitude of a postponed/advanced demand can change. To explain this feature, let assume that the order sizes in periods 1 and 2 for some store i are 4 and 0, respectively. Based on its inventory control system, in case its initial order size for period 1 is postponed, store i may have an order size different from 4, e.g., 2 or 5, for period 2. This is justifiable, in particular, if the demand of period 1 cannot be backlogged.

We assume that each order in period 1 (period 2) must be completely satisfied in that period or completely be postponed (advanced) to the other period. There is an apriori defined penalty associated with each postponing and advancing, which could depend on the magnitude of the order. Figure 2 shows a solution of the 2-period VRP where orders of the stores are satisfied in their associated periods, i.e., neither postponement nor advancement take place. In this figure, each store is represented by a circle, and the depot is depicted by a triangle. The quantity above each vertex (store) represents its initial order, and the capacity of each vehicle is taken to be 10.

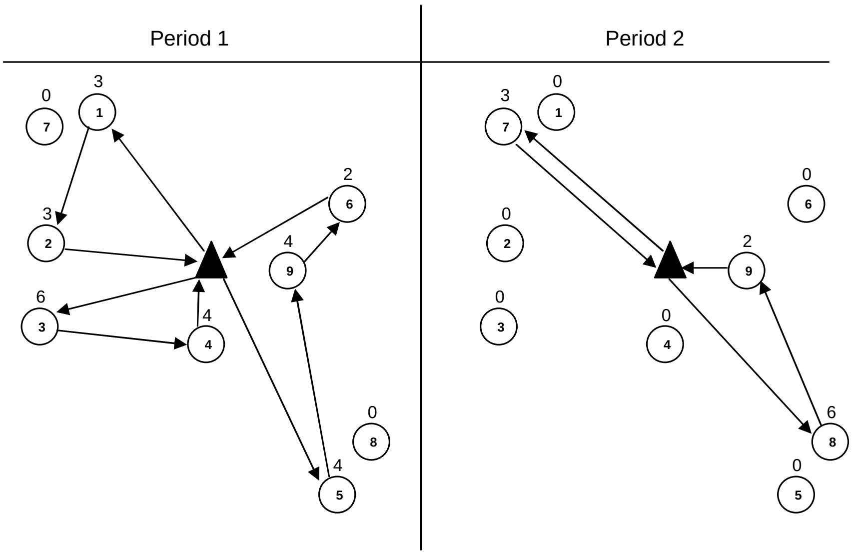

Figure 2 Optimal routes when postponing and advancing are not allowed.

Visually, it is easy to see that, in Figure 2, we can decrease the routing costs in period 1 by postponing the order of store 5, while just slightly adding to the routing costs in period 2 by inserting store 5 into route $0-8-9-0$. Note that in this case, the quantity requested by store 5 in period 2 is 2 units only according to Figure 3, as opposed to 4 units in case the same store is served in period 1. Simultaneously, we can further decrease the routing costs in period 2 by advancing the order of store 7 and serve it in period 1 through route $0-1-7-2-0$. In this case, store 7 may request a different quantity than 3 units, e.g., 4 units according to Figure 3. The new solution where demands of stores 5 and 7 are postponed and advanced, respectively, is shown in Figure 3. In this figure, we have to take into consideration two penalties; the penalty of postponing the order of store 5 and the penalty of advancing the order of store 7. Whether the new solution in Figure 3 is better than the solution in Figure 2 depends on how much the LSP saves in the routing costs and how much it has to pay for the penalties.

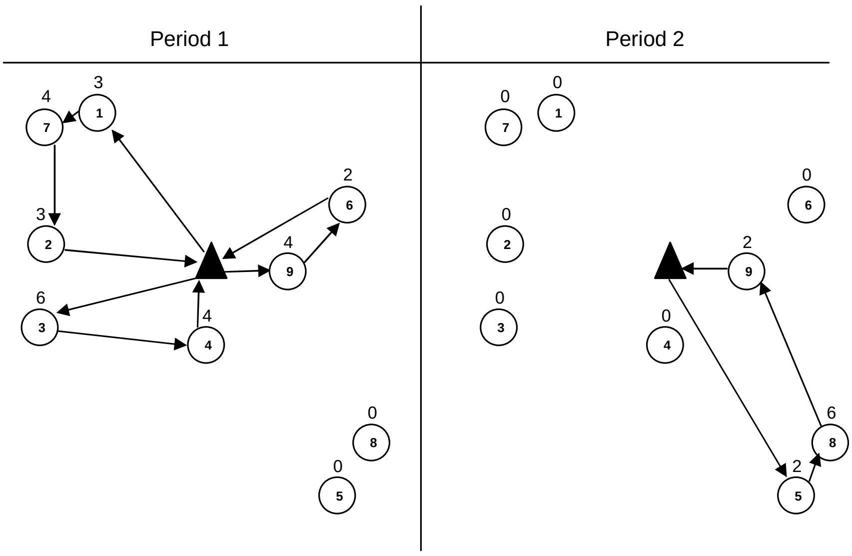

Figure 3 Optimal routes when postponing and advancing are allowed.

We assume that each order is significantly smaller than the vehicle capacity. Otherwise, direct shipping could be the most efficient delivery policy for this store, which could be separated from the others. Indeed, Gallego and Simchi-Levi (1990) conclude that, with a very high probability, direct shipping is preferable over all routing strategies provided that the economic lot size is a large fraction of the vehicle capacity; see also Bertazzi (2008).

1.3. Scientific contributions

We formulate the 2-period VRP as an Integer Linear Programming (ILP) problem and solve it by a branch-and-price algorithm to optimality. The main contributions of this paper can be summarized as follows.

- Unlike the existing deterministic models in the MPVRP literature, our model can deal with the case where the sum of the orders for two periods is not a fixed number. We do not discuss partial postponing/advancing, although it may be regarded as a natural variant of our problem.
- The existing models to solve the MPVRP to optimality use arc formulations and can deal with more than two periods. However, they become nonlinear in the presence of time windows, even for the 1-period VRP. In contrast, we use route formulations and bring the nonlinear constraints induced by the time windows to the pricing problem, so that our master problem remains linear and can be solved to optimality.
- We draw algorithmic insights by solving an ILP model with a restricted number of generated columns in each node, and we analyze the trade-off between the computational time and the optimality gap.
- We draw managerial insights on cost improvements based on the results obtained from the test instances.

The paper is organized as follows. Section 2 contains a brief literature review. The problem formulation, including the master problem and two pricing sub-problems, is presented in Section 3. We deal with details of implementation issues in Section 4. Computational results including algorithmic and managerial insights are presented in Secion 5, and finally conclusions are drawn in Section 6. We succinctly discuss a generalized model in the appendix.

2. Literature review

A first related topic is the Inventory Routing Problem (IRP), where there exists a central system making decisions about the delivery quantities as well as the routes in each period so that the total cost/profit of the network is optimized while the stores are not allowed to run out of stock. So, in the IRP, demand from the final customers is centrally managed, say, by the LSP. Obviously, the no-stockout constraint only makes sense in case the final demand is deterministic. Otherwise, when demand is stochastic, the no-stockout constraint should be replaced by a constraint on the service level. Interested readers are referred to Andersson et al. (2010) for a comprehensive literature survey on IRPs, where existing works are classified based on deterministic/stochastic demands from final customers, the planning horizon, and other relevant features. Dealing with an IRP, deterministic or stochastic, is daunting enough that, in order to model and solve it, many researchers have resorted to simplifying assumptions such as constant demand by each store (Raa and Aghezaaf 2008, 2009), consideration of a single vehicle (Archetti et al. 2007), and delivery policies (Bertazzi et al. 2002). Not surprisingly, even under these assumptions, solving an IRP is not a trivial task. This is why researchers have resorted to heuristic methods, predominantly local search algorithms (Bertazzi et al. 2002) and decomposition approaches (Campbell and Savelsbergh 2004), to find a good solution for the problem.

The Multi-Period Vehicle Routing Problem (MPVRP) is the second area related to our work. Here, unlike the IRP, the demand of the final customers is managed by the stores. Each store places its orders for some periods, and an LSP is responsible to decide about the routings. Depending on the problem statement, the orders placed by the stores could be deterministic or stochastic. In most of the papers on MPVRP, the LSP is committed to schedule the delivery quantities to each store during the planning horizon in such a way that no backlogging of the orders occurs. In other words, the LSP is usually allowed to advance the orders but postponing is not allowed Bertazzi and Speranza. The LSP may be charged some penalty for advancing deliveries, but has no further responsibility about the demand of the final customers. MPVRP formulations which do not fully respect this classic definition exist in the literature, and our problem could be regarded as one of them. Similar MPVRPs are defined by Wen et al. (2010) and Albareda-Sambola et al. (2014), where at the end of each period exact information about the orders placed in that period and earlier ones is available, but little information about the orders of the following periods is at hand. In their setting, each order is indexed by a due date and must be served before it. Therefore, we can say that the existing orders can be advanced or postponed without penalty. Wen et al. (2010) formulate the problem as an integer linear programming problem and develop a heuristic method to solve it. Albareda-Sambola et al. (2014) develop a formula to measure the approximate profit of serving each store in the current period. Then, they formulate a VRP with the objective function consisting of the profit collection as well as the routing costs, in order to decide which stores should be served in the current period. Angelelli et al. (2007a) and Angelelli et al. (2007b) tackle a MPVRP where in each period a set of orders appear which must be served either in that period or in the next period. In both works, a single uncapacitated vehicle is used, postponement is treated without being penalized, and there is no information about the upcoming orders in the next period; hence, advancing does not make sense. Angelelli et al. (2009) analyze a similar problem where each order could be postponable or not. Hence, unpostponable orders must be served in period 1, whereas postponable orders can be served either in period 1 or period 2. The authors consider a limited number of uncapacitated vehicles and develop a heuristic method to solve the problem. They do not penalize postponement and do not consider advancement.

The big picture represented in Figure 1 could be regarded as an infinite planning horizon IRP with stochastic demands if demands of the final customers were managed by the LSP. However, as mentioned in Section 1, we assume that each store has its own inventory control system and determines its orders based on the demands from the final customers. Thus, we have a DMPVRP from the LSP's perspective. We model and solve only a chunk of this problem, i.e., a 2-period VRP. In this sense, our problem is a MPVRP with deterministic orders, as the LSP plays no role in determining the order sizes. In other words, in a classic IRP, authors usually assume that the customer demand is forecasted by the stores, whereas the supplier takes care of managing their inventories. In our model, we assume that the stores determine their optimal orders, but leave some flexibility to the LSP to choose its optimal delivery period. So, the LSP does not really manage the inventory in our model, and its focus is only on transportation costs.

Although at first glance our problem might also be regarded as a specific deterministic MPVRP, the lion's share of the existing work in this domain either do not allow postponing at all (Coelho and Laporte 2013, Bertazzi and Speranza 2012) or, if they do, it is not penalized (Wen et al. 2010, Albareda-Sambola et al. 2014, Angelelli et al. 2007a,b, 2009). An exception is the work by Abdelmaguid and Dessouky (2006) in which they model and solve a deterministic MRVRP, where orders can be advanced or postponed (both penalized), but the total delivery quantity to each store during the planning horizon is fixed (unlike our model). They do not consider time windows and solve their model by a genetic algorithm.

3. Problem formulation

In the development of our model, we use the following notations. For the sake of reader's convenience, we will redefine some of them in the course of our discussion.

Table 1 Indices and sets

i,j	indices for vertices (stores)
r	index for routes
V_I	set of stores with a positive order for period 1 and no order for period 2
V_{II}	set of stores with no order for period 1 and a positive order for period 2
V_{III}	set of stores with positive orders for both periods
$V_{III}^{'}$	set of virtual stores associated with the real stores in V_{III}
V	$V_I \cup V_{II} \cup V_{III} \cup V_{III}^{'}$
V^+	$V \cup \{0\}$ where vertex 0 denotes the depot
A	set of arcs
R_1	set of feasible routes in period 1
R_2	set of feasible routes in period 2

Consider a graph $G = (V^+, A)$ where vertices represent the depot (denoted by 0) and the stores, and arcs represent transportation links. Products are picked up from the depot and delivered to the stores. Each route starts and ends at the depot. The total transportation costs over two periods, including the penalties, should be minimized. The LSP has access to an unlimited homogeneous fleet with a given capacity for each vehicle. When it is used, each vehicle incurs a fixed cost per period f. It also incurs a variable cost equal to c_{ij} when it traverses the arc (i,j). Each vehicle can perform at most one single route per period within a limited time (time window of the depot). Split deliveries within a period are not allowed, i.e., in each period each store is served by at most one vehicle.

Table 2 Parameters

d_{i1}	order of store i for period 1
d_{i2}	order of store i for period 2
$d_{i1}^{'}$	order of store i for period 1 if it is not served in period 2, i.e., if its order is advanced
$d_{i2}^{'}$	order of store i for period 2 if it is not served in period 1, i.e., if its order is postponed
Δ_i	postponement penalty imposed by store i
$\Delta_i^{'}$	advancement penalty imposed by store i
n_1	number of stores in set V_I
n_2	number of stores in set V_{II}
n_3	number of stores in set V_{III}
n	total number of stores including virtual stores $(n = n_1 + n_2 + 2n_3)$
c_{ij}	cost of using arc (i,j)
f	fixed cost of using a vehicle per period
α_{ir}	1 if store i belongs to route r; 0 otherwise.
g_i	1 if order of store i can be postponed; 0 otherwise.
$g_i^{'}$	1 if order of store i can be advanced; 0 otherwise.

Table 3 Decision variables

u_{r1}	1 if route $r \in R_1$ is used in period 1; 0 otherwise.
u_{r2}	1 if route $r \in R_2$ is used in period 2; 0 otherwise.

Define d_{i1} and d_{i2} as the orders of store i in periods 1 and 2, respectively. Without loss of generality, we assume that each store has a positive order at least in one period. If we look at Figure 2, we can distinguish three classes of stores. Any store i in class V_I has a positive order d_{i1} for period 1 but a zero order for period 2. If the LSP decides not to deliver to store i in period 1 (postponing), then it must deliver $d_{i2}^{'} > 0$ in period 2, where $d_{i2}^{'}$ could be smaller than, equal to, or greater than d_{i1}. Indeed, the decision to deliver zero in period 1 and $d_{i2}^{'}$ in period 2 is an alternative decision. However, the LSP is charged a penalty Δ_i for making this alternative decision (postponing). Similarly, a store i in class V_{II} has no order for period 1 but a positive order d_{i2} for period 2. The alternative decision (advancing) for the LSP is to deliver quantities $d_{i1}^{'} > 0$ and zero in periods 1 and 2, respectively. Here again, $d_{i1}^{'}$ could be different from d_{i2}, and the LSP is charged a penalty $\Delta_i^{'}$ for advancing the order. Finally, class V_{III} includes stores with positive orders d_{i1} and d_{i2} for both periods. In this class, two alternative decisions can be made for each store. The first alternative decision (postponing) is to deliver quantity zero and $d_{i2}^{'} > 0$ in periods 1 and 2, respectively. The second alternative decision (advancing) is to deliver $d_{i1}^{'} > 0$ and zero in periods 1 and 2, respectively. Neither $d_{i1}^{'}$ nor $d_{i2}^{'}$ needs to be equal to $d_{i1} + d_{i2}$. Penalties Δ_i and $\Delta_i^{'}$ are considered for postponing and advancing, respectively. Table 4 shows all three classes and the possible decisions regarding delivery quantities for each class.

If some store $i \in V_I$ is served in period 1, it implies that the initial decision is made for this store and the delivery quantities to this store in periods 1 and 2 are d_{i1} and zero, respectively. On the other hand, if the LSP decides to serve store $i \in V_I$ in period 2, then it has made the

Table 4 Three different classees of stores

Class	Initial decision			Alternative dec. 1			Alternative dec. 2		
	del.$_1$	del.$_2$	pen.	del.$_1$	del.$_2$	pen.	del.$_1$	del.$_2$	pen.
V_I	d_{i1}	0	0	0	$d^{'}_{i2}$	Δ_i	—	—	—
V_{II}	0	d_{i2}	0	$d^{'}_{i1}$	0	$\Delta^{'}_i$	—	—	—
V_{III}	d_{i1}	d_{i2}	0	0	$d^{'}_{i2}$	Δ_i	$d^{'}_{i1}$	0	$\Delta^{'}_i$

alternative decision with delivery quantities zero and $d^{'}_{i2}$. A similar reasoning applied for stores $i \in V_{II}$. However, this reasoning fails for a store $i \in V_{III}$. Indeed, if store $i \in V_{III}$ is served in period 1, the size of the delivery quantity to this store in period 1 is not immediately known. Nor is the size of the delivery quantity to it in period 2. It only appears, from Table 4, that the LSP has made either the initial decision with delivery quantities d_{i1} and d_{i2} in periods 1 and 2, or the second alternative decision with delivery quantities $d^{'}_{i1}$ and zero in periods 1 and 2. Similarly, if some store $i \in V_{III}$ is served in period 2, then it implies that the LSP has made either the initial decision with delivery quantities d_{i1} and d_{i2} in periods 1 and 2, or the first alternative decision with delivery quantities zero and $d^{'}_{i2}$ in periods 1 and 2. As we consider capacitated vehicles in our model, in either period we have to know one specific delivery quantity to each store, and consequently determine the vehicle load for each route. But, such specific values are not known for stores $i \in V_{III}$.

To resolve this ambiguity, we assume from now on that if store $i \in V_{III}$ is visited in period 1 (respectively, period 2), then its delivery quantity in period 1 (respectively, period 2) is d_{i1} (respectively, d_{i2}). Furthermore,, we define a virtual store $i + n_3$ corresponding to store $i \in V_{III}$ with orders $(d^{'}_{i1} - d_{i1})$ and $(d^{'}_{i2} - d_{i2})$ for periods 1 and 2, respectively; these quantities could be negative. The cost of going from store $i \in V_{III}$ to its corresponding virtual store j is zero, i.e. $c_{j-n_3,j} = 0$ for every $j \in V^{'}_{III}$. More generally, the costs of the links from real stores to virtual stores are represented by Equation (1). The costs from virtual stores to real stores are represented by Equation (2).

$$c_{ij} = \begin{cases} 0 & j \in V^{'}_{III},\ i = j - n_3, \\ \infty & j \in V^{'}_{III},\ i \neq j - n_3. \end{cases} \tag{1}$$

$$c_{ij} = c_{i-n_3,j} \quad i \in V^{'}_{III},\ j \in V_I \cup V_{II} \cup V_{III} \tag{2}$$

According to our settings, a store $i \in V_I \cup V_{II}$ must be served in either period 1 or period 2, a store $i \in V_{III}$ can be served in period 1 or period 2 or both periods, and a virtual store $i \in V^{'}_{III}$ can be served in period 1 or period 2 or neither period. Moreover, a virtual store $i \in V^{'}_{III}$ can be served in period 1 (respectively, period 2) only if its associated real store $i \in V_{III}$ is served in period 1 (respectively, period 2), by definition of the cost coefficients in Equations (1) and (2). Serving a virtual store in period 1 implies that advancing has happened, i.e., the second alternative decision is made. In this case, since in any solution a virtual store $i+n_3$ can only succeed its associated real store $i \in V_{III}$, it also implies that the actual delivery quantity to the real store in period 1 is $d_{i1}+(d'_{i1}-d_{i1})=d'_{i1}$. On the other hand, if a virtual store is served in period 2, then postponing has happened, i.e., the first alternative decision is made. In this case, the only observation that matters is that when a vehicle visits stores $i \in V_{III}$ and $i+n_3$, it must carry the total delivery quantity $d_{i2}+(d'_{i2}-d_{i2})=d'_{i2}$ to the real store in period 2. If a virtual store is served in neither period, then the initial decision is made by the LSP. We can also say that if a real store is served in period 1(2), then its associated virtual store in period 2(1) must not be served because postponing(advancing) has not happened. The exploitation of virtual stores and the aforementioned setting leads to a model where the delivery quantity to any store $i \in V$ in a given period is known when a route visits i in this period.

The role of parameters g_i and g'_i is to enforce the model to serve any store i in its requested period, should postponing or advancing its order not be possible due, for instance, to its desired service level or to its limited capacity. Based on such restrictions, the LSP may be obliged to satisfy the orders of some stores exactly in the requested period.

3.1. Master problem

Our problem is obviously NP-hard since it generalizes the VRP. We can formulate it as an ILP problem where the decision variables correspond to feasible routes. By defining R_1 and R_2 as the set of feasible routes in periods 1 and 2, the ILP problem is formulated as follows.

$$\begin{aligned}\min \sum_{r\in R_1}(f+\sum_{(i,j)\in r} c_{ij})u_{r1}+\sum_{r\in R_2}(f+\sum_{(i,j)\in r} c_{ij})u_{r2}\\ +\sum_{i\in V_I}\Delta_i(\sum_{r\in R_2}\alpha_{ir}u_{r2})+\sum_{i\in V_{II}}\Delta'_i(\sum_{r\in R_1}\alpha_{ir}u_{r1})\\ +\sum_{i\in V_{III}}\Delta_i(\sum_{r\in R_2}\alpha_{i+n_3,r}u_{r2})+\sum_{i\in V_{III}}\Delta'_i(\sum_{r\in R_1}\alpha_{i+n_3,r}u_{r1})\end{aligned} \tag{3}$$

subject to

$$\sum_{r\in R_1}\alpha_{ir}u_{r1}+\sum_{r\in R_2}\alpha_{ir}u_{r2}=1;\ \forall i\in V_I,\ \text{(dual variable: } \beta) \tag{4}$$

$$\sum_{r\in R_1}\alpha_{ir}u_{r1}+\sum_{r\in R_2}\alpha_{ir}u_{r2}=1;\ \forall i\in V_{II},\ \text{(dual variable: } \gamma) \tag{5}$$

$$\sum_{r\in R_1}\alpha_{ir}u_{r1}+\sum_{r\in R_2}\alpha_{i+n_3,r}u_{r2}=1;\ \forall i\in V_{III},\ \text{(dual variable: } \lambda) \tag{6}$$

$$\sum_{r\in R_1}\alpha_{i+n_3,r}u_{r1}+\sum_{r\in R_2}\alpha_{ir}u_{r2}=1;\ \forall i\in V_{III},\ \text{(dual variable: }\mu) \tag{7}$$

$$\sum_{r\in R_1}\alpha_{ir}u_{r1}\geq 1-g_i;\ \forall i\in V_I\cup V_{III},\ \text{(dual variable: }\theta) \tag{8}$$

$$\sum_{r\in R_2}\alpha_{ir}u_{r2}\geq 1-g_i^{'};\ \forall i\in V_{II}\cup V_{III},\ \text{(dual variable: }\pi) \tag{9}$$

$$u_{r1}\in\{0,1\};\ \forall r\in R_1 \tag{10}$$

$$u_{r2}\in\{0,1\};\ \forall r\in R_2 \tag{11}$$

The objective function (3) consists of fixed and variable costs of each route in both periods, postponement penalty for any store in class V_I if a selected route in period 2 includes it, advancement penalty for any store in class V_{II} if it is included in a selected route in period 1, postponement penalty for any store in class V_{III} if a selected route includes its associated virtual store in period 2, and advancement penalty for any store in class V_{III} if its associated virtual store is included in a selected route in period 1. Constraints (4) guarantee that every store in V_I is served either in period 1 or in period 2. Constraints (5) do the same for stores in V_{II}. Constraints (6) impose that if any store in class V_{III} is served in period 1, then its associated virtual store is not served in period 2 (postponement has not happened), and conversely. Constraints (7) are interpreted in the same way by considering advancement, i.e., if a store in V_{III} is served in period 2, then its associated virtual store is not served in period 1 (advancement has not happened), and conversely. Constraints (8) express that if $g_i=0$, then store i must be served in period 1. Similarly, constraints (9) express that the order of store i cannot be advanced when $g_i^{'}=0$. Finally, constraints (10)-(11) define the binary variables.

To shed further light on the concept of the virtual stores, suppose that we select a solution of model (3)-(11) such that $\alpha_{i+n_3,s}u_{s1}=1$ for some route $s\in R_1$, i.e., $\sum_{r\in R_1}\alpha_{i+n_3,r}u_{r1}=1$. Since route s necessarily includes store i, it must be the case that $\alpha_{is}u_{s1}=1$. In view of constraint (6), this implies in turn that $\alpha_{i+n_3,r}u_{r2}=0$ for all routes $r\in R_2$, i.e., $\sum_{r\in R_2}\alpha_{i+n_3,r}u_{r2}=0$. Moreover, in view of constraint (7), we also find that $\alpha_{ir}u_{r2}=0$ for all routes $r\in R_2$, i.e., $\sum_{r\in R_2}\alpha_{ir}u_{r2}=0$. The interpretation is that, if some route includes $i+n_3$ in period 1 (meaning that the quantity $d_{i1}^{'}$ is delivered to store i in period 1), then necessarily vertices i and $i+n_3$ are not included in any route in period 2 (meaning that no delivery is made to store i in period 2). Symmetrically, it can be checked that if the quantity $d_{i2}^{'}$ is delivered to store i in period 2, then no delivery to this store can take place in period 1. This shows that any feasible solution of model (3)-(11) is consistent with the definition of sets V_{III} and V'_{III}.

The LP relaxation of problem (3)-(11) is viewed as a master problem that can be solved by column generation (Lubbecke and Desrosiers 2005, Dabia et al. 2013). In the next sub-section, we formulate two pricing subproblems to generate feasible routes in periods 1 and 2. The pricing problems are Elementary Shortest Path Problems with Resource Constraints (ESPPRC). We do not need to consider constraints (8)-(9) in the master problem provided that we respect them during column generation: If $g_i = 0$ for some store $i \in V_I$ (respectively, $i \in V_{III}$), we never generate any route in period 2 that includes store i (respectively, store $i + n_3$). Similarly, if $g'_i = 0$ for some store $i \in V_{II}$ (respectively, $i \in V_{III}$), then we never generate any route in period 1 that includes store i (respectively, $i + n_3$). By following this strategy, we neither need to consider constraints (8)-(9) in the master problem nor do we need them in the pricing problems.

3.2. Pricing problems

In order to solve the master problem, we set a pricing problem for each period as an ESPPRC (Irnich and Desaulniers 2005). Each solution in the ESPPRC is a route which starts and ends at the depot while including a subset of the vertices, and respecting the side constraints related to the vehicle load and time windows. The settings are done in such a way that the cost of a route (solution) in the ESPPRC is equivalent to the reduced cost of the same route in the master problem. Feasibility of a route in the ESPPRC implies that it is feasible in the master problem, too. Each solution of the ESPPRC in period 1 or 2 with a negative cost (not necessarily the optimal solution) can be introduced in the master problem in the next iteration. We stop when neither the ESPPRC in period 1 nor the ESPPRC in period 2 is able to identify any route with a negative cost.

Based on the dual prices obtained in the optimal solution of the restricted master problem, the reduced cost of a route r in period 1 or 2 is calculated as $(f + \sum_{(i,j)\in r} c'_{ij})$, where the cost coefficients c'_{ij}'s for periods 1 and 2 are calculated based on Equations (12) and (13), respectively, with $\Delta_0 = 0$, and $\beta_0 = 0$.

$$c'_{ij} = \begin{cases} c_{ij} - \beta_j; & \forall i \in V^+,\ j \in V_I^+ \backslash \{i\} \\ c_{ij} + \Delta'_j - \gamma_j; & \forall i \in V^+,\ j \in V_{II} \backslash \{i\} \\ c_{ij} - \lambda_j; & \forall i \in V^+,\ j \in V_{III} \backslash \{i\} \\ c_{ij} + \Delta'_{j-n_3} - \mu_{j-n_3}; & \forall i \in V^+,\ j \in V'_{III} \backslash \{i\} \end{cases} \tag{12}$$

$$c'_{ij} = \begin{cases} c_{ij} + \Delta_j - \beta_j; & \forall i \in V^+,\ j \in V_I^+ \backslash \{i\} \\ c_{ij} - \gamma_j; & \forall i \in V^+,\ j \in V_{II} \backslash \{i\} \\ c_{ij} - \mu_j; & \forall i \in V^+,\ j \in V_{III} \backslash \{i\} \\ c_{ij} + \Delta'_{j-n_3} - \lambda_{j-n_3}; & \forall i \in V^+,\ j \in V'_{III} \backslash \{i\} \end{cases} \tag{13}$$

By using the c'_{ij}'s in periods 1 and 2, we set an independent network for each period. In each network, we seek the feasible routes with negative costs, i.e., the feasible routes which have negative reduced costs in the master problem. A route is feasible if (1) the total delivery quantity in the route does not exceed the vehicle capacity, (2) it respects the time windows, and (3) it starts and ends at the depot and visits each vertex at most once (elementarity).

3.3. The label-setting algorithm

The label-setting algorithm is used to identify new routes with negative reduced costs in each iteration of the column generation. As long as $\min(f + \sum_{(i,j)\in r} c'_{ij})$ in either period is negative there exists a route which is able to potentially improve the objective function of the master problem. Note that if no route with a negative reduced cost in period 1 is found in iteration k, but the ESPPRC for period 2 introduces a new route with a negative reduced cost, in the next iteration we have to solve again the ESPPRC for both periods. In other words, we have to solve two ESPPRCs in each iteration regardless of whether we succeed to find any route in period 1 or 2 in the previous iteration. The stopping condition is that we do not find any route with a negative reduced cost neither for period 1 nor for period 2.

In the label-setting algorithm, each route is represented by a multi-dimensional label which is further expanded to create bigger routes. In order to exploit a label-setting algorithm, we define some parameters and decision variables represented in Tables 5 and 6, respectively.

Table 5 Parameters of the label-setting algorithm

Q	capacity of each vehicle
t_{ij}	travel time to traverse arc (i,j)
s_i	service time in store i
(e_i, l_i)	time window to visit store i

Table 6 Decision variables of the label-setting algorithm

z_i	1 if store i is included in the generated route; 0 otherwise.
v_{ij}	1 if arc (i,j) in used in the generated route; 0 otherwise.
T_i	start time of service at store i
P_{ij}	load on the vehicle when it traverses arc (i,j)

The first resource constraint which must be taken into account is the vehicle capacity Q. When generating a route, the load on the vehicle which is used in that route must not exceed the vehicle capacity. To this end, we must know the delivery quantity to each store that the vehicle visits. Table 7 summarizes the positive deliveries to each store in each period. As an example, if a route in period 2 includes a store $i \in V_I$ then the delivery quantity to this store must be d'_{i2}. If a solution

Table 7 **Positive delivery quantities in periods 1 and 2**

Class	del.$_1$	del.$_2$
V_I	d_{i1}	d'_{i2}
V_{II}	d'_{i1}	d_{i2}
V_{III}	d_{i1}	d_{i2}
V'_{III}	$(d'_{i-n_3,1}-d_{i-n_3,1})$	$(d'_{i-n_3,2}-d_{i-n_3,2})$

of the master problem includes no route containing $i \in V_I$ in period 2, this means that the delivery quantity in period 2 to this store is zero in this solution.

When a route includes arc (i,j), i.e., when $v_{ij}=1$, the vehicle load while traversing arc (i,j) must respect the vehicle capacity. This is expressed by inequality (14).

$$P_{ij} \le v_{ij} Q;\ \forall i,j \in V^+ \tag{14}$$

By taking Table 7 into consideration, the P_{ij}'s are calculated based on Equation (15), where $d_{01}=0$, if a route is generated in period 1. Similarly, the P_{ij}'s are calculated based on Equation (16), where $d'_{02}=0$, if a route is generated in period 2.

$$P_{ij} = \begin{cases} \sum_{j\in V^+} P_{ji} + d_{i1} z_i; & \forall i \in V^+,\ j \in V_I^+ \backslash \{i\} \\ \sum_{j\in V^+} P_{ji} + d'_{i1} z_i; & \forall i \in V^+,\ j \in V_{II} \backslash \{i\} \\ \sum_{j\in V^+} P_{ji} + d_{i1} z_i; & \forall i \in V^+,\ j \in V_{III} \backslash \{i\} \\ \sum_{j\in V^+} P_{ji} + (d'_{i-n_3,1} - d_{i-n_3,1}) z_i; & \forall i \in V^+,\ j \in V'_{III} \backslash \{i\} \end{cases} \tag{15}$$

$$P_{ij} = \begin{cases} \sum_{j\in V^+} P_{ji} + d'_{i2} z_i; & \forall i \in V^+,\ j \in V_I^+ \backslash \{i\} \\ \sum_{j\in V^+} P_{ji} + d_{i2} z_i; & \forall i \in V^+,\ j \in V_{II} \backslash \{i\} \\ \sum_{j\in V^+} P_{ji} + d_{i2} z_i; & \forall i \in V^+,\ j \in V_{III} \backslash \{i\} \\ \sum_{j\in V^+} P_{ji} + (d'_{i-n_3,2} - d_{i-n_3,2}) z_i; & \forall i \in V^+,\ j \in V_{III}^+ \backslash \{i\} \end{cases} \tag{16}$$

The second constraint is the service start time at each store, which must respect the associated time window, i.e.,

$$e_i \le T_i \le l_i;\ \forall i \in V^+ \tag{17}$$

We calculate T_i's based on Equation (18), where $s_0 = 0$:

$$T_i = \max_{j\in V^+}\{e_i, v_{ji}(T_j + s_j + t_{ji})\};\ \forall i \in V^+. \tag{18}$$

Finally, as the third constraint, the elementarity of each route must be respected. We do it by introducing a vector of resources $I = (1,1,\ldots,1)$. All elements of I are initially zero. The ith element of I is set to 1 when the route visits store i. This resource guarantees that in every generated route each store is visited at most once (Gutierrez-Jarpa et al. 2010, Irnich and Desaulniers 2005).

Since the label-setting algorithm is well-known in the literature (Irnich and Desaulniers 2005) we avoid further details. The classic version of the label-setting algorithm is not very efficient. In order to accelerate computations in this algorithm, some advanced concepts such as domination rules (Gutierrez-Jarpa et al. 2010), elementarity relaxation (Boland et al. 2006, Desaulniers et al. 2008, Righini and Salani 2008), and inaccessible vertices (Feillet et al. 2004) have been introduced. We use these techniques to improve the efficiency of our label-setting algorithm.

4. Implementation

In order to have an efficient branch-and-price algorithm in terms of computation time, we implement a number of different techniques. Such techniques are partly associated with the column generation and partly related to the branch-and-bound algorithm. In this section, we briefly discuss the techniques, but deal with route generation in more detail since it is not explicitly explained in the literature.

4.1. Lower bounding and upper bounding

During the course of branch-and-price we need to solve an LP relaxation via column generation in each node. In the root node we solve the LP relaxation of problem (3)-(11) to obtain an initial Global Lower Bound (GLB), which is progressively updated by setting it equal to the lowest LP relaxation value among all nodes which are not pruned. As in every minimization problem, the role of a GLB is to terminate branching when the value of a best feasible solution, i.e., the Global Upper Bound (GUB), is close enough to it. In addition to the GLB, we consider a Tentative Local Lower Bound (TLLB) in each node. By a node's TLLB we mean a lower bound on the LP relaxation value (and hence, on the IP value) in that node. The TLLB in each node is set to the optimal value of the objective function in its father node's LP relaxation. We exploit these TLLBs in the following way. When the LP relaxation is solved in a specific node through column generation, it may happen that, due to degeneracy, new routes with negative reduced costs can still be found even though the optimal value has been reached. The TLLBs help us to avoid such degenerate iterations in the nodes where the optimal value of the LP relaxation is equal to the TLLB. It turns out that such nodes are abundant.

We need a GUB during the branching procedure so that we can close any node in which the associated LP relaxation value exceeds the GUB. In order to obtain an initial GUB, we compute as follows a feasible integral solution of problem (3)-(11). We use the savings algorithm (Clarke and Wright 1964, Paessens 1988) to determine a feasible VRP solution for period 1 that includes all stores $i \in V_I \cup V_{III}$. Similarly, we use the algorithm to determine a feasible VRP solution for period 2 over all stores $i \in V_{II} \cup V_{III}$. These two independent VRP solutions yield a GUB which is updated whenever a better feasible solution is obtained. Besides the classic way to improve the

GUB when a better integer solution is found at the end of column generation in each node, we have tested two additional ideas to possibly improve the GUB. First, we record the objective function value for any integer solution we may find during the course of column generation, and we use it to improve the current GUB when possible. Note that in a node, we may encounter some integer solutions during the course of column generation which are better than the GUB at hand, while the optimal solution of the LP relaxation in this node is not integer. Investigating every solution to check whether it is integer takes some time, but it may improve the GUB and decrease the total computation time. Second, in each node we consider the formulation of the master problem obtained at the end of column generation, and we solve this formulation as an ILP problem, using CPlex. Since this ILP problem only contains a restricted subset of routes, its optimal solution provides a heuristic solution (and hence, an upper bound) for the complete formulation (3)-(11). Again, this is computationally time consuming, but may improve the upper bound and decrease the total computation time. We will describe the results of these tests in Section 5.

4.2. Branching

As suggested in the literature (Gutierrez-Jarpa et al. 2010), we branch on arcs even though decision variables in the master problem are routes. Indeed, fixing a route variable to zero complicates the solution process. As proposed by many authors, based on the values of routes in the master problem the value of arc (i,j) is calculated as follows:

$$x_{ij}^{(1)} = \sum_{r \in R_1 : (i,j) \in r} u_{r1} \tag{19}$$

$$x_{ij}^{(2)} = \sum_{r \in R_2 : (i,j) \in r} u_{r2} \tag{20}$$

When the optimal solution of the LP relaxation in some node is not integral we calculate $x_{ij}^{(1)}$ and $x_{ij}^{(2)}$ using Equations (19) and (20). Then, among all non-integral values $x_{ij}^{(1)},\ i,j \in V$, and $x_{ij}^{(2)},\ i,j \in V$, we branch on the variable with value closest to 1. During the course of branching we do not branch on any arc $(0,j)$, nor on any arc $(i,0)$, as we will automatically obtain integral values for these arcs at the end. Here, we discuss the relation between the arcs we have already branched on and the routes we should keep in the master problem. By setting an arc (i,j) to 0 in period 1 we must eliminate any route in this period that includes arc (i,j). However, setting an arc to 1 needs more work. Indeed, when we set arc (i,j) to 1, many other arcs can be set to 0 as a direct consequence. Depending on the class to which stores i and j belong we can specify the arcs to be set to 0.

Example

Consider Figure 2 where $V_I = \{1,2,3,4,5,6\}$, $V_{II} = \{7,8\}$, and $V_{III} = \{9\}$ and neglect the current routes. If somewhere in the branching tree we branch on arc $(3,9)$ in period 1 and set $x_{39}^{(1)} = 1$. Then, we can conclude that:

- $x_{93}^{(1)} = 0$ as we cannot simultaneously use arcs $(3,9)$ and $(9,3)$ in the same period in a solution,
- $x_{3j'}^{(1)} = 0,\ \forall j' \in V^+ \backslash \{9\}$ as the outgoing flow in period 1 from store 3 must be towards store 9,
- $x_{i'9}^{(1)} = 0,\ \forall i' \in V^+ \backslash \{3\}$ as the ingoing flow in period 1 to store 9 must be from store 3,
- $x_{3j'}^{(2)} = 0,\ \forall j' \in V^+$ and $x_{i'3}^{(2)} = 0,\ \forall i' \in V^+$ as store $3 \in V_I$ and when it is served in period 1, it cannot be served in period 2,
- $x_{99'}^{(2)} = 0$ and $x_{9'j'}^{(2)} = 0,\ \forall j' \in V^+$ as store $9 \in V_{III}$ and when it is served in period 1, its virtual store cannot be served in period 2 (postponing has not happened).

The necessary instructions in the general case are summarized in Tables 8 and 9. For example, suppose that we branch on arc (i,j) in period 1 and set $x_{ij}^{(1)}$ equal to 1, where $i,j \in V_{III}$. According to Table 8, actions 1-3,6,7,10,13,14 must be taken. These actions are represented in Table 9. Hence, in the corresponding child node we must have at least one route in period 1 that includes arc (i,j), i.e., action 1. Moreover, we must eliminate any route in period 1 which includes any of the arcs impacted by actions 2,3,10, and eliminate any route in period 2 which includes any of the arcs impacted by actions 6,7,13,14. Note that the instructions in Tables 8 and 9 pertain to branching on arc (i,j) in period 1. A similar procedure applies when the branching period is 2.

Table 8 Branching on arc (i,j) in period 1: $x_{ij}^{(1)} = 1$

	$j \in V_I \cup V_{II}$	$j \in V_{III}$	$j \in V'_{III}$
$i \in V_I \cup V_{II}$	1-5,10-12	1-5,10,13,14	—
$i \in V_{III}$	1-3,6,7,10-12	1-3,6,7,10,13,14	1-3,6,7,10,15,16
$i \in V'_{III}$	1-3,8,9,10-12	1-3,8,9,10,13,14	—

We have tested both a breadth-first strategy and a depth-first strategy to explore the nodes of the branching tree. There was no overall significant difference between the efficiency of these strategies in our test instances. Our computational results are based upon the breadth-first strategy.

4.3. Route generation

In each node, a set of initial routes should be introduced in the master problem to guarantee feasibility of the corresponding LP relaxation problem. Then, new routes are introduced in the master problem by the column generation procedure until we reach optimality of the LP relaxation. These new routes must be compatible with the status of the node in terms of the earlier branching decisions, where the status of a node consists of a set of arcs with value fixed to 1 and a set of arcs with value fixed to 0 for each period. The classic way in the literature to generate new routes in any node (Gutierrez-Jarpa et al. 2010) is considered as part of the label-setting algorithm, and

Table 9 Setting the values of different arcs

#	Action
1	$x^{(1)}_{ij} = 1$
2	$x^{(1)}_{ji} = 0$
3	$x^{(1)}_{ij'} = 0,\ \forall j' \in V^+ \backslash \{j\}$
4	$x^{(2)}_{i'i} = 0,\ \forall i' \in V^+$
5	$x^{(2)}_{ij'} = 0;\ \forall j' \in V^+$
6	$x^{(2)}_{i',i+n_3} = 0,\ \forall i' \in V^+$
7	$x^{(2)}_{i+n_3,j'} = 0,\ \forall j' \in V^+$
8	$x^{(2)}_{i',i-n_3} = 0,\ \forall i' \in V^+$
9	$x^{(2)}_{i-n_3,j'} = 0,\ \forall j' \in V^+$
10	$x^{(1)}_{i'j} = 0,\ \forall i' \in V^+ \backslash \{i\}$
11	$x^{(2)}_{i'j} = 0,\ \forall i' \in V^+$
12	$x^{(2)}_{jj'} = 0,\ \forall j' \in V^+$
13	$x^{(2)}_{i',j+n_3} = 0,\ \forall i' \in V^+$
14	$x^{(2)}_{j+n_3,j'} = 0,\ \forall j' \in V^+$
15	$x^{(2)}_{i',j-n_3} = 0,\ \forall i' \in V^+$
16	$x^{(2)}_{j-n_3,j'} = 0,\ \forall j' \in V^+$

works as follows. First, single-vertex paths from the depot, i.e., $0-i,\ \forall i \in V$, are built. Then, these paths are further extended to two-vertex paths, i.e., $0-i-j,\ \forall i,j \in V$. A path is discarded if it does not respect some of the resource constraints. Otherwise, its corresponding route is considered: the route that corresponds to path $0-i-j-k$, for example, is $0-i-j-k-0$. It can be added to the set of routes in the master problem provided that it respects all resource constraints, is compatible with the status of the node, and has a negative reduced cost. Unlike the classic way to generate new routes, we generate only those routes which are compatible with the status of the node. In the sequel, we explain in more detail how we provide an initial feasible solution for the master problem in each node and how we generate new compatible routes.

4.3.1. Initial routes must guarantee feasibility

Initial routes in any node are of two distinct types: (1) seed routes that guarantee feasibility, (2) auxiliary routes that provide an additional set of good routes.

In the root node:

—Seed routes for period 1 are $0-i-0$ for all $i \in V_I \cup V_{III}$ as well as for any $i \in V_{II}$ such that $g'_i = 1$. Seed routes for period 2 are $0-i-0$ for all $i \in V_{II} \cup V_{III}$ as well as for any $i \in V_I$ such that $g_i = 1$.

—Auxiliary routes are the routes determined by the savings algorithm.

In any child node where we set the value of a new arc equal to 0:

—Seed routes are exactly the same seed routes as in its father node.

—Auxiliary routes are all routes included in the master problem of its father node at the end of column generation, provided that they are compatible with the new branch. In other words, if we branch on arc (i,j) in period 1 in this child node and we set $x_{ij}^{(1)} = 0$, then those routes from the father node in period 1 that include arc (i,j) are discarded and the remaining ones are kept. In this example, all routes from the father node in period 2 are kept, too.

In any child node where we set the value of a new arc equal to 1:

—Seed routes are exactly the same seed routes as in its father node except that two seed routes merge in the corresponding period and one or two seed routes are discarded from the other period.

Example

Consider Figure 2 where $V_I = \{1,2,3,4,5,6\}$, $V_{II} = \{7,8\}$, and $V_{III} = \{9\}$ and neglect the current routes. Let us assume that postponing and advancing are allowed for all stores. The set of routes in the root node is $S_0 = \{0-1-0,\ 0-2-0,\ 0-3-0,\ 0-4-0,\ 0-5-0,\ 0-6-0,\ 0-7-0,\ 0-8-0,\ 0-9-0\}$ in period 1 and in period 2. If we branch on arc $(7,2)$ in period 1 and set its value to 1, i.e., $x_{72}^{(1)} = 1$, then the seed sets in periods 1 and 2 are $S_1 = \{0-1-0,\ 0-7-2-0,\ 0-3-0,\ 0-4-0,\ 0-5-0,\ 0-6-0,\ 0-8-0,\ 0-9-0\}$ and $S_2 = \{0-1-0,\ 0-3-0,\ 0-4-0,\ 0-5-0,\ 0-6-0,\ 0-8-0,\ 0-9-0\}$, respectively. We have discarded routes $0-2-0$ and $0-7-0$ from seed routes in period 2 because $x_{72}^{(1)} = 1$ implies that stores 7 and 2 are served in period 1; we know that $7 \in V_{II}$, $2 \in V_I$, and every $i \in V_I \cup V_{II}$ is served either in period 1 or in period 2, and not in both periods. Now, let us consider this newly created node as a father node with seed sets S_1 and S_2 in periods 1 and 2, respectively. If we branch on $x_{59}^{(2)} = 1$ the seed routes for the child node in periods 1 and 2 are $\{0-1-0,\ 0-7-2-0,\ 0-3-0,\ 0-4-0,\ 0-6-0,\ 0-8-0,\ 0-9-0\}$ and $\{0-1-0,\ 0-3-0,\ 0-4-0,\ 0-5-9-0,\ 0-6-0,\ 0-8-0\}$, respectively. Note that routes $0-5-0$ and $0-9-0$ in period 2 merge and route $0-5-0$ is discarded from the seed routes in period 1 because $5 \in V_I$ is served in one period only. However, route $0-9-0$ remains in the seed routes in period 1 because store $9 \in V_{III}$ can be served in both periods. Again, if we consider this new node as a father node and branch on $x_{99'}^{(2)} = 1$, the seed routes for the child node in periods 1 and 2 are $\{0-1-0,\ 0-7-2-0,\ 0-3-0,\ 0-4-0,\ 0-6-0,\ 0-8-0\}$ and $\{0-1-0,\ 0-3-0,\ 0-4-0,\ 0-5-9-9'-0,\ 0-6-0,\ 0-8-0\}$, respectively. In this child node, seed route $0-9-0$ is discarded from period 1 because serving store $9'$ in period 2 implies that the order of store 9 is postponed; so, it is not served in period 1 anymore.

—Auxiliary routes are all routes we eventually have in the master problem of the father node provided that they are compatible with the new branch. In other words, if we branch on arc (i,j) in period 1 in this child node and we set $x_{ij}^{(1)} = 1$, then based on Tables 8 and 9 the appropriate actions are taken to discard a set of routes and the remaining routes from the father node are kept. Clearly, all routes from the father node in period 2 are kept, too.

4.3.2. Every generated route must be compatible

Once we have correctly specified seed routes as well as auxiliary routes in a node we can solve the first restricted master problem in the node, obtain dual variables, and solve two ESPPRCs whereby we can generate new routes in each period. The interesting point is that all possible compatible routes can be created from the seed routes. Two seed routes $0-i-j-k-0$ and $0-i^{'}-j^{'}-0$ in period 1, for example, can be combined to build new routes $0-i-j-k-i^{'}-j^{'}-0$ and $0-i^{'}-j^{'}-i-j-k-0$. These seed routes imply that $x^{(1)}_{ij} = x^{(1)}_{jk} = x^{(1)}_{i^{'}j^{'}} = 1$ in the associated node. A new generated route is not compatible if it includes only part of the sequence $i-j-k$.

Example

In order to explain how all possible compatible routes are generated from the seed routes let us consider a node at level 8 of the branching tree; see Figure 4.

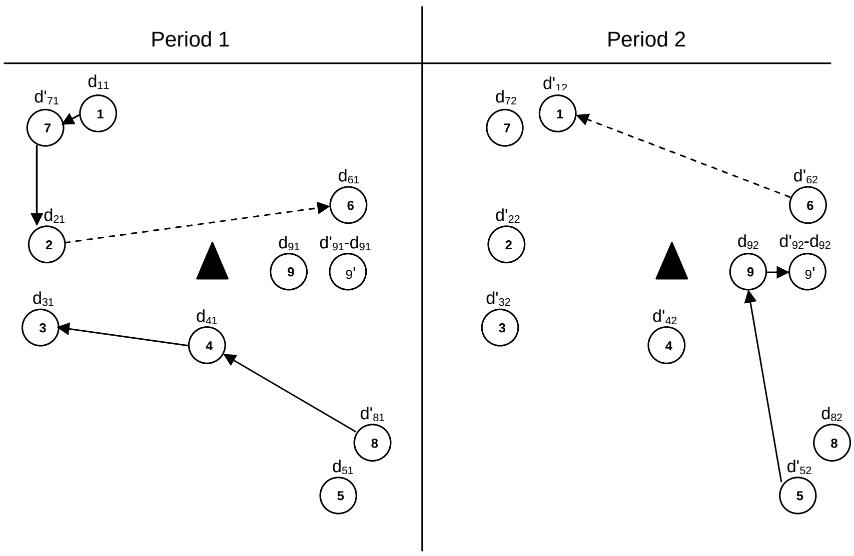

Figure 4 A node at level 8 of the branching tree.

Obviously, at this level we have already branched on 8 arcs, starting from the root node downstream to this node. Some of these arcs are set to 1, as represented with solid arcs in Figure 4, and the others are set to 0, as represented with dashed arcs. However, in this node, arcs $(2,6)$ in period 1 and $(6,1)$ in period 2 are not the only arcs with value forced to 0. As explained before, by setting an arc equal to 1 in period 1, for example, some other arcs in periods 1 and 2 must be set equal to 0. Figure 5 shows the status of this node in terms of arc values.

$$
\begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 9' \end{array}
\begin{array}{c}
\begin{array}{ccccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 9' \end{array} \\
\begin{pmatrix}
0 & - & 0 & 0 & 0 & 0 & - & 0 & - & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
- & 0 & 0 & 0 & 0 & 0 & 0 & 0 & - & 0 & 0 \\
- & - & 0 & 0 & 0 & 0 & - & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
- & - & 0 & 0 & 0 & 0 & 0 & 0 & - & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\end{array}
\qquad
\begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 9' \end{array}
\begin{array}{c}
\begin{array}{ccccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 9' \end{array} \\
\begin{pmatrix}
0 & 0 & 0 & 0 & 0 & - & - & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
- & 0 & 0 & 0 & 0 & - & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & - & 0 & 0 & 0 & 0
\end{pmatrix}
\end{array}
$$

Figure 5 Status of a node.

By considering the solid arcs and the fact that $V_I = \{1,2,3,4,5,6\}$, $V_{II} = \{7,8\}$, and $V_{III} = \{9\}$, it is not difficult to verify that the seed routes in periods 1 and 2 are $\{0-1-7-2-0,\ 0-8-4-3-0,\ 0-6-0\}$ and $\{0-5-9-9'-0,\ 0-6-0\}$, respectively. A compatible route in period 1 must not include any arc set to 0 in the left side matrix in Figure 5. All possible compatible routes in period 1 can be generated by combination of two or more seed routes in period 1, provided that connecting the conjunctive stores in each combination is admissible in terms of the arcs we have branched and fixed to 0, i.e., $x^{(1)}_{26}$. By this method we do not need to check that the generated route does not include any arc with value 0 in the left side matrix. In other words, if any combined route includes arc $(2,6)$, it is not admissible; otherwise it is. If it is not admissible or if the generated route is not feasible in terms of time windows or vehicle load we discard the generated route and continue with other combinations. So, the first group of generated routes are $\{0-1-7-2-8-4-3-0,\ 0-8-4-3-1-7-2-0,\ 0-8-4-3-6-0,\ 0-6-1-7-2-0,\ 0-6-8-4-3-0\}$. As we observe, combination of $\{0-1-7-2-0\}$ with $\{0-6-0\}$ and generating route $\{0-1-7-2-6-0\}$ is not admissible, in that connecting conjunctive stores 2 and 6 is not admissible. Then, we continue by combination of each generated route in this set with one further seed route which does not exist in its current combination until all seed routes are included in each generated route. Again, we need to check admissibility of connecting conjunctive vertices as well as feasibility of time windows and vehicle load. The second group of generated routes are $\{0-1-7-2-8-4-3-6-0,\ 0-8-4-3-6-1-7-2-0,\ 0-6-1-7-2-8-4-3-0,\ 0-6-8-4-3-1-7-2-0\}$. Again, $\{0-8-4-3-1-7-2-0\}$ cannot be combined with $\{0-6-0\}$ because connecting store 2 to store 6 is not allowed. These two groups of routes denote all possible compatible routes in period 1 to be considered in the ESPPRC. The compatible routes in period 2 are generated in the same way.

4.4. Using the last basis

During the course of column generation, when we add a new route (column) to the master problem in a node, we do not start solving the new master problem from scratch. We can start from the optimal basis of the master problem in the previous iteration, add the new column, and update the basis.

4.5. Column management

Classically, in each iteration of column generation we should independently solve a pricing problem for period 1 and a pricing problem for period 2. In each pricing problem the label-setting algorithm is used to find a route or a set of routes with negative reduced cost. In each period, we may stop expanding the routes in the algorithm once we obtain a route with a negative reduced cost. Alternatively, we could stop when we obtain a pre-specified number of routes with negative reduced costs or we could capture all of them. Our experience shows that in average the best performance of the column generation algorithm is obtained when we simultaneously introduce a big bunch of routes with negative reduced costs, e.g., 1000 routes. Besides adding new routes to the master problem, some authors consider deleting inefficient routes, in terms of not appearing in the optimal solution of the master problem during many consecutive iterations, or having a very big positive reduced cost (Dell'Amico et al. 2006). We do not delete any route except for the sake of guaranteeing compatibility and feasibility. The reason to keep more routes is that the IP problem we solve at the end of column generation in each node would have a better chance to improve the upper bound if it uses more columns.

4.6. Stabilization

Degeneracy is a very common phenomenon when we apply column generation (Lubbecke and Desrosiers 2005). To decrease the number of degenerate solutions some authors exploit the stabilization techniques. To our knowledge, no absolute superiority of any stabilization technique over others has been reported yet (Lubbecke and Desrosiers 2005). We used the stabilization technique introduced by Du Merle et al. (1999). Our test results demonstrate that the efficiency of this technique highly depends on the number of routes introduced in the master problem in each iteration of column generation (the bunch size). In case where only a small number of routes are introduced in the master problem in each iteration, the stabilization technique helps in terms of decreasing the number of degenerate solutions, and hence the total computation time. On the other hand, if we introduce many routes in the master problem in each iteration of column generation it is more likely to capture a new solution and so to avoid degeneracy. Following the discussion in the previous sub-section about column management, we add many routes (up to 1000) with negative reduced costs in the master problem in each period, and stabilization did not prove useful in this framework.

5. Computational results

The branch-and-price algorithm was coded in Java and the instances were run on an Intel Core Duo processor with 2.8GHz CPU and 4GB RAM. We used ILOG CPLEX 12.4 to solve the restricted master problems. A time limit of one hour was set for each instance.

5.1. Instances

The 100-series instances created by Solomon (1987) were considered for the test problems. These include randomly distributed stores (R101-R112), clustered stores (C101-C109), and randomly-clustered stores (RC101-RC108). As in the original instances, capacity of the homogeneous freight is $Q = 200$, fixed cost of using each vehicle is zero, Euclidean distances represent the cost c_{ij} of traversing from store i to store j, time windows and service times are considered. The small instances consist of 30 stores, and 10 stores are included in each class; they are denoted as instances V_I-V_{II}-V_{III} =10-10-10. This implies that where postponing and advancing are both allowed, we need to solve two dependent VRPs, each with 30 real stores and 10 virtual stores. The medium instances are of size 40, and we consider 10, 10, and 20 stores in each class, respectively; these instances are represented by 10-10-20. By the same reasoning, where advancing and postponing are both allowed, the main problem consists of two dependent VRPs each with 40 real stores and 20 virtual ones. Finally, the big instances consist of 50 stores with the classification 20-20-10. When generating the medium and the big instances, we always start from the first store in Solomon instances. This means, for example, that for medium instances stores 1 to 10 are included in class V_I, stores 11 to 20 are included in class V_{II}, and stores 21 to 40 are included in class V_{III}; virtual stores in this example are numbered from 41 to 60 so that the virtual store i is associated with the real store $i-20$. Since all instances in each type (R, C, and RC) of Solomon instances consider the same location for all stores, e.g. coordinates of store 1 are the same in all instances R101-R112, we decided to diversify our small instances. We consider stores 1-30 in R101, stores 11-40 in R102, stores 21-50 in R103, ..., and stores 71-100 for instances R108-R112. Hence, the small R instances are represented in the tables as R101-1, R102-11, R103-21, and so on. We followed the same diversification strategy for types C and RC instances.

The order size of each store in class V_I (V_{II}) in period 1 (period 2) is equal to the corresponding order size in the original instances and zero in the other period. For the stores in class V_{III} we consider the same orders for both periods, each equal to the corresponding order in the original instances. When we allow postponing, it means that orders of all stores $i \in V_I \cup V_{III}$ can be postponed, and when advancing is allowed, it implies that orders of all stores $i \in V_{II} \cup V_{III}$ can be advanced. We solve each instance twice; once with a positive advancement penalty per unit and once with a zero advancement penalty. The postponement penalty is twice the advancement penalty. The positive advancement penalty per unit for type R (respectively, C, and RC) instances is equal to 2 (respectively, 0.2, and 1).

5.2. Results

The numerical results are presented in Tables 10-12, respectively for instances of types R, C, and RC. For each instance we have solved three problems; A0P0 in which neither advancing nor postponing is allowed, A1P0 in which advancing is allowed but postponing is not, and A1P1 in which both advancing and postponing are allowed. In the tables, $\text{Gap}_{\text{opt.}}$ shows the optimality gap of the best integer solution we find in the branching tree after one hour, i.e., the percentage deviation between the value of the best GUB and the best GLB available. An absolute zero, 0, for $\text{Gap}_{\text{opt.}}$ shows that we have obtained an integer solution in the root node; so, no branching is performed. A decimal zero, 0.000, for $\text{Gap}_{\text{opt.}}$ shows that the instance is solved to optimality after branching; the number of nodes is not reported in the tables. Gap_{IP} shows the gap between the value of the integer solution obtained by solving an IP in the root node and the best GLB (the GLB available after the instance is solved to optimality or one hour is elapsed). The next columns, Z, Veh., and Time, show the total costs in both periods, number of vehicles in both periods, and total time in seconds to solve the instance, respectively. %Z Imp. for problems A1P0 and A1P1 indicates the percentage improvement of Z in these problems with respect to the value of Z for problem A0P0. A dash sign (-) indicates instances for which we could not even solve the LP relaxation in the root node within the time limit.

Table 10 Small, medium and large instances of type R

	A0P0					A1P0						A1P1						
Instance	$Gap_{opt.}$	Gap_{IP}	Z	Veh.	Time	$Gap_{opt.}$	Gap_{IP}	% Z Imp.	Veh.	Adv.	Time	$Gap_{opt.}$	Gap_{IP}	% Z Imp.	Veh.	Adv.	Pos.	Time
10-10-10																		
R101-1	0	0	977.9	15	0	0	0	6.4	13	4	0	0	0	6.7	12	4	1	0
R102-11	0.000	0.020	1075.1	13	0	0.000	0.023	11.1	11	4	2	0.000	0.023	11.1	11	4	0	3
R103-21	0.000	0.000	926.2	10	0	0.000	0.000	6.7	9	4	5	0.000	0.000	8.1	9	3	1	7
R104-31	0.000	0.013	696.0	8	5	0.000	0.016	3.5	7	4	168	0	0	8.6	6	2	1	22
R105-41	0	0	1085.1	12	0	0	0	6.9	9	8	1	0	0	6.9	9	8	0	1
R106-51	0	0	811.5	10	0	0	0	3.7	9	3	1	0	0	3.7	9	3	0	2
R107-61	0	0	767.1	8	0	0.000	0.000	3.1	8	2	41	0.000	0.000	3.1	8	2	0	21
R108-71	0.000	0.000	464.5	6	304	0.009	0.010	0.1	5	3	3600	0.009	0.010	0.1	5	3	0	3600
R109-71	0.000	0.006	498.8	6	1	0.000	0.003	0.5	6	1	29	0.000	0.015	0.5	6	1	0	18
R110-71	0.000	0.003	525.3	7	4	0	0	1.7	7	1	15	0.000	0.015	1.7	7	1	0	47
R111-71	0.000	0.000	473.7	6	15	0.000	0.009	0.7	6	2	42	0.000	0.003	1.0	6	1	1	382
R112-71	0.000	0.009	460.4	6	779	0.021	0.030	0.2	6	1	3600	0.018	0.024	0.9	6	0	1	3600
10-10-20																		
R101	0.000	0.012	1433.5	18	0	0.000	0.001	9.3	15	10	4	0.000	0.001	9.6	14	9	2	2
R102	0.000	0.014	1298.8	16	1	0	0	10.3	14	9	6	0	0	10.6	13	5	2	7
R103	0.000	0.029	1127.8	13	9	0.000	0.000	6.7	11	8	449	0.000	0.000	8.1	11	7	1	73
R104	0.000	0.008	938.6	9	69	0.000	0.011	6.0	7	6	1914	0.016	0.053	5.0	7	6	0	3600
R105	0.000	0.025	1305.3	14	3	0.000	0.002	8.4	12	9	19	0.000	0.002	8.4	12	9	0	29
R106	0	0	1161.0	13	1	0.000	0.000	7.4	11	7	27	0.000	0.000	7.4	11	7	0	32
R107	0	0	1034.9	11	2	0	0	6.3	10	5	33	0	0	6.3	10	5	0	54
R108	0.000	0.009	903.8	8	92	0.008	0.012	3.2	8	4	3600	0.010	0.018	4.5	7	5	0	3600
R109	0.000	0.019	1126.2	12	9	0.000	0.000	6.6	10	7	28	0	0	6.9	9	7	1	28
R110	0.000	0.023	1026.5	10	16	0.000	0.013	4.7	9	6	264	0.000	0.013	4.7	9	6	0	332
R111	0.000	0.049	1032.0	10	38	0	0	6.0	9	4	49	0	0	6.0	9	4	0	63
R112	0	0	907.4	8	5	0.000	0.000	3.4	7	6	260	0.000	0.000	3.4	7	6	0	432
20-20-10																		
R101	0.000	0.003	1422.2	17	1	0.000	0.002	9.3	16	4	3	0.000	0.004	9.6	16	4	0	6
R102	0.000	0.002	1249.8	15	1	0.000	0.004	10.3	13	4	15	0.000	0.004	10.6	13	4	0	26
R103	0.000	0.000	1113.8	12	3	0.000	0.000	6.7	11	5	53	0.000	0.000	8.1	11	5	0	43
R104	0.000	0.046	899.8	9	313	0.003	0.003	6.0	9	2	3600	0.012	0.028	5.0	9	2	0	3600
R105	0.000	0.003	1229.7	13	2	0	0	8.4	12	4	3	0	0	8.4	12	4	1	3
R106	0.000	0.016	1126.2	11	12	0.000	0.004	7.4	11	3	128	0.000	0.000	7.4	11	2	3	72
R107	0.000	0.001	1001.1	10	23	0.000	0.002	6.3	10	1	639	0.000	0.000	6.3	10	1	0	1359
R108	0.000	0.046	876.1	8	179	0.002	0.022	3.2	8	1	3600	0.021	0.002	4.5	8	1	0	3600
R109	0.000	0.014	1093.6	11	7	0.000	0.016	6.6	10	3	96	0	0.021	6.9	10	2	1	21
R110	0.000	0.006	1009.8	10	47	0.000	0.006	4.7	10	2	2037	0.000	0.017	4.7	10	0	2	2705
R111	0.000	0.010	1002.5	10	61	0.000	0.001	6.0	9	6	1360	0.000	0.017	6.0	9	6	0	1877
R112	0.000	0.001	879.7	8	53	0.000	0.021	3.4	8	1	1300	0.000	0.025	3.4	8	1	1	757

Table 11 Small, medium and large instances of type C

Instance	A0P0					A1P0						A1P1						
	$Gap_{opt.}$	Gap_{IP}	Z	Veh.	Time	$Gap_{opt.}$	Gap_{IP}	% Z Imp.	Veh.	Adv.	Time	$Gap_{opt.}$	Gap_{IP}	% Z Imp.	Veh.	Adv.	Pos.	Time
10-10-10																		
C101-1	0	0	294.5	5	0	0.000	0.019	13.8	4	3	13	0.000	0.049	13.8	4	3	0	14
C102-11	0	0	470.9	6	0	0.000	0.000	8.3	5	1	84	0.000	0.018	9.6	5	1	1	250
C103-21	0	0	265.7	4	2	0.073	0.209	0.0	4	0	3600	0.080	0.084	0.0	4	0	0	3600
C104-31	0.071	0.146	419.6	5	3600	—	—	—	–	–	3600	—	—	—	–	–	–	3600
C105-41	0.000	0.059	487.1	4	2	0.000	0.071	23.4	4	12	26	0.000	0.008	24.9	4	6	2	68
C106-51	0.000	0.034	533.4	5	3	0.098	0.189	9.1	4	10	3600	0.169	0.245	9.1	4	10	0	3600
C107-61	0	0	501.9	6	0	0.000	0.169	24.7	4	12	17	0.000	0.133	24.7	4	4	2	47
C108-71	0.000	0.159	546.6	6	32	0.000	0.141	14.0	5	1	155	0	0	16.9	5	1	2	24
C109-81	0.000	0.159	509.3	5	5	0.000	0.137	7.2	5	1	1111	0	0	10.8	5	1	2	37
10-10-20																		
C101	0	0	569.7	8	0	0.000	0.040	13.8	7	4	166	0.000	0.010	13.8	7	4	0	624
C102	0	0	568.7	8	2	0.018	0.018	13.7	7	6	3600	0.022	0.022	13.7	7	6	0	3600
C103	0	0	564.7	8	6	—	—	—	–	–	3600	—	—	—	–	–	–	3600
C104	—	—	—-	–	3600	—	—	—	–	–	3600	—	—	—	–	–	–	3600
C105	0.000	0.002	569.7	8	4	0.000	0.010	13.8	7	4	1865	0.028	0.037	13.8	7	4	0	3600
C106	0	0	569.7	8	0	0.000	0.043	13.8	7	4	964	0.016	0.034	13.8	7	4	0	3600
C107	0.000	0.002	569.7	8	4	0.024	0.025	13.8	7	4	3600	0.035	0.039	13.8	7	4	0	3600
C108	0.000	0.074	568.9	8	41	0.031	0.078	13.7	7	4	3600	0.023	0.059	14.7	6	5	1	3600
C109	0.018	0.067	548.9	7	3600	0.042	0.091	10.5	7	4	3600	0.035	0.085	11.1	6	7	2	3600
20-20-10																		
C101	0	0	440.7	7	0	0.000	0.067	0.7	6	11	131	0.000	0.017	3.6	6	0	1	337
C102	0	0	432.5	6	3	0.000	0.080	0.3	6	4	1057	0.000	0.027	2.3	6	4	1	2262
C103	0	0	427.0	6	54	—	—	—	–	–	3600	—	—	—	–	–	–	3600
C104	—	—	—-	–	3600	—	—	—	–	–	3600	—	—	—	–	–	–	3600
C105	0	0	440.7	7	2	0.000	0.027	1.1	6	11	697	0.000	0.055	3.4	6	3	1	1996
C106	0	0	439.9	7	1	0.000	0.064	0.6	6	11	524	0.000	0.028	3.9	6	4	1	451
C107	0	0	440.7	7	2	0.000	0.084	0.8	6	11	619	0.000	0.019	3.9	6	4	1	1022
C108	0	0	428.6	6	3	0.000	0.007	0.5	6	3	2267	0.000	0.018	1.6	6	3	1	2973
C109	0	0	425.3	6	8	0.033	0.045	0.2	6	5	3600	0.029	0.048	0.8	6	3	1	3600

Table 12 Small, medium and large instances of type RC

	A0P0					A1P0						A1P1						
Instance	$Gap_{opt.}$	Gap_{IP}	Z	Veh.	Time	$Gap_{opt.}$	Gap_{IP}	% Z Imp.	Veh.	Adv.	Time	$Gap_{opt.}$	Gap_{IP}	% Z Imp.	Veh.	Adv.	Pos.	Time
10-10-10																		
RC101-1	0.000	0.040	1087.7	10	108	0.000	0.038	19.1	7	7	37	0.000	0.032	20.8	7	4	1	32
RC102-11	0.000	0.001	864.1	8	12	0.000	0.001	9.3	7	2	7	0.000	0.039	9.3	7	2	0	13
RC103-21	0.000	0.002	924.2	8	7	0	0	20.1	6	6	4	0	0	20.1	6	6	0	8
RC104-31	0	0	774.8	6	1	0	0	5.1	5	6	15	0	0	5.1	5	6	0	25
RC105-41	0.000	0.018	873.5	10	0	0	0	6.7	9	6	1	0	0	7.4	9	3	2	2
RC106-51	0	0	835.9	8	0	0.000	0.002	12.6	6	10	6	0.000	0.010	13.8	6	7	1	16
RC107-61	0.000	0.005	740.1	9	2	0.000	0.000	5.0	7	6	20	0.000	0.015	5.2	6	4	2	12
RC108-71	0	0	670.2	6	0	0.000	0.000	2.6	7	7	58	0.000	0.011	2.6	7	7	0	55
10-10-20																		
RC101	0.000	0.019	1306.7	12	176	0.000	0.019	10.9	9	11	257	0.000	0.000	12.9	9	4	2	18
RC102	0.000	0.002	1195.8	11	87	0.026	0.031	7.6	10	2	3600	0	0	12.6	9	2	2	19
RC103	0	0	994.4	9	2	0	0	0.4	9	1	46	0	0	0.4	9	1	0	49
RC104	0	0	901.9	9	5	0	0	0.0	9	0	115	0	0	0.0	9	0	0	183
RC105	0.000	0.000	1254.9	12	839	0.052	0.057	8.5	10	2	3600	0	0	14.7	9	2	1	19
RC106	0.000	0.009	1145.8	11	1381	0.059	0.064	7.6	10	2	3600	0	0	13.6	9	2	1	21
RC107	0.000	0.029	925.2	9	9	0.000	0.013	0.4	9	1	267	0	0	2.6	8	1	1	54
RC108	0.000	0.000	894.4	9	137	0.006	0.006	0.0	9	0	3600	0.000	0.011	1.6	8	0	1	732
20-20-10																		
RC101	0.000	0.025	1347.2	13	592	0.041	0.063	14.4	10	6	3600	0.049	0.068	14.4	10	6	0	3600
RC102	0.000	0.008	1143.2	10	2340	0.084	0.113	9.4	9	6	3600	0.092	0.106	9.4	9	6	0	3600
RC103	0.031	0.081	1017.2	9	3600	0.074	0.147	6.9	7	5	3600	0.116	0.120	6.9	7	5	0	3600
RC104	0.000	0.006	855.1	8	40	0.053	0.104	4.1	7	3	3600	0.053	0.096	4.2	7	3	0	3600
RC105	0.000	0.004	1197.1	11	2478	0.048	0.093	13.5	9	5	3600	0.050	0.067	13.5	9	5	0	3600
RC106	0.000	0.000	1070.6	10	663	0.089	0.109	4.0	9	7	3600	0.110	0.121	4.0	9	7	0	3600
RC107	0.013	0.013	981.6	10	3600	0.056	0.074	5.6	8	4	3600	0.075	0.082	5.8	8	4	1	3600
RC108	0.052	0.058	917.8	9	3600	0.092	0.141	5.5	7	3	3600	0.093	0.115	5.5	7	3	0	3600

5.3. Algorithmic insights

As explained in Section 4.1, we tested the effect of solving an IP model in each node after column generation. We applied this idea on small instances only. For each small instance we solve three problems, i.e., A0P0, A1P0, and A1P1, each with four different values of advancing penalties (the postponing penalties are twice the advancing penalties). Obviously, the penalty does not impact A0P0 because neither advancing nor postponing is allowed there. So, for each small instance we solve $1 + 2 \times 4 = 9$ sub-instances, for a total of $(12 + 9 + 8) \times 9 = 261$ small instances, each with and without IP solution at each node. Out of these 261, 221 are solved to optimality within one hour. For these 221 instances the computation time, on average, increases by 2% when we solve IPs. The remaining 40 instances were not solved to optimality after one hour. Figure 6 shows the optimality gap of the best integer solution with/without solving the IPs.

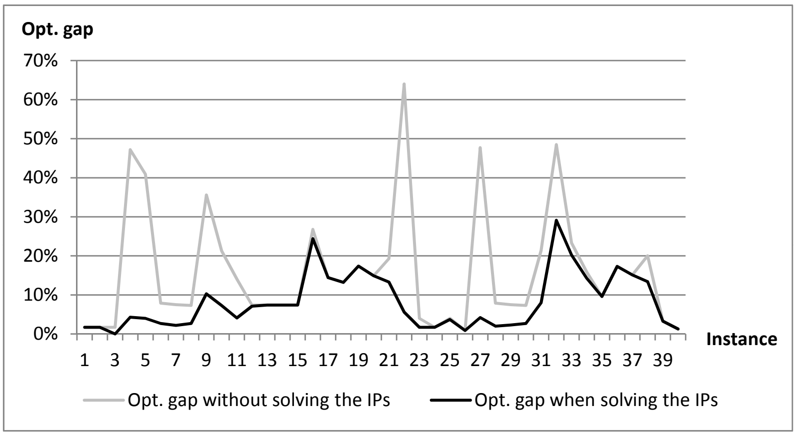

Figure 6 Optimality gap in the small instances with different penalties which are not solved to optimality within one hour.

As it appears from Figure 6, when solving the IPs we obtain a much smaller optimality gap, mostly due to improved upper bounds. On average, the optimality gap improves by 33.6%. In sum, our test results on 261 instances show that solving an IP in each node does not have any significant impact on the computation time for the instances which are solved to optimality, but considerably improves the optimality gap of those instances which are not solved to optimality. Hence, we decided to use this procedure for all medium and big instances.

The second algorithmic idea was to record the integer solutions obtained by chance during the course of column generation, and to check whether they improve the current upper bound. This was not a successful test since many routes are added to the master problem in each iteration of the column generation, and we rarely obtain integer solutions during the course of column generation.

The third issue we tested was the quality of the IP solution in the root node compared to the best integer solution (the best upper bound) we find within one hour. To this end, we consider all small, medium, and big instances (including all problems A0P0, A1P0, A1P1). For each instance of type R (respectively, C, and RC) with the advancement penalty per unit equal to 2 (respectively, 0.2, and 1) and for each problem type (A0P0, A1P0, and A1P1), we have three problem sizes, i.e., small, medium, and big. Hence, we solve $3 \times 3 = 9$ (sub-)instances for each instance of type R (respectively, C, and RC), which counts for solving a total number of $(12 + 9 + 8) \times 9 = 261$ instances.

We divide these 261 instances into those solved to optimality (209 instances) and those not solved to optimality (52 instances) within the time limit. Figure 7 shows the optimality gap of the IP solution in root node, i.e., Gap_{IP}, with respect to the optimal solution for 135 instances solved to optimality for which branching was required. In other words, in Figure 7, we have excluded 74 instances which are solved to optimality in the root node, in order to avoid overestimating the quality of an IP solution. We observe very low optimality gaps for these integer root solutions, which are obtained before any branching. The average optimality gap of the IP solutions for these 135 instances is 2.1%.

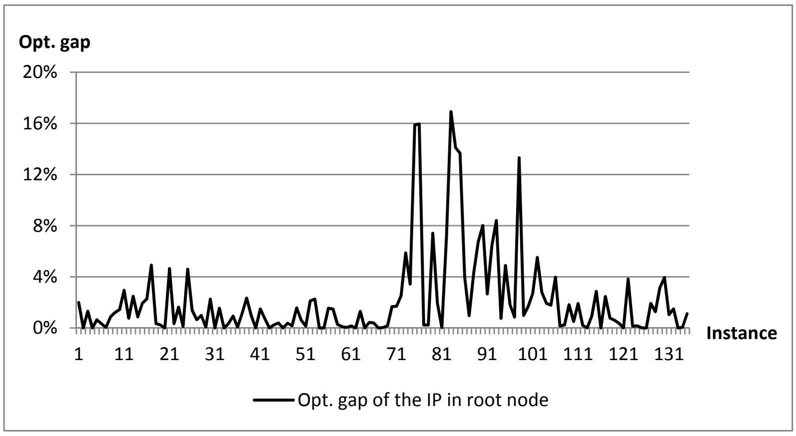

Figure 7 **Optimality gap of the IP in root node, Gap_{IP}, in the instances solved to optimality.**

The optimality gap of the IP solution in root node, Gap_{IP}, for the second category containing 52 instances which were not solved to optimality is represented in Figure 8 along with the optimality gap of the best integer solution within one hour, $\text{Gap}_{\text{opt.}}$, both with respect to the best lower bound after the time limit. Again, we observe that the gap of the IP solution we immediately obtain in the root node is very close to the optimality gap we obtain after one hour.

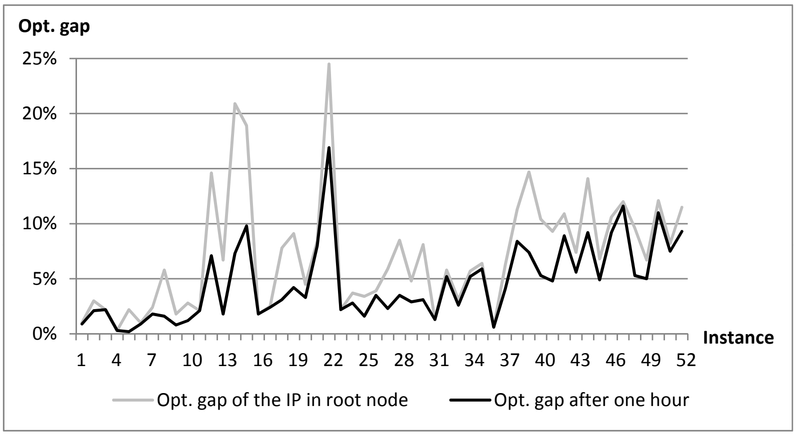

Figure 8 **Optimality gap of the IP in root node, Gap_{IP}, and the optimality gap, $\text{Gap}_{\text{opt.}}$, in the instances which are not solved to optimality within one hour.**

Finally, it is worth mentioning that besides three acceleration techniques mentioned in Section 4.4, considering all compatible routes of a father node when we start solving the LP relaxation in a child node greatly helped us to increase the speed of our branch-and-price algorithm.

5.4. Managerial insights

By allowing the orders to be advanced and/or postponed, we logically expect to obtain lower transportation costs for two periods. We have tried to quantify the improvement by testing our models A1P0 and A1P1 against the basic model A0P0. Table 13 shows the results of solving instances when no advancement or postponement penalty applies. So, it indicates the maximum savings we may reap in problems A1P0 and A1P1 as compared to A0P0. The value %Adv. shows the average percentage of the number of orders in period 2 which are advanced to period 1, out of the number of orders which could be advanced, for the model A1P0. This is replaced by %(Adv.+Pos.) that shows the average percentage of the orders postponed or advanced for the model A1P1. The value %Z Imp. shows the average percentage of improvement in transportation costs in both periods. The values %Occ. Inc. and %Veh. Dec. show the average percentage of increase in occupation of the vehicles and the average percentage of decrease in the number of vehicles, respectively, in both periods compared to the basic model A0P0. This table only takes into account the instances solved to optimality.

We observe that the average maximum cost improvement (when penalty is zero) is almost the same for both models A1P0 and A1P1, i.e., 30.6%. This implies that if the advancing penalty is very low and the postponing penalty is at least twice the advancing penalty, then allowing advancing is almost as efficient as allowing both advancing and postponing. Moreover, Table 13 shows that in model A1P0 (A1P1) we can increase the occupation of the vehicles by 40% (34%) and decrease the number of vehicles by 25.2% (26%), which are both desirable results.

Table 13 Maximum improvements when penalty is zero

		A1P0				A1P1			
Instance	Pen.	% Adv.	% Z Imp.	% Occ. Inc.	% Veh. Dec.	% (Adv.+Pos.)	% Z Imp.	% Occ. Inc.	% Veh. Dec.
R	0	96.1	33.3	45.4	29.4	62.8	33.3	43.6	33.7
C	0	74.3	25.3	23.2	14.4	74.3	25.4	15.3	14.0
RC	0	98.7	33.1	52.8	32.0	72.2	33.1	43.1	28.1
Average		89	30.6	40.0	25.2	60.4	30.6	34.0	26.0

Table 14 is similar to Table 13. However, in Table 14 we have considered positive penalties for all instances. These penalties are 2, 0.2, and 1 per unit of advanced order for instances R, C, and RC, respectively. The postponing penalties are twice the advancing ones. These penalty values have been adjusted in such a way that the %Adv. as well as %(Adv.+Pos.) is more or less the same for all instances R, C, and RC.

Table 14 Improvements when a positive penalty is considered

		A1P0				A1P1			
Instance	Pen.	% Adv.	% Z Imp.	% Occ. Inc.	% Veh. Dec.	% (Adv.+Pos.)	% Z Imp.	% Occ. Inc.	% Veh. Dec.
R	2	17.5	5	11.3	9.5	12	5.3	12.8	10.4
C	0.2	24	8.8	16.1	11.3	10.9	11.1	15.4	12.2
RC	1	24.4	7.7	16.7	12.5	13.6	8.5	20.2	15.3
Average		20.6	6.5	13.7	10.6	11.3	7.4	15.4	12.1

In Table 14 the savings are reduced since a positive penalty is exercised. However, the cost savings are still appealing: 6.5% for model A1P0 and 7.4% for model A1P1, in average. By considering positive penalties we observe that, in our test instances, there is little difference between models A1P0 and A1P1 in terms of cost saving, i.e., 0.9%, as long as the postponing penalty is at least twice the advancing penalty. However, it is clear that the latter ratio may change for other instances. For the sake of illustration, Figure 9 displays the optimal routes for instance R103 (10-10-20) when solving model A0P0, whereas Figure 10 displays the improved routes obtained when solving model A1P1. In these figures, stores 1-10 belong to class V_I, stores 11-20 are in class V_{II}, and stores 21-40 belong to class V_{III}. We observe that, in model A1P1, the orders of stores 17, 25, 33, 35, 36, 38, and 40 are advanced, whereas the demand of store 24 is postponed. According to Table 10, the savings (while also taking the penalties into account) when applying model A1P1 is 8.1% as compared to the basic model A0P0.

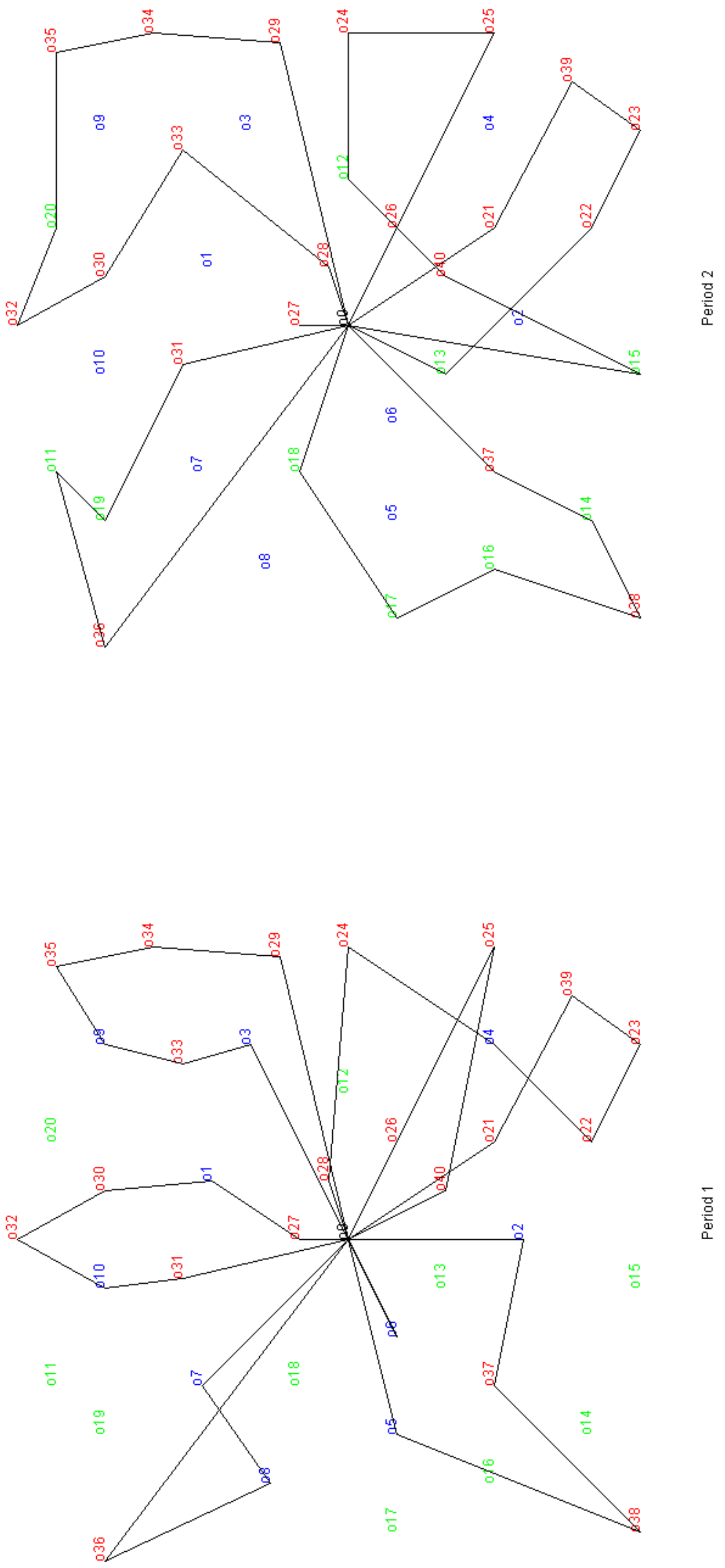

Figure 9 Optimal routes for instance R103 (10-10-20) in the basic model A0P0.

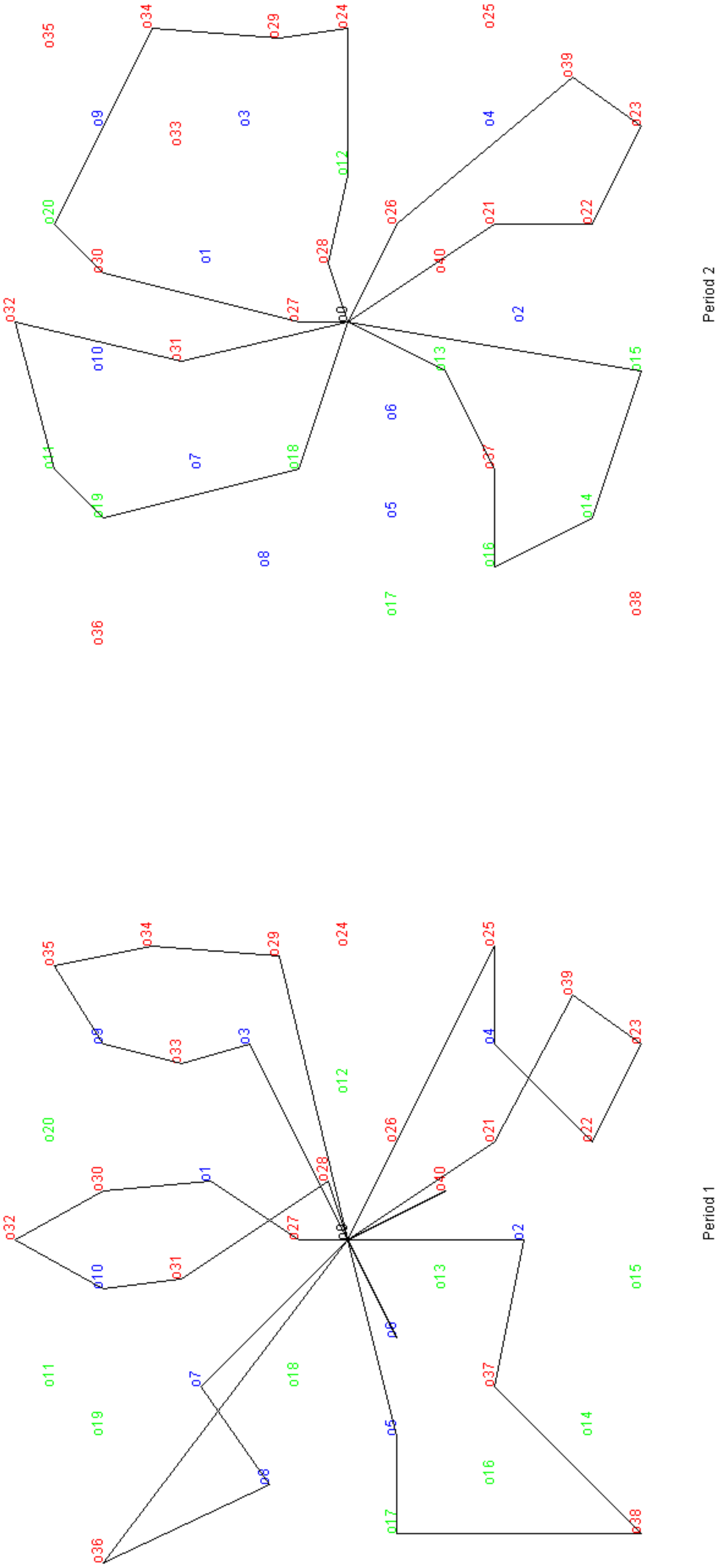

Figure 10 Optimal routes for instance R103 (10-10-20) in model A1P1.

6. Conclusions and future research

We have defined a 2-period VRP inspired from a real problem, namely inventory control of fresh products in chain stores. The main focus of this paper is on modeling the resulting 2-period VRP from a LSP's perspective. Branch-and-price has been exploited to solve the 2-period VRP where the master problem is solved by column generation and two pricing sub-problems are solved through the label-setting algorithm. We have used many tricks and acceleration techniques for the column generation and the pricing problems to remain competitive, in terms of computation time, with the existing works on the VRP. Algorithmically, we investigated two ideas in the branching tree in order to potentially improve the upper bound during the branching process; namely, we record integer solutions we encounter during the course of column generation in each node and we solve an IP at the end of column generation in each node. Our experimental results show that the first idea is not efficient but the second idea considerably improves the upper bound. Moreover, solving an IP formulation with a restricted number of columns when the column generation procedure terminates in the root node already delivers a very good integer solution. In managerial terms, the results demonstrate that we can significantly decrease the routing costs when orders are allowed to be advanced, especially in case the advancement penalty is low. Moreover, for our test instances, advancement and postponement compared to merely advancement does not yield a significantly better solution as long as the postponement penalty is at least twice the advancement penalty.

As the numerical results appear promising, working on further acceleration techniques such as bidirectional search (Righini and Salani 2006) to solve the pricing problems could be beneficial. The problem described in this paper can be generalized in order to describe other real situations; a more general problem formulation is presented in the Appendix. Furthermore, a natural extension of the model is to deal with partial advancement and postponement rather than full ones.

Acknowledgments

The first two authors have been partially supported in the course of this research by the Interuniversity Attraction Poles Programme initiated by the Belgian Science Policy Office (Grant COMEX P7/36).

Appendix. More classes

We assumed in section 1 that for the stores in class V_I the alternative decision about the delivery quantities is zero for period 1 and a positive quantity $d^{'}_{i2}$ for period 2. While the alternative delivery quantity for these stores in period 1 is always zero, the alternative requested order in period 2 could be zero, too. This could be the case when period 2 coincides with a holiday, e.g., Sunday. In order to extend our initial model to include such stores we introduce a new class of stores V_{IV} for which the initial decision about delivery quantities is similar to class V_I but the alternative decision is to deliver zero in both periods. Note that in class V_{IV}, as opposed to V_I, a store is not necessarily visited. This implies in particular that constraint (4) has to be replaced by a weaken one, as shown by constraint (26). Moreover, the terms associated to stores in class V_{IV} in the objective function (3) is no longer $\sum_{i\in V_I}\Delta_i(\sum_{r\in R_2}\alpha_{ir}u_{r2})$, but it is the term represented in (21).

By a similar reasoning, we can also introduce other classes of stores as follows.

Table 15 Further classees of stores

	Initial decision			Alternative dec. 1			Alternative dec. 2		
Class	del.$_1$	del.$_2$	pen.	del.$_1$	del.$_2$	pen.	del.$_1$	del.$_2$	pen.
V_I	d_{i1}	0	0	0	d'_{i2}	Δ_i	—	—	—
V_{II}	0	d_{i2}	0	d'_{i1}	0	Δ'_i	—	—	—
V_{III}	d_{i1}	d_{i2}	0	0	d'_{i2}	Δ_i	d'_{i1}	0	Δ'_i
V_{IV}	d_{i1}	0	0	0	0	Δ_i	—	—	—
V_V	0	d_{i2}	0	0	0	Δ'_i	—	—	—
V_{VI}	d_{i1}	d_{i2}	0	0	d'_{i2}	Δ_i	0	0	Δ'_i
V_{VII}	d_{i1}	d_{i2}	0	0	0	Δ_i	d'_{i1}	0	Δ'_i
V_{VIII}	d_{i1}	d_{i2}	0	0	0	Δ_i	0	0	Δ'_i

Our initial model can be further developed to encompass such classes of stores. The terms to be embedded in the objective function (3) for these five classes are presented by terms (21)-(25).

$$\sum_{i\in V_{IV}}\Delta_i(1-\sum_{r\in R_1}\alpha_{ir}u_{r1}) \tag{21}$$

$$\sum_{i\in V_V}\Delta'_i(1-\sum_{r\in R_2}\alpha_{ir}u_{r2}) \tag{22}$$

$$\sum_{i\in V_{VI}}\Delta_i\sum_{r\in R_2}\alpha_{i+n_3,r}u_{r2}+\sum_{i\in V_{VI}}\Delta'_i(1-\sum_{r\in R_2}\alpha_{ir}u_{r2}) \tag{23}$$

$$\sum_{i\in V_{VII}}\Delta_i\sum_{r\in R_1}\alpha_{i+n_3,r}u_{r1}+\sum_{i\in V_{VII}}\Delta'_i(1-\sum_{r\in R_1}\alpha_{ir}u_{r1}) \tag{24}$$

$$\sum_{i\in V_{VIII}}\Delta_i(1-\sum_{r\in R_1}\alpha_{ir}u_{r1}) \tag{25}$$

Moreover, the pertinent constraints for these classes are as follows:

$$\sum_{r\in R_1}\alpha_{ir}u_{r1}\le 1;\ \forall i\in V_{IV} \tag{26}$$

$$\sum_{r\in R_1}\alpha_{ir}u_{r1}\ge 1-g_i;\ \forall i\in V_{IV} \tag{27}$$

$$\sum_{r\in R_2}\alpha_{ir}u_{r2}\le 1;\ \forall i\in V_V \tag{28}$$

$$\sum_{r\in R_2}\alpha_{ir}u_{r2}\ge 1-g'_i;\ \forall i\in V_V \tag{29}$$

$$\sum_{r\in R_1}\alpha_{ir}u_{r1}\leq 1;\ \forall i\in V_{VI}\cup V_{VII}\cup V_{VIII} \tag{30}$$

$$\sum_{r\in R_2}\alpha_{ir}u_{r2}\leq 1;\ \forall i\in V_{VI}\cup V_{VII}\cup V_{VIII} \tag{31}$$

$$\sum_{r\in R_1}\alpha_{ir}u_{r1}\geq 1-g_i;\ \forall i\in V_{VI}\cup V_{VII}\cup V_{VIII} \tag{32}$$

$$\sum_{r\in R_2}\alpha_{ir}u_{r2}\geq 1-g_i^{'};\ \forall i\in V_{VI}\cup V_{VII}\cup V_{VIII} \tag{33}$$

If we do not impose constraints (26) and (27) the master problem tends to consider many routes that include $i\in V_{IV}$ and $i\in V_V$, in that they decrease value of the objective function due to their negative terms. We can eliminate constraints (32) and (33) and respect them during column generation. The pricing problems should be altered accordingly, too.

References

Abdelmaguid T F, Dessouky M M (2006) A genetic algorithm approach to the integrated inventory-distribution problem. *International Journal of Production Research*, 44(21):4445-4464.

Albareda-Sambola M, Fernandez E, Laporte G (2014) The dynamic multiperiod vehicle routing problem with probabilistic information. *Computers & Operations Research*, 48:31-39.

Andersson H, Hoff A, Christiansen M, Hasle G, Lokketangen A (2010) Industrial aspects and literature survey: combined inventory management and routing. *Computers & Operations Research*, 37:1515-1536.

Angelelli E, Savelsbergh M, Speranza M G (2007) Competitive analysis for dynamic multi-period uncapacitated routing problems. *Networks*, 49(4):318-329.

Angelelli E, Savelsbergh M, Speranza M G (2007) Competitive analysis of a dispatch policy for a dynamic multi-period routing problem. *Operations Research Letters*, 35:713-721.

Angelelli E, Bianchessi N, Mansini R, Speranza M G (2009) Short term strategies for a dynamic multi-period routing problem. *Transportation Research Part C*, 17:106-119.

Archetti C, Bertazzi L, Laporte G, Speranza M G (2008) A branch-and-cut algorithm for a vendor managed inventory routing problem. *Transportation Science*, 41:382-391.

Bertazzi L, Paletta G, Speranza M G (2002) Deterministic order-up-to level policies in an inventory routing problem. *Transportation Science*, 36:119-132.

Bertazzi L (2008) Analysis of direct shipping policies in an inventory routing problem with discrete shipping times. *Management Science*, 54(4):748-762.

Bertazzi L, Speranza M G (2012) Matheuristics for inventory routing problems. In: Montoya-Torres J R, Juan A A, Huatuco L H, Faulin J, Rodriguez-Verjan G L, Hybric algorithms for service, computing, and manufacturing systems: routing and scheduling solutions. IGI Global, Hershey, 1-14.

Boland N, Dethridge J, Dumitrescu I (2006) Accelerated label setting algorithms for the elementary resource constrained shortest path problem. *Operations Research Letters*, 34(1):58-68.

Campbell A M, Savelsbergh M (2004) A decomposition approach for the inventory routing problem. *Transportation Science*, 38(4):488-502.

Coelho L C, Laporte G (2013) The exact solution of several classes of inventory routing problems. *Computers & Operations Research*, 40:558-565.

Clarke G, Wright J W (1964) Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4):568-581.

Dabia S, Ropke S, Van Woensel T, De Kok T (2013) Branch and price for the time-dependent vehicle routing problem with time windows. *Transportation Science*, 47(3):380-396.

Dell'Amico M, Righini G, Salani M (2006) A branch-and-price approach to the vehicle routing problem with simultaneous distribution and collection. *Transportation Science*, 40(2):235-247.

Desaulniers G, Lessard F, Hadjar A (2008) Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. *Transportation Science*, 42(3):387-404.

Du Merle O, Villeneuve D, Desrosiers J, Hansen P (1999) Stabilized column generation. *Discrete Mathematics*, 194:229-237.

Feillet D, Dejax P, Gendreau M, Gueguen C (2004) An exact algorithm for the elementary shortest path problem with resource constraints: application to some vehicle routing problems. *Networks*, 44(3):216-229.

Gallego G, Simchi-Levi D (1990) On the effectiveness of direct shipping strategy for the one-warehouse multi-retailer R-systems. *Management Science*, 36:240-243.

Gutierrez-Jarpa G, Desaulniers G, Laporte G, Marianov V (2010) A branch-and-price algorithm for the vehicle routing problem with deliveries, selective pickups and time windows. *European Journal of Operational Research*, 206:341-349.

Irnich S, Desaulniers G (2005) Shortest path problems with resource constraints. Desaulniers G, Desrosiers J, Solomon M M, eds. *Column Generation* (Springer, New York), 33-65.

Lubbecke M E, Desrosiers J (2005) Selected topics in column generation. *Operations Research*, 53(6):1007-1023.

Ozener O, Ergun O, Savelsbergh M (2013) Allocation cost of service to customers in inventory routing. *Operations Research*, 60(5):1-14.

Paessens H (1988) The savings algorithm for the vehicle routing problem. *European Journal of Operational Research*, 34(3):336-344.

Powell W E (2011) *Approximate dynamic programming: solving the curses of dimensionality*, (Wiley, New Jersey).

Raa B, Aghezaaf E H (2008) Designing distribution patterns for long-term inventory routing with constant demand rates. *International Journal of Production Economics*, 112:255-263.

Raa B, Aghezaaf E H (2009) A practical solution approach for the cyclic inventory routing problem. *European Journal of Operational Research*, 192:429-441.

Righini G, Salani M (2006) Symmetry helps: bounded bi-directional dynamic programming for the elementary shorttest path problem with resource constraints. *Discrete Optimization*, 3:255-273.

Righini G, Salani M (2008) New dynamic programming algorithms for the resource constrained elementary shortest path problem. *Networks*, 51(3):155-170.

Solomon M M (1987) Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254-265.

Van Donselaar K, Van Woensel T, Broekmeulen R, Fransoo J (2006) Inventory control of perishables in supermarkets. *International Journal of Production Economics*, 104:462-472.

Wen M, Cordeau J F, Laporte G, Larsen J (2010) The dynamic multi-period vehicle routing problem. *Computers & Operations Research*, 37:1615-1623.