An Exact Solution Approach for the Inventory Routing Problem with Time Windows

Gizem Ozbaygin[1], Esra Koca[1], and Hande Yaman[2]

[1]Faculty of Engineering and Natural Sciences, Sabanci University, Istanbul 34956, Turkey

[2]ORSTAT, Faculty of Economics and Business (FEB), KU Leuven, 3000 Leuven, Belgium

March 2020

Abstract

The inventory routing problem (IRP) is an integrated inventory and transportation planning problem that jointly determines the replenishment schedules for a given set of retailers, and the routing decisions for a supplier that distributes a product to the retailers over a finite planning horizon typically consisting of multiple periods. In the classical IRP, retailers are assumed to accept deliveries at any time during a given period. However, this assumption does not always hold in practice. Although the supplier is the central decision maker, the retailers may also have operational restrictions preventing them from receiving deliveries at arbitrary times. With this motivation, we study the IRP with time windows and develop an exact algorithm to solve it. In particular, we propose a new formulation and an efficient branch-price-and-cut algorithm, which, to the best of our knowledge, is the first exact approach specifically designed for solving the IRPTW. Our formulation involves both arc and route based variables to obtain an easier structure in the pricing problems. We use valid inequalities –originally derived for the lot sizing problem– to strengthen our formulation. The results of a computational study conducted on a set of newly generated benchmark instances demonstrate the effectiveness of our algorithm, which is capable of identifying optimal or fairly good solutions to medium and large problem instances in reasonable computation times.

Keywords: Inventory routing; time windows; branch-price-and-cut; lot sizing; valid inequalities

1 Introduction

The inventory routing problem (IRP) is an integrated inventory and transportation planning problem that jointly determines the replenishment schedules for a given set of retailers, and the routing decisions for a supplier that needs to distribute a product to the retailers over a finite planning horizon typically consisting of multiple periods. Demands of the retailers in every period are assumed to be known and

should be satisfied without any shortages. Deliveries to the retailers are made by a set of vehicles, each with a limited capacity, through a distribution network. Replenishment policies of the retailers are determined by the supplier. There are two well-known and frequently used policies in the literature: the order-up-to level (OU) and the maximum level (ML) policies. Under the OU policy, the inventory level of a retailer is set to a quantity specified by the retailer –known as the order-up-to level– whenever replenishment occurs. Under the ML policy, the inventory level upon replenishment should not exceed, but is allowed to be less than, the order-up-to level in any period. The objective of the IRP is to minimize the total transportation and inventory holding costs.

Applications of the IRP can be observed in many companies and industries today due to its success in reducing the operational costs. Studies such as Andersson et al. (2010), Christiansen (1999) (liquefied natural gas distribution through ships), Dauzère-Pérès et al. (2007) (raw material distribution in the paper industry), Mercer and Tao (1996) (food distribution to supermarkets) are based on some practical applications of the problem.

Different IRP variants have been studied in the literature. The special case of the problem that includes only one vehicle was studied by Archetti et al. (2007) and Solyalı and Süral (2011). In both of these studies, the problem was solved by a branch-and-cut algorithm. Solyalı and Süral (2011) applied the shortest-path reformulation of the lot sizing problem (Eppen and Martin, 1987) to the IRP with a single vehicle. A similar approach was used by Avella et al. (2015) to derive tight reformulations for the lot sizing subproblems that arise for each retailer. Their reformulation makes use of the special structure of the benchmark instances and its size grows quickly for arbitrary demands under the ML policy.

There are a few studies on the exact solution methods for the multi-vehicle IRP. Coelho and Laporte (2013) proposed a branch-and-cut algorithm to solve several variants of the problem such as the IRP with transshipment and the IRP with consistency features. Adulyasak et al. (2013) developed formulations of the IRP (with/without using vehicle indexed variables) under OU and ML policies. Archetti et al. (2014) extended the formulations proposed by Archetti et al. (2007) and Solyalı and Süral (2011) for the single vehicle IRP to the multi-vehicle case, and tested them on benchmark instances. The authors conclude that the vehicle indexed formulations are superior to more compact flow formulations. The branch-and-cut algorithm developed by Coelho and Laporte (2014) was the best one in terms of its ability to solve larger problem instances until the branch-price-and-cut algorithm of Desaulniers et al. (2016). The latter study introduced an extended IRP formulation based on "route delivery patterns", which was inspired by a previously proposed model for the split delivery vehicle routing problem by Desaulniers (2010). Recently, Avella et al. (2017) introduced a generic family of valid inequalities based on the single period substructure of the problem and solved the IRP by a branch-and-cut algorithm. This algorithm outperformed the algorithms of Desaulniers et al. (2016) and Coelho and Laporte (2014) on the benchmark instances.

All the studies mentioned above assumes that there are no time window restrictions of the retailers, i.e. a retailer could be visited at any time of the day. This assumption is rarely satisfied in practice. For instance, a supermarket that receives products from many different suppliers has to manage the operations in its loading dock and may not accept deliveries at arbitrary times. There are also restrictions imposed

by local authorities concerning the entry of delivery vehicles into certain zones during busy hours of traffic. Consequently, the supplier has to take into account these restrictions in planning the timing of the deliveries. This can be achieved by introducing time windows to the IRP.

Even though heuristic approaches have been proposed for variants of the IRP with time windows (IRPTW) (see, e.g., Liu and Lee (2011)), we are not aware of any exact algorithm for this problem. In this study, we develop a branch-price-and-cut algorithm to solve the IRPTW. We formulate the problem using both arc and route based variables to obtain an easier structure in the pricing problem. We model the capacity restrictions using an exponential family of constraints. Moreover, we use the valid inequalities derived for the lot sizing problem to improve our formulation. We generate a set of benchmark instances for the IRPTW based on the available instances for the vehicle routing problem with time windows (VRPTW) and the IRP, which can be seen as special cases of the IRPTW. The computational results reveal that our algorithm is able to solve instances including up to 50 retailers and 6 periods optimally, and produce satisfactory solutions to the instances for which an optimal solution cannot be obtained within the specified time limit.

The remainder of this paper is organized as follows. In Section 2 we provide a formal definition of the IRPTW and introduce our formulation. We present the valid inequalities for this formulation in Section 3, and describe the proposed branch-price-and-cut algorithm in detail in Section 4. The results of our computational experiments are given in Section 5, followed by the concluding remarks in Section 6.

2 Problem Definition and Formulation

Let n be the length of the planning horizon $T = \{1, \ldots, n\}$. In each period t, the supplier, denoted by 0, produces or receives a delivery of d_{0t} units of the product, and uses a homogeneous fleet of vehicles with capacity Q to supply a set of retailers V. The vehicles are unlimited in number, but there is a fixed cost f associated with using a vehicle. The demand d_{it} of retailer $i \in V$ in period $t \in T$ should be satisfied on time. Each retailer $i \in V$ has a stock capacity u_i, an initial inventory of $\bar{s}_i$, and unit inventory holding cost of h_{it} for period $t \in T$. The deliveries must be planned so that retailer $i \in V$ is visited within the time interval $[e_i, l_i]$ where $e_i \leq l_i$. Here, we assume that the time windows are time-invariant, i.e., the delivery time window of a retailer does not change from one period to another. This is a reasonable assumption to facilitate planning for both the supplier and the retailers. The supplier has an initial stock of $\bar{s}_0$, and a unit inventory holding cost of h_{0t} for period $t \in T$. We assume that $h_{0t} \leq h_{it}$ for all $i \in V$ and $t \in T$. To simplify the presentation, let $T_0 = T \cup \{0\}$ and $V_0 = V \cup \{0\}$.

The distribution network is represented by a directed graph $G = (V_0, A)$ where $A = \{(i, j) : i, j \in V_0, i \neq j\}$ is the set of arcs. The travel cost and the travel time for arc $a \in A$ are denoted by c_a and τ_a, respectively.The vehicle routes for all periods should be determined by the supplier. Each retailer can be visited at most once in any given period. The problem is to determine the retailers that will be served in each period, the amounts to be delivered to them, and the routes of the vehicles that minimize the total storage and transportation costs.

We propose a formulation –involving both arc and route based variables– for the IRPTW. As the

delivery quantities are not known in advance, we call a route feasible if it starts and ends at the supplier, visits each retailer at most once, and satisfies the time window restrictions of the retailers in the route.

To formulate the problem, we define the following decision variables.

s_{it}: amount of inventory at node $i \in V_0$ at the end of period $t \in T_0$

q_{it}: amount shipped from the supplier to retailer $i \in V$ in period $t \in T$

z_r^t: 1 if route $r \in R$ is used in period $t \in T$ and 0 otherwise

x_a^t: 1 if arc $a \in A$ is traversed in period $t \in T$ and 0 otherwise.

For any subset $S \subseteq V$, let $\delta^+(S)$ be the set of outgoing arcs, i.e., arcs (i,j) with $i \in S$ and $j \in V_0 \setminus S$, and $\delta^-(S)$ be the incoming arcs, i.e., arcs (i,j) with $i \in V_0 \setminus S$ and $j \in S$. For ease of notation, we use $x^t(A') = \sum_{a \in A'} x_a^t$ for $A' \subseteq A$, $\delta^-(i) = \delta^-(\{i\})$ and $\delta^+(i) = \delta^+(\{i\})$ for $i \in V_0$.

The IRPTW can be formulated as follows:

$$\min \sum_{i \in V_0} \sum_{t \in T} h_{it} s_{it} + \sum_{t \in T} \sum_{r \in R} (f + \sum_{a \in r} c_a) z_r^t \tag{1}$$

$$\text{s.t. } s_{0,t-1} + d_{0t} = \sum_{i \in V} q_{it} + s_{0t} \qquad t \in T \tag{2}$$

$$s_{i,t-1} + q_{it} = d_{it} + s_{it} \qquad i \in V, t \in T \tag{3}$$

$$s_{i0} = \bar{s}_i \qquad i \in V_0 \tag{4}$$

$$s_{it} + d_{it} \leq u_i \qquad i \in V, t \in T \tag{5}$$

$$x^t(\delta^-(i)) = x^t(\delta^+(i)) \leq 1 \qquad i \in V, t \in T \tag{6}$$

$$Q_S^t x^t(\delta^-(S)) \geq \sum_{i \in S} q_{it} \qquad S \subseteq V, t \in T \tag{7}$$

$$\sum_{r \in R: a \in r} z_r^t = x_a^t \qquad a \in A, t \in T \tag{8}$$

$$s_{it} \geq 0 \qquad i \in V_0, t \in T \tag{9}$$

$$q_{it} \geq 0 \qquad i \in V, t \in T \tag{10}$$

$$x_a^t \in \{0,1\} \qquad a \in A, t \in T \tag{11}$$

$$z_r^t \in \{0,1\} \qquad r \in R, t \in T \tag{12}$$

where $Q_S^t = \min\{Q, \sum_{i \in S} \min\{u_i, \sum_{t'=t}^n d_{it'}\}\}$ for $S \subseteq V$ and $t \in T$. Note here that there exists an optimal solution with $s_{in} = 0$ for all $i \in V$ since $h_{0t} \leq h_{it}$ for $t \in T$. Such a solution satisfies $q_{it} \leq \min\{u_i, \sum_{t'=t}^n d_{it'}\}$ for $i \in V$ and $t \in T$.

The objective function (1) minimizes the total inventory holding and routing costs. Constraints (2) and (3) are the inventory balance equations for the supplier and the retailers, respectively. Constraints (4) define the initial stock levels at each node. Constraints (5) impose the inventory upper bounds at each retailer. Constraints (6) ensure that each retailer is visited at most once per period. Constraints (7) are the capacity constraints: for any subset $S \subseteq V$, the total amount sent to the retailers in S cannot

be larger than the total capacities of the vehicles that visit these retailers. Here, we strengthened this constraint by replacing the capacity Q with Q_S^t. Constraints (8) relate the z and x variables, and variable domain restrictions are given by (9)–(12).

Formulation (1)–(12) involves an exponential number of variables and constraints. Hence, we devise a branch-price-and-cut algorithm to solve it efficiently. Note that because we use arc based variables to formulate the constraints (7), the resulting pricing problems have a less complicated structure. Moreover, since the formulation contains the lot sizing problem as a subproblem for each retailer, we use the valid inequalities derived for the lot sizing problem to improve our formulation. Before going into the details of our branch-price-and-cut algorithm, we first present these inequalities in the next section.

3 Valid Inequalities

Since the IRP contains the lot sizing problem (LS) as a subproblem, valid inequalities derived for LS can be utilized to strengthen the formulation (see, e.g., Solyalı and Süral (2011), Avella et al. (2015), and Mahmutoğulları and Yaman (2019)).

Pochet and Wolsey (1994) describe the convex hull of the feasible solutions for the LS problem with Wagner-Whitin costs using $O(n^2)$ inequalities and variables. Although the inequalities (will be referred to as WW inequalities from now on) derived by the authors are not sufficient to describe the polyhedra associated with more complex LS problems, they can still improve the initial formulations. The modified non-trivial WW inequalities for our problem are as follows:

$$\sum_{t=k}^{l} \sum_{a \in \delta^-(i)} x_a^t \geq \lceil * \rceil \frac{\bar{d}_{kl}^i}{\min\{Q, u_i\}} \qquad i \in V, 1 \leq k \leq l \leq n \; s.t. \; d_{kl}^i > u_i \tag{13}$$

where $\bar{d}_{1l}^i = d_{1l}^i - \bar{s}_i$, $\bar{d}_{kl}^i = d_{k-1,l}^i - u_i$ if $k \geq 2$, and $d_{kl}^i = \sum_{t=k}^{l} d_{it}$. The right-hand-side of (13) represents the minimum number of visits that should be made to retailer i to satisfy the demand between periods k and l, thus, inequality (13) defines a lower bound on the number of visits to i between every pair of periods k and l. Since the number of inequalities (13) is $O(n^2|V|)$, they can be added to our formulation a priori. Similar inequalities have also been proposed by Archetti et al. (2007) and Coelho and Laporte (2014).

Due to the assumption on stock bounds in the IRP literature, and consequently constraints (5), the inventory upper bound for retailer i in period t is $u_i - d_{it}$. Atamtürk and Küçükyavuz (2005) study LS with stock upper bounds and develop two classes of facet defining inequalities which are also valid for our problem. The valid inequalities of Atamtürk and Küçükyavuz (2005) modified for our problem are as follows:

$$\sum_{t \in S} q_{it} \leq \sum_{t \in S} \min\{d_{1,t-1}^i + u_i - \bar{s}_i, d_{1l}^i - \bar{s}_i, d_{tl}^i\} \sum_{a \in \delta^-(i)} x_a^t + s_{il} \tag{14}$$

$$S \subseteq [1, l], \; l \in T \; s.t. \; d_{1l}^i > \bar{s}_i, \; i \in V$$

$$s_{i,k-1} + \sum_{t \in S} q_{it} \leq u_i - d_{i,k-1} + \sum_{t \in S} \min\{d_{k-1,t-1}^i, d_{k-1,l}^i - u_i, d_{tl}^i\} \sum_{a \in \delta^-(i)} x_a^t + s_{il} \tag{15}$$

$$S \subseteq [k,l],\ 2 \le k \le l \le n\ s.t.\ d^i_{k-1,l} > u_i,\ i \in V$$

$$\sum_{t\in S} q_{it} \le \sum_{t\in S}(d^i_{k,t-1} + u_i - \bar{s}_i) \sum_{a\in\delta^-(i)} x^t_a \qquad S \subseteq [1,n],\ i \in V \tag{16}$$

$$s_{i,k-1} + \sum_{t\in S} q_{it} \le u_i - d_{i,k-1} + \sum_{t\in S} d^i_{k-1,t-1} \sum_{a\in\delta^-(i)} x^t_a \qquad S \subseteq [k,n],\ 2 \le k \le n,\ i \in V \tag{17}$$

Inequalities (14)–(17) introduce upper bounds on the maximum amount that can be sent to a retailer $i \in V$ by setting the inventory level at the retailer to its upper bound $u_i - d_{i,t}$ at some periods, i.e., at the end of period $k-1$ in inequalities (14)–(15), and at the end of periods $k-1$ and l in inequalities (16)–(17). Considering the large number of inequalities (14)–(17), one can try to add a small subset of them, or use a cutting plane approach to separate the violated ones on the fly. The separation problem for these inequalities will be described in the next section.

The (l, S) inequalities of Barany et al. (1984) derived for the uncapacitated LS problem are also valid for our problem, but they are dominated by the inequalities above, and we choose not to present them here as they are not used in our computations.

Inequalities (13)–(17), which are due to demand satisfaction and capacity constraints, can be written for a subset of time periods for each retailer separately. In addition to the LS substructure of our problem, the single period subproblem of (1)–(12) includes a special case of the single node flow set, and therefore flow cover inequalities of Pochet and Wolsey (1994), and Van Roy and Wolsey (1986) can also be employed to improve the formulation. Since we do not use them in our experiments, they are not provided here either.

4 Branch-price-and-cut algorithm

As mentioned earlier, there are exponentially many variables and constraints in the formulation (1)–(12). Therefore, we propose a branch-price-and-cut approach to solve it, that is, we iterate between a column generation algorithm and a cutting plane algorithm to solve the LP relaxation –referred to as the master problem (MP)– at each node of the branch-and-bound tree.

We include the capacity constraints (7) only for singletons (for $|S| = 1$) in the initial LP relaxation:

$$\min\{Q, u_i, \sum_{t'=t}^{n} d_{it'}\} x^t(\delta^-(i)) \ge q_{it} \qquad i \in V, t \in T \tag{18}$$

Since $z^t_r \le 1$ for $r \in R, t \in T$ is implied by $x^t_a \le 1$ for $a \in A, t \in T$, we drop them from the LP relaxation.

At each node of the solution tree, we first check if the node can be pruned by bound. If not, we start the column and cut generation procedures to solve the relaxation. If the master problem is infeasible (due to the constraints added to the model to enforce branching decisions), we prune the node by infeasibility. Otherwise, we continue with column generation. When the pricing algorithm does not return any negative reduced cost columns, we check whether it is possible to prune the node by bound. If not, we move on to

the cutting plane phase. If one or more violated cuts are identified, we add them and return to column generation. Else, we check if the solution is integer. If yes, we prune the node by optimality. Otherwise, we branch. In the sequel, we explain how we generate columns and cuts as well as our branching scheme.

4.1 Column generation

Let $\bar{R} \subset R$ be a subset of the routes for which there exists a feasible solution to the MP with $z_r^t = 0$ for all $r \in R \setminus \bar{R}, t \in T$. The LP relaxation of the formulation (1)–(12) that includes only the routes in $\bar{R}$ is called the restricted master problem (RMP). As mentioned earlier, we relax the capacity constraints (7) except the ones with $|S| = 1$, and add them on the fly. Therefore, with respect to the MP at a given node of the solution tree, we are in fact solving a relaxation during each column generation phase. Let RMP-R denote the relaxed RMP, which includes only the capacity constraints that have been separated so far.

At each iteration of the column generation phase, first the RMP-R is solved, and using the optimal dual solution, we check for every period $t \in T$ whether there exists a variable z_r^t (a route) with negative reduced cost. This is also known as a column generation iteration or a pricing iteration, and we will use these two terms interchangeably from this point forward. If we associate the dual variable vector σ with the set of constraints (8), the pricing problem, for each $t \in T$, seeks to find (if any) $r \in R \setminus \bar{R}$ such that

$$\sum_{a \in r} w_a^t < 0 \tag{19}$$

where

$$w_a^t = \begin{cases} c_a - \sigma_a^t + f & \text{if } a \in \delta^+(0) \\ c_a - \sigma_a^t & \text{otherwise} \end{cases} \qquad \text{for } a \in A.$$

Thus, the pricing problem for period t is an elementary shortest path problem with time window constraints (ESPPTW), where the cost of arc a is set to w_a^t, i.e., the goal is to find an elementary path of shortest length starting and ending at the depot and respecting the time window restrictions.

4.1.1 Solving the pricing problems

To solve the pricing problems efficiently, we employ the state-space augmenting technique proposed by Boland et al. (2006), which is essentially an iterative label setting approach. Given a directed graph with arbitrary arc lengths and a specified pair of source and sink nodes s and t, the elementary shortest path problem with resource constraints is the problem of finding the shortest elementary resource-feasible path from s to t. In the ESPPTW, we have a single resource arising from the time window restrictions. Hence, an elementary resource-feasible path in our case is one that is free of cycles (i.e., elementary) and that respects the availability of time resource. A common way to ensure that elementary paths are obtained in a label setting algorithm is to use node-visit resources, which were introduced by Feillet et al. (2004). A node-visit resource can be considered as a resource having unit capacity, which is saturated upon a visit to the associated node.

Label setting is a dynamic programming approach which constructs resource-feasible paths originating at the source node s and ending at the sink node t. Starting with the trivial partial path $\{s\}$, the unprocessed partial paths are extended along all feasible arcs to create new (partial) paths. The extension of a partial path along an arc is infeasible if the resulting path would not be resource-feasible or if it cannot be augmented to reach the sink node within the available resource limits. The efficiency of a label setting algorithm highly depends on the use of *dominance rules* that are helpful in eliminating those partial paths that cannot appear in any optimal path, and storing and extending only the so-called *efficient* or *non-dominated* partial paths.

Every partial path is linked with a state determined by its cost and the resource consumption along the path. The state corresponding to a partial path is represented using a label, i.e., a vector that has as many components as the number of resources plus one (to store the cost). Hence, the size of the state-space increases with the number of resources.

In order to limit the size of the state-space and to accelerate the solution procedure, we start with a state-space relaxation in which node-visit resources are initially not present, as suggested in Boland et al. (2006). After a label setting iteration is completed, if an elementary optimal path is found, it means that the problem is solved to optimality. Otherwise, we compute the number of times each node is visited in an optimal path, and introduce a node-visit resource for the node that takes the largest number of visits among others. The label setting procedure is iteratively executed, each time starting with the original resources and all the node-visit resources introduced during the previous iterations, until an elementary optimal path is returned at the end of an iteration. The advantage of this state-space augmenting approach is that the number of node-visit resources actually required to obtain a least-cost elementary path is usually much smaller than the number of nodes in the graph.

4.1.2 Heuristic pricing

Although employing a straightforward implementation of the state-space augmenting technique can be more efficient compared to using an off-the-shelf software to solve the pricing problems, it may still be (too) time-consuming. Fortunately, to solve the master problem, it is not necessary to solve the pricing problems optimally at every column generation iteration; finding (at least one) negative reduced cost column suffices –as long as one exists. Returning any negative reduced cost column rather than a most-negative one for each pricing problem may lead to a smaller improvement in the objective value of the RMP-R, and consequently, more column generation iterations may have to be performed to achieve optimality. Nonetheless, the reduction in solution time in each pricing step typically reduces the overall computation time. Identifying most negative reduced cost columns is usually needed only towards the end of a column generation phase, because few, if any, negative reduced cost columns exist at that time. Hence, in our implementation, we attempt to solve a pricing problem optimally only if no negative reduced cost columns can be detected heuristically, and even in that case, we terminate the state-space augmenting algorithm as soon as a pre-specified number of negative reduced cost columns (for that pricing problem) is obtained.

The above idea is easy to apply thanks to the constructive nature of label setting. In general, when

solving a pricing problem, many elementary negative cost paths are discovered during the first few label setting iterations. We collect elementary negative cost paths found in each iteration and stop the algorithm prematurely, i.e., possibly before an optimal path has been identified, and return all elementary negative cost paths accumulated up to that point. Another useful idea to speed up column generation stems from the following observation: as the dual values are updated after each pricing iteration, some elementary paths with a non-negative cost in one iteration may have a negative cost in a subsequent iteration. Therefore, at the end of every column generation iteration, we store, in a column pool, all non-dominated elementary paths with non-negative costs constructed in the last label setting iteration performed during the solution process of each pricing problem. At the beginning of the next column generation iteration, we first evaluate the columns in the pool to see if they (now) have a negative reduced cost with respect to their associated pricing problems. We limit the size of the column pool to ensure that it does not become too costly to explore. Every time new columns are added to the pool after it becomes full, the oldest columns are removed until the desired pool size is reached.

We also implement a truncated version of the state-space augmenting approach, which essentially serves as a heuristic to detect negative reduced cost columns quickly. Rather than maintaining all non-dominated labels and their associated partial paths, we keep only a small number of such labels at each node to accelerate the search procedure. In particular, we allow storing up to a pre-specified number of efficient labels per node. Once the number of labels at any node hits the limit, every time a new efficient label with smaller cost compared to at least one of the existing labels at that node is detected, it replaces the one having the largest cost. This can facilitate faster detection of negative cost elementary paths by means of reducing the number of labels treated at each label setting iteration. Note that excluding a part of the search space may come at the expense of performing more iterations, i.e., there is a trade-off between the number of efficient labels maintained and the number of label setting iterations performed by the state-space augmenting approach.

A column generation/pricing iteration can be summarized as follows: for each pricing problem, we try to identify negative reduced cost columns first by exploring the column pool, then by invoking the truncated-search version of the state-space augmentation algorithm, and finally by invoking the full-search version of the state-space augmentation algorithm when the other approaches fail to detect *any* negative reduced cost column. To control the time spent in column generation iterations even further, we terminate the solution process of any pricing problem as soon as a predetermined number of elementary negative cost paths (in regard to that pricing problem) has been found. Based on some initial experiments, our parameter choices are as follows: when solving the ESPPTW corresponding to any period, (1) we terminate the state-space augmenting algorithm as soon as five negative reduced cost columns are constructed, (2) we store at most five labels in truncated-search mode, and (3) we limit the size of the column pool by 1000.

We would like to note here that when applying full-search version of the state-space augmentation algorithm to solve the pricing problem associated with period $t \in T$, we keep track of the minimum cost of the elementary paths detected during the search, which gives an upper bound on the optimal value of the ESPPTW, and the cost of the optimal path obtained at the end of each label setting iteration, which

gives a lower bound on the optimal value of the ESPPTW. Once the gap between these bounds is closed, we can conclude that the pricing problem corresponding to period t has been solved optimally, i.e., there is no need to continue with label setting iterations thereafter.

Finally, as mentioned earlier, the state-space augmenting algorithm starts without any node-visit resources. However, the set S of *critical nodes*, i.e., the nodes for which a node-visit resource have been introduced during a pricing iteration, are likely to be visited more than once in subsequent pricing iterations, and initializing the algorithm with no node-visit resources (i.e., with $S = \emptyset$) at every pricing iteration may therefore result in an increase in the number of label setting iterations. To prevent this issue, we retain S throughout the column generation process at each node of the search tree, and clear it only after the master problem solve at the node is completed as done in Ozbaygin et al. (2017).

4.2 Cutting planes

When the column generation phase terminates, the relaxation of the MP (which includes only a subset of inequalities (7)) is solved to optimality. Provided that the optimal value is lower than the best upper bound found so far, the next step is to check whether the optimal solution of the relaxed MP satisfies the rest of the capacity constraints, i.e., the inequalities (7) that are excluded from the MP. To achieve this, we solve a separation problem which either establishes that no violated capacity constraint exists, or identifies and returns at least one such inequality.

The separation problem decomposes into n minimum cut problems, one for each $t \in T$. In particular, given the optimal values of the variables x and q obtained at the end of a column generation phase, for every period t, we construct the corresponding support graph $G^t = (V_0 \cup \{0'\}, A^t)$, where $0'$ is a dummy node, and $A^t = \{a \in A : x_a^t > 0\} \cup \{(i, 0') : i \in V\}$ is the set of arcs. We set the capacity of arc $(i, 0')$ with $i \in V$ to q_{it} and the capacity of the remaining arcs a to Qx_a^t. The capacity of the minimum cut separating the nodes 0 and $0'$ on this graph is given by

$$Q \sum_{a \in \delta^-(S)} x_a^t + \sum_{i \in V \setminus S} q_{it}.$$

If we add $\sum_{i \in V \setminus S} q_{it}$ to both sides of a weaker version of the capacity constraints (7) where Q_S^t is replaced by Q, we get

$$Q \sum_{a \in \delta^-(S)} x_a^t + \sum_{i \in V \setminus S} q_{it} \geq \sum_{i \in V} q_{it} \qquad S \subseteq V, t \in T.$$

Therefore, if the capacity of the minimum cut of the graph G^t –separating the nodes 0 and $0'$– is greater than or equal to $\sum_{i \in V} q_{it}$, then no violated constraints are identified for period t. Otherwise, we add the inequality (7) defined by the nodes in S, where $0' \in S$ and $(V_0 \setminus S, S)$ is the node partition corresponding to the minimum cut.

Separation of the capacity constraints (7) must be performed in order to end up with a solution that is feasible for the MP. However, when no violated capacity cuts are detected at a cutting plane iteration, we can still look for and add other valid inequalities to strengthen the relaxed MP. To this end, we also consider the separation of inequalities (14)–(17), which can be easily solved by inspection for each

retailer i and period pair (k, l) that satisfies the conditions given in (14)–(17). For example, to separate inequalities (14), given a solution $(\tilde{x}, \tilde{q})$, if $\tilde{q}_{it} > \min\{d^i_{1,t-1} + u_i - \bar{s}_i, d^i_{1l} - \bar{s}_i, d^i_{tl}\} \sum_{a \in \delta^-(i)} \tilde{x}^t_a$, then t will be added to the set S for $t = 1, \ldots, l$.

As long as at least one violated inequality is found during the cutting plane phase, it is added to the relaxed MP, which is then solved by re-invoking the column generation procedure. Otherwise, the solution at hand is optimal for the MP.

4.3 Initializing RMP-R at the root node of the search tree

A delivery schedule where all retailers are visited in each period is feasible with respect to the IRPTW as long as the number of vehicles is large enough. Such a delivery schedule can be constructed by finding a feasible solution to the VRPTW given the demands and the time windows of the retailers, and repeating it in each period (since the demands and the time windows are time-invariant). Although this solution may be far from optimum as it does not consider the possibility of delivering altogether an amount that would meet the demand at a given retailer for several periods, we adopt this approach to provide our algorithm with a starting solution. To find a feasible VRPTW solution, we use the free software Heuristiclab (Klempous et al., 2014).

4.4 Branching

The optimal solution of the master problem at a given node of the solution tree may be fractional. In that case, we need to branch, and update the master problem as well as the pricing problems at the child nodes according to the branching decisions.

Traditionally, branching is performed on the arc flow variables in vehicle routing problems, and it usually works well in a single period setting. However, since the planning horizon spans multiple periods in inventory routing, branching on the arc flow variables may slow down the convergence of the algorithm. Hence, we devise a hierarchical branching scheme in which we first attempt to branch on the retailers, and then on the arcs. In particular, we first check if there exists $i \in V$ and $t \in T$ with fractional $x^t(\delta^-(i))$ value. If we can find such i and t, then we define the child nodes by setting $x^t(\delta^-(i)) = 1$ in one branch, and $x^t(\delta^-(i)) = 0$ in the other. Otherwise, we look for $a \in A$ and $t \in T$ for which x^t_a is fractional. If such an arc is identified, we define one child node with $x^t_a = 1$, and another with $x^t_a = 0$. Our preliminary analyses showed that the branch-price-and-cut algorithm converges much faster using this hierarchical strategy, which is therefore much more efficient compared to the conventional arc branching.

When a branching decision is implemented at a node of the solution tree, the columns and the cuts that are present in the final RMP-R associated with that node are inherited to its children after a filtering step. The cuts (i.e. constraints (7)) are globally valid, i.e., they remain valid throughout the entire tree. Hence, they are passed to the child nodes without any filtering. However, for a child node, there may be columns in the last solved RMP-R that do not comply with the branching decision defining the child node. We note here that an important advantage of using arc based variables for branching is that even though we may have to modify the master problem to enforce branching decisions, the structure of the

pricing problems does not change.

To ensure compatibility with a *fix retailer* type of branching decision, i.e., $x^t(\delta^-(i)) = 1$ for some $i \in V$ and $t \in T$, we do not have to eliminate any columns coming from the parent, but we add $x^t(\delta^-(i)) = 1$ as a constraint to the master problem of the corresponding child node. Otherwise, it is not possible to force the retailer i to be visited in period t. Notice that in this case, there is no need to modify the pricing problems, because all columns generated at any pricing iteration will comply with the branching decision.

On the other hand, if a *remove retailer* type of branching decision is executed, i.e. $x^t(\delta^-(i)) = 0$, the resulting child node should not contain any columns in its master problem where retailer i is visited in period t. Therefore, such columns are not inherited from the parent node. Moreover, in the pricing problem corresponding to period t, we eliminate all incoming arcs of retailer i so that no columns containing i in period t are generated during the solution process. Consequently, there is no need to include the constraint $x^t(\delta^-(i)) = 0$ in the master problem.

If all retailers in every period have integer inflow in a fractional solution, we choose an arc a and a period t with fractional x_a^t value. For the child node defined by the *fix arc* branching decision, i.e., $x_a^t = 1$, the columns that are inherited from the parent node are those that use arc $a = (i, j)$ in period t, or that do not visit either of the nodes i and j in period t. Moreover, in the pricing problem corresponding to period t, we get rid of all outgoing arcs of node i and all incoming arcs of node j except for the arc (i, j) itself. In this way, all columns generated will be consistent with the branching decision. Nevertheless, to guarantee that the vehicle flow on arc a is equal to 1 in period t, we also have to introduce the restriction $x_a^t = 1$ to the master problem.

The way in which we ensure compatibility in case of a *remove arc* type of branching decision is similar to that for *remove retailer* decisions. More specifically, we do not pass any columns that involve arc a in period t to the child node, we eliminate arc a when solving the pricing problem of period t, and there is no need to add $x_a^t = 0$ to the master problem explicitly.

The rule based on which the retailer/arc and the period for branching is selected can also have a significant impact on the performance of the branch-price-and-cut algorithm. A widely adopted strategy is to select the most fractional variable, i.e, the one with value closest to 0.5. We employ a different approach, which usually performed better in our preliminary experiments. Given a fractional solution, we compute, for each retailer (arc) and each period, the number of columns containing that retailer (arc). We select the period, among others, in which a retailer (an arc) appears most frequently, and that particular retailer (arc). In case of ties, we choose retailer i (arc a) and period t for which the value $x^t(\delta^-(i))$ (x_a^t) is closest to 0.5. If there are still multiple options, we break ties by picking the last retailer (arc) and period pair encountered during our search.

4.5 Artificial columns

When a branching decision is executed and a new child node is defined accordingly, the columns inherited from the final RMP-R of the parent node (after filtering) may not be sufficient to guarantee the existence of a feasible solution to the initial RMP-R associated with the child node. To supply the initial RMP-

R with a starting solution, we introduce a set of artificial columns, which depends on the sequence of branching decisions leading to the current node. The reason why we cannot use the same set of artificial columns at every node of the solution tree is that the master problem at distinct nodes can contain different constraints (added to enforce different branching decisions).

At the root node, we define an artificial column of the form $0 - i - 0$ for each retailer $i \in V$ and for each period $t \in T$. At deeper nodes of the tree, every time a *fix arc* decision is executed, e.g. to force arc (i, j) to be used in period t, we pass the artificial columns coming from the parent node, merge the two columns containing i and j (for period t) along the arc (i, j), and use the rest as is. A small example with four retailers and a single period is provided in Figure 1 below where the set of routes next to each node of the solution tree represents the artificial columns added to the initial RMP-R associated with that node.

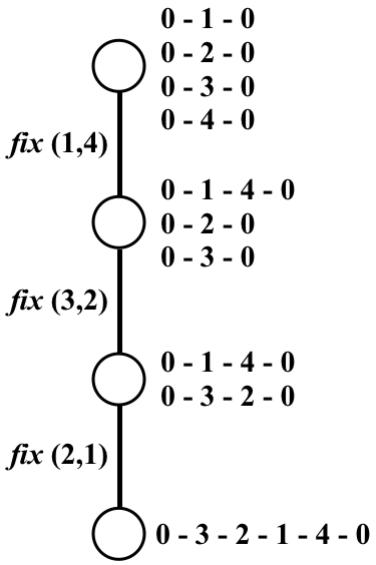

Figure 1: Artificial columns for an example with four retailers and a single period

It is important to note here that with the addition of branching related constraints (to fix retailers/arcs), the master problem associated with some nodes of the tree can turn out to be infeasible. In those situations, artificial columns do not help, and we prune the node by infeasibility. Otherwise, the artificial columns guarantee the existence of a starting basis, and they never appear in the final solution since we assign a very high cost to those columns.

5 Computational Experiments

In this section, we report the detailed results of our computational study with the proposed branch-price-and-cut algorithm (BP&C) on a large set of newly generated IRPTW instances.

All tests were performed on a virtual machine running Intel Xeon processor E5-2640 v4 at 2.40 GHz

with 16GB of RAM. The algorithm have been implemented in Java using the libraries JGraphT (Michail et al. (2020)) and jORLib (Kinable et al. (2016)), and CPLEX 12.7. We allow at most four threads to run in parallel during certain steps of the algorithm. In particular, at a column generation iteration, jORLib can parallelize the solution of pricing problems depending on the number of threads available. Furthermore, we let CPLEX use up to four threads whenever it is invoked. Notice that this does not mean all four threads will be employed the entire time, jORLib manages the search tree and parallelization is enabled when solving the pricing problems. Finally, we adopt a depth-first search strategy in our implementation.

Next, we present the specifications of the problem instances we used in our computational experiments.

5.1 Test Instances

The test instances are generated on the basis of the VRPTW instances of Solomon (1987) and the IRP instances of Archetti et al. (2007).

The x and y coordinates, demand, and time window of each node are taken from $R105$, $R109$ and $R201$ instances of Solomon (1987). These three classes of problem instances have different time window characteristics as it can be seen from Table 1: $R105$ and $R109$ have shorter scheduling horizons (230 for both) compared to $R201$ (1000). Different from Solomon (1987), we assume that the service times are equal to zero for all retailers. To compare the time window restrictions and determine the difficulty levels of these instance classes, we compute the ratio $\frac{\sum_{t=1}^{TW}\gamma_t^+}{(|V|-1)TW}$, where TW is the length of the tightest time window that covers the time windows of all retailers, $\gamma_t^+ = max\{0, \gamma_t - 1\}$ with γ_t being the number of retailers whose time windows include time point t. We call this ratio as the *time window density*, which essentially provides an indication of how much the time windows of the nodes in a problem instance overlap. For example, if there exist three retailers with time windows $[1, 10]$, $[5, 10]$ and $[3, 8]$, then the time window density of this instance will be $10/20 = 0.50$. If the time windows were $[1, 5]$, $[5, 10]$ and $[10, 15]$, the density would be 0, whereas for $[1, 5]$, $[1, 5]$ and $[1, 5]$, it would be 1. Based on the time window densities of these three instance classes given in Table 1, we argue that the IRPTW instances generated based on $R109$ are much harder than the others, which, as we shall see later, is also supported by our computational results presented in the next two subsections. Note that our algorithm can also solve the inventory routing problem without time windows by defining large enough time windows. However this does not make much sense as one can use a more effective pricing algorithm if time windows do not exist.

	$R105$	$R109$	$R201$
Time window of the supplier	$[0, 230]$	$[0, 230]$	$[0, 1000]$
Minimum time window length	30	37	27
Average time window length	30	59	116
Maximum time window length	30	83	212
Time window density for 100 retailers	0.15	0.28	0.11

Table 1: Time window characteristics of different problem instances

Similar to Archetti et al. (2007) and Desaulniers et al. (2016), the demand of each retailer is assumed to be constant over time, i.e. $d_{it} = d_i, t \in T, i \in V$. The capacity of each vehicle is given by $\lceil \frac{3}{2m} \sum_{i \in V} d_i \rceil$ for $m \in \{2, 3, 4\}$ in $R105$ and $R109$ instances. Since $R201$ instances have longer scheduling horizons, we replaced the multiplier $\frac{3}{2}$ by $\frac{5}{2}$ in these instances to ensure that the vehicle capacity is not too restrictive relative to time window constraints, thereby allowing more retailers to be visited in the same route as long as their time windows permit. We assume that there is no limit on the number of vehicles available at the supplier, and the fixed cost of using each vehicle is zero. The latter assumption is based on the observation that regardless of whether the fixed cost of using a vehicle is taken to be a positive number or zero, the solutions we obtain did not change in our tests, mainly because the transportation costs satisfy the triangle inequality. The transportation costs and travel times are taken to be equal to one another, and computed by solving all pairs shortest path problem on a complete directed graph in which arc lengths correspond to Euclidean distances rounded to nearest integer values.

A quantity equal to the total demand of all retailers is made available at the supplier at the beginning of each time period, i.e., $d_{0t} = d_0 = \sum_{i \in V} d_i$, $t \in T$. The inventory upper bound for each retailer is randomly generated by setting $u_i = \alpha_i d_i$ and selecting α randomly from the set $\{2, 3\}$ or $\{1, 2, 3\}$. The initial stock levels are set to $\bar{s}_0 = \sum_{i \in V} u_i$ at the supplier, and $\bar{s}_i = u_i - d_i$ at the retailers. The unit cost of holding inventory at retailer i is randomly generated in the intervals $[0.03, 0.05]$ -*low*- and $[0.3, 0.5]$-*high*-, and is set to 0.03 and 0.3 at the supplier for these intervals, respectively. Note that, in this way, we ensure that $h_0 \leq h_i, i \in V$, which implies that keeping inventory at the supplier is at least as cheap as keeping the same inventory at the retailers. Thus, we know that there exists an optimal solution to the IRPTW such that the ending inventory level at each retailer is zero at the end of the last period.

Each instance is labeled as $Y - n - \{l, h\} - m - ib$, where Y is the number of retailers, n is the number of periods, the attribute $\{l, h\}$ denotes the level of the inventory holding costs (low or high), m is an integer determining the vehicle capacity, and $ib \in \{1, 2\}$ is an attribute indicating the inventory upper bound at the retailer: 1 and 2 stand for the sets $\{2, 3\}$ or $\{1, 2, 3\}$, respectively.

The instances with $ib = 1$ have the same inventory bound settings with Archetti et al. (2007), in which the inventory bound (u_i) of each retailer is selected randomly from the set $\{2d_i, 3d_i\}$ and the initial inventory level ($\bar{s}_i$) is set to $u_i - d_i$. However, different from Archetti et al. (2007), we assume that sufficiently many vehicles are available at the supplier location. Therefore, in these instances, first period's demand can be satisfied from the inventory at each retailer, and there exists an optimal solution in which no delivery takes place within the first period since the inventory holding costs are positive, the transportation costs are time-invariant, and the vehicles are not limited in number. Actually, this means that an n period problem instance can be reduced to an $n - 1$ period problem instance when $ib = 1$. To consider a more general setting, we introduce another instance class with $ib = 2$ where the inventory bound of each retailer is selected randomly from the set $\{d_i, 2d_i, 3d_i\}$, i.e., there may be retailers with zero initial inventory and they have to be visited in every period.

All test instances generated for the IRPTW are available upon request from the corresponding author.

5.2 Results for Different Enhancements

First, we carried out an experiment in which several ideas to enhance the performance of our BP&C algorithm are implemented and tested on a set of small IRPTW instances so that optimal or near optimal solutions can be obtained for larger instances in reasonable computation times.

We experimented with the WW inequalities (13) for each retailer and every pair k, l of periods, the most violated inequalities of (14)–(17) for each retailer (only at the root node of the solution tree), and the (l, S) inequalities derived for the lot sizing problem. Among these, the utmost improvements were observed by the inclusion of inequalities (14)–(15).

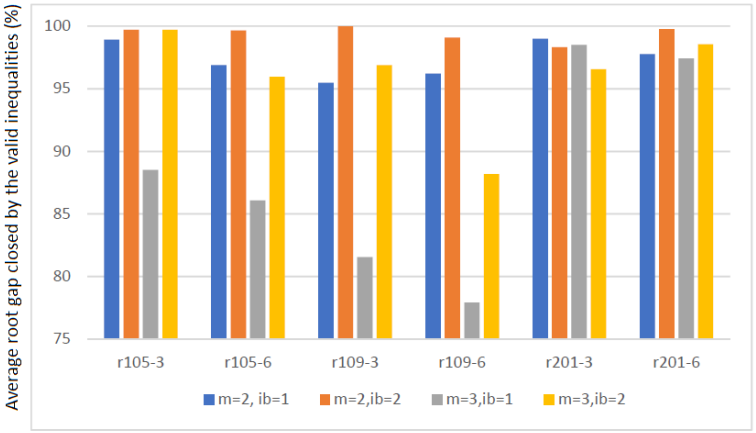

Figure 2: Average root gap closed by the valid inequalities in BP&C

Moreover, to achieve better upper bounds early on in the solution process, as soon as the column and cut generation terminates at the root node of the search tree, i.e., the master problem is solved to optimality at the root, we solve the integer version of the current RMP-R, which contains all the columns and cuts that have been generated so far, via branch-and-cut. We refer to this mixed integer program as *rootIP*, and store the capacity cuts (7) added while solving the rootIP, which are later included in the master problem associated with deeper nodes of the tree.

We set the time limit to 1800 seconds, and evaluate the efficacy of the above mentioned ideas on relatively small problem instances with 10 or 20 retailers and 3 or 6 periods. For the sake of brevity, we report only the results achieved with the enhancements that proved to be most useful, and hence, later employed when solving larger problem instances. In Tables 2–4, the results with our implementation of BP&C before and after adding the valid inequalities (14)–(15), and additionally, solving the rootIP are presented.

In Tables 2–4, BUB stands for best upper bound available when the BP&C algorithm terminates, which is equal to the optimal value of a problem instance if it can be solved within the time limit. Gap represents the percentage root gap which is computed as $(\bar{z} - \underline{z})/\bar{z}$, where $\bar{z}$ is the best upper bound achieved considering all three implementations of the algorithm and $\underline{z}$ is the best lower bound obtained at the root node (which is equal to the optimal value of the linear programming relaxation of the problem

instance upon solving the master problem at the root). Time represents the solution time (in seconds) if the instance is solved within the time limit, otherwise we use "TL" to indicate that the algorithm hits the time limit. rootIP represents the upper bound obtained by solving the rootIP, and rTime shows the total time (in seconds) elapsed, from the instant the execution starts, until the end of the rootIP solve. Since the rootIP can be solved rather quickly via CPLEX, we do not report rootIP solution times separately (we include that when reporting the solution time for each instance). Besides, we use N/A to indicate that an optimal solution is identified at the root node of the search tree without solving the rootIP. Node, pricing and cutting columns include the number of nodes explored in the solution tree, the number of pricing and cutting iterations performed, respectively.

The differences between the root gaps of BP&C with and without the valid inequalities show that the formulation can be significantly improved by adding the cuts (14) – (15). To demonstrate the improvements clearly, for each instance, we compute the root gap reduction accomplished by the valid inequalities through the ratio $(\underline{z}^c - \underline{z})/(\bar{z} - \underline{z})$, where $\underline{z}^c$ and $\underline{z}$ represent the lower bounds at the root node with and without the inequalities, and $\bar{z}$ is the best known upper bound for the problem instance under consideration. We group the instances based on the instance class they belong to, the number of periods, the capacity of each vehicle as well as the initial inventory setting, and report the average root gap reductions in Figure 2 for each instance group. According to Figure 2, the valid inequalities strengthen the lower bounds by more that 75% in all groups, and for 19 (out of 24) instance groups the average improvements are larger than 95%. The root gap reductions are smaller when the vehicle capacities and initial inventories are larger. The improvements slightly decrease as we increase the number of periods: for 3 period instances, the average root gap reductions are 97%, 93%, and 98% for r105, r109, and r201 instances, respectively, whereas for the 6 period instances, the corresponding improvements are 95%, 90%, and 98%. We observe the least improvements in R109 instances which are the hardest instances considered in our computational experiments. After the inclusion of valid inequalities, 8, 5 and 7 instances of R105, R109, and R201, respectively, are solved at the root node (without branching).

BP&C algorithm terminates by reaching the time limit in 16, 13 and 16 instances of R105, R109 and R201, respectively. After the enhancements, we can solve most of these instances to optimality in very short times: only 3 of R105 and 6 of R109 instances could not be solved within the time limit of 1800 seconds. But for those instances, the final gaps are reduced by 90% and 86% on the average. Besides, for the instances solved to optimality by BP&C, the average solution time is decreased by 95%. The improvement with respect to solution is depicted in Figure 3, where the average solution times of both implementations (initial and the best) for the given instance groups are plotted. Note that only the instances solved to optimality within the time limit are considered in this figure.

Tables 2–4 also reveal that the upper bounds identified by solving the rootIP are notably close to the optimal values of the instances solved within the time limit; the average gap is about 0.13% between the two values. On top of that, an optimal solution of the rootIP is found almost instantaneously in most cases, the average time is 1 second across all instances. Solving the rootIP yields remarkable performance improvements to our algorithm, which can also be observed by comparing the solution times with and without the rootIP throughout the Tables 2–4.

Table 2: Results for R105 instances

Instance	BUB	Branch-Price-and-Cut					+AK cuts					+ Root IP					
		Gap	Time	Node	Pricing	Cutting	Gap	Time	Node	Pricing	Cutting	Time	rootIP	rTime	Node	Pricing	Cutting
10-3-l-2-1	282.57	30.35	28.1	5,525	15,765	2,843	0	0.18	1	30	10	0.18	NA	NA	1	30	10
10-3-l-3-1	305.57	31	31	5,219	14,758	2,750	5.89	2.67	335	1,053	225	1.97	312.57	0.28	171	593	116
10-3-l-2-2	537.36	4.38	0.76	161	478	96	0	0.05	1	14	4	0.05	NA	NA	1	14	4
10-3-l-3-2	537.36	4.38	1.11	237	688	146	0	0.03	1	16	4	0.02	NA	NA	1	16	4
10-3-h-2-1	726.87	11.01	17	3,679	10,492	1,908	0	0.04	1	27	10	0.04	NA	NA	1	27	10
10-3-h-3-1	747.21	11.66	16.01	2,915	8,128	1,570	2.05	1.26	191	653	128	0.73	754.21	0.21	73	295	60
10-3-h-2-2	803.21	2.67	0.43	101	297	54	0	0.02	1	12	4	0.02	NA	NA	1	12	4
10-3-h-3-2	803.21	2.67	1.85	431	1,116	249	0	0.02	1	13	4	0.02	NA	NA	1	13	4
10-6-l-2-1	675.09	19.93	TL	70,004	196,260	35,190	1.03	13.54	703	2,247	420	7.65	675.09	0.66	311	1,048	199
10-6-l-3-1	723.23	17.67	TL	62,251	161,012	31,571	3.7	TL	44,219	127,586	22,600	TL	725.09	6.55	43,363	122,945	22,064
10-6-l-2-2	1128.62	4.5	70.5	8,323	21,799	4,190	0	0.07	1	29	12	0.07	NA	NA	1	29	12
10-6-l-3-2	1131	4.69	181	14,631	37,419	7,443	0.21	449	24,313	58,111	12,509	4.20	1131	0.45	251	702	158
10-6-h-2-1	1560.39	8.44	TL	70,629	200,931	35,530	0.29	16.17	827	2,663	503	1.36	1560.39	0.40	51	214	55
10-6-h-3-1	1610.75	7.92	TL	58,525	155,500	29,691	1.82	TL	38,479	108,517	19,691	1203	1610.75	2.99	28,623	80,575	14,633
10-6-h-2-2	1657.94	2.95	58	6,373	16,389	3,221	0	0.06	1	25	11	0.06	NA	NA	1	25	11
10-6-h-3-2	1661.62	3.17	152	11,803	30,468	6,029	0.19	58.40	2,915	8,364	1,574	1.84	1661.62	0.31	109	364	88
20-3-l-2-1	502.29	29.4	TL	35,680	246,945	17,890	0.88	1.06	17	197	15	1.38	504.37	0.61	17	197	15
20-3-l-3-1	504.37	28.5	TL	29,915	212,618	15,088	1.29	4.70	79	717	63	2.50	504.37	0.58	29	314	26
20-3-l-2-2	930.28	3.67	608	21,371	125,005	10,820	0.05	0.31	7	74	8	0.36	930.28	0.20	7	74	8
20-3-l-3-2	930.28	3.67	677	22,605	133,097	11,465	0.05	0.31	7	74	8	0.36	930.28	0.20	7	74	8
20-3-h-2-1	1387.89	9.61	TL	37,943	252,824	19,078	0.13	0.44	7	100	8	0.55	1392.55	0.30	7	100	8
20-3-h-3-1	1392.55	9.56	TL	29,560	213,181	14,871	0.47	3.04	55	504	40	2.63	1392.55	0.43	41	392	32
20-3-h-2-2	1560.74	1.75	130	4,809	27,217	2,437	0	0.15	3	39	6	0.20	1560.74	0.19	3	39	6
20-3-h-3-2	1560.74	1.75	137	4,921	27,852	2,498	0	0.15	3	39	6	0.21	1560.74	0.18	3	39	6
20-6-l-2-1	1169.49	20.13	TL	11,313	77,127	5,693	0.38	193	541	3,591	311	191	1169.57	3.72	497	3,233	281
20-6-l-3-1	1178.78	18.75	TL	7,545	54,558	3,957	1.06	TL	5,522	35,456	3,113	TL	1178.78	12.07	3,268	22,212	1,887
20-6-l-2-2	1921.95	3.53	TL	18,170	100,961	9,134	0.05	39.39	425	2,502	239	4.74	1922.07	1.36	35	251	33
20-6-l-3-2	1922.07	3.53	TL	15,439	90,427	7,803	0.06	70.45	555	3,570	350	6.18	1922.07	1.56	37	317	40
20-6-h-2-1	2941.91	7.93	TL	12,493	74,158	6,292	0.15	50.38	123	1,072	98	29.81	2943.91	2.41	69	672	55
20-6-h-3-1	2954.11	7.59	TL	7,718	53,334	4,019	0.48	TL	6,424	35,810	3,576	TL	2954.11	5.98	3,245	20,845	1,884
20-6-h-2-2	3184.28	2.16	TL	18,048	98,434	9,071	0	0.58	3	60	12	0.72	3184.28	0.72	3	60	12
20-6-h-3-2	3187.83	2.27	TL	15,433	84,777	7,815	0.1	624.15	5,595	33,438	2,926	30.07	3189.07	1.27	315	2,078	184

Table 3: Results for R109 instances

Instance	BUB	Branch-Price-and-Cut					+AK cuts					+ Root IP					
		Gap	Time	Node	Pricing	Cutting	Gap	Time	Node	Pricing	Cutting	Time	rootIP	rTime	Node	Pricing	Cutting
10-3-l-2-1	274.57	31.88	35.63	3,733	15,087	1,943	0.72	1.16	49	253	38	1.24	274.57	0.35	45	261	35
10-3-l-3-1	295.57	30.99	23.96	2,385	9,158	1,344	7	9.45	821	3,131	500	4.27	295.57	0.42	331	1,316	206
10-3-l-2-2	524.36	5.19	1.28	129	574	73	0	0.07	1	18	4	0.04	NA	NA	1	18	4
10-3-l-3-2	526.36	5.51	13.78	1,501	5,472	834	0.33	291.05	39,599	71,347	20,228	0.61	526.36	0.11	55	250	37
10-3-h-2-1	718.87	11.36	25.75	2,663	10,947	1,410	0.64	0.80	75	336	49	0.93	723.87	0.17	81	379	50
10-3-h-3-1	737.21	11.49	17.98	1,843	7,023	1,061	2.48	11.10	1,031	4,083	613	5.09	746.21	0.33	423	1,693	251
10-3-h-2-2	790.21	3.16	1.08	117	537	61	0	0.03	1	15	4	0.03	NA	NA	1	15	4
10-3-h-3-2	792.21	3.38	19.97	2,069	7,535	1,161	0.22	61.44	7,459	19,220	3,986	0.45	792.21	0.07	41	182	30
10-6-l-2-1	639.19	19.87	TL	43,846	189,484	22,128	0.4	207.98	4,595	18,751	2,438	64.48	639.19	1.16	1,271	5,809	682
10-6-l-3-1	697.14	16.21	TL	57,367	166,729	29,141	4.3	TL	45,009	131,541	22,950	TL	697.14	6.92	33,430	110,815	17,069
10-6-l-2-2	1088.62	4.35	TL	55,045	235,538	27,595	0	0.16	3	40	10	0.20	1088.62	0.20	3	40	10
10-6-l-3-2	1091.87	4.57	TL	57,692	186,240	29,336	0.29	TL	82,204	183,139	41,474	34.83	1091.87	0.43	1,005	4,652	559
10-6-h-2-1	1527.45	8.29	TL	43,332	184,183	21,902	0.39	256.51	5,255	21,695	2,786	31.82	1527.45	0.97	593	2,739	346
10-6-h-3-1	1582.62	7.08	TL	46,043	160,692	23,388	1.9	TL	34,665	113,401	17,956	TL	1582.62	5.22	30,024	104,837	15,329
10-6-h-2-2	1617.94	2.82	587.26	25,965	107,473	13,064	0	0.29	11	54	21	0.33	1617.94	0.19	11	54	21
10-6-h-3-2	1622.62	3.03	TL	45,756	183,180	23,233	0.25	TL	82,985	196,623	41,848	10.11	1622.62	0.32	339	1,382	216
20-3-l-2-1	442.37	27.76	TL	15,850	184,619	7,974	1.36	1.68	3	120	7	1.80	442.37	0.79	3	120	7
20-3-l-3-1	455.37	27.78	TL	9,265	127,126	4,873	3.99	263.46	1,167	12,647	715	213.88	462.37	5.32	849	9,558	509
20-3-l-2-2	851.28	2.31	54.54	661	6,757	347	0	0.52	3	64	5	0.62	851.28	0.46	3	64	5
20-3-l-3-2	851.28	2.31	234.57	2,199	23,419	1,131	0	0.55	3	66	7	0.65	851.28	0.51	3	65	6
20-3-h-2-1	1330.55	8.42	TL	15,813	193,229	7,942	0.45	1.57	5	131	7	1.67	1330.55	0.81	5	131	7
20-3-h-3-1	1343.55	8.91	TL	7,338	107,185	4,013	1.35	395.08	2,673	28,895	1,469	144.46	1344.89	1.90	717	8,458	415
20-3-h-2-2	1482.37	0.86	51.88	701	6,417	382	0	0.27	1	39	2	0.27	NA	NA	1	39	2
20-3-h-3-2	1482.37	0.86	54.13	657	6,503	346	0	0.27	1	39	2	0.28	NA	NA	1	39	2
20-6-l-2-1	1030.82	19.44	TL	4,536	49,525	2,377	0.7	1346.58	2,089	20,674	1,147	254.04	1031.82	7.51	417	3,630	261
20-6-l-3-1	1068.95	18.44	TL	4,589	41,482	2,661	3.24	TL	3,918	37,699	2,350	TL	1068.95	69.19	1,942	16,673	1,184
20-6-l-2-2	1775.95	3.21	TL	5,339	58,671	2,693	0	1.29	1	66	14	1.35	NA	NA	1	66	14
20-6-l-3-2	1786.42	3.78	TL	4,836	52,872	2,607	0.59	TL	4,261	45,423	2,240	TL	1788.65	4.24	4,300	43,862	2,280
20-6-h-2-1	2808.18	7.31	TL	3,429	46,348	1,774	0.36	TL	2,878	33,468	1,585	452.66	2809.18	7.09	735	6,139	431
20-6-h-3-1	2844.96	7.15	TL	5,036	53,832	2,761	1.25	TL	3,439	28,822	2,117	TL	2844.96	76.12	2,094	18,191	1,218
20-6-h-2-2	3039.83	2.09	TL	4,943	61,298	2,522	0.08	1270.08	3,605	32,501	1,903	379.88	3039.83	1.77	1,143	10,472	596
20-6-h-3-2	3053.23	2.52	TL	4,887	59,196	2,562	0.43	TL	4,669	43,569	2,523	TL	3053.23	2.57	4,255	46,418	2,196

Table 4: Results for R201 instances

Instance	BUB	Branch-Price-and-Cut					+AK cuts					+ Root IP					
		Gap	Time	Node	Pricing	Cutting	Gap	Time	Node	Pricing	Cutting	Time	rootIP	rTime	Node	Pricing	Cutting
10-3-l-2-1	297.57	34.03	78.76	12,673	43,657	6,405	0.67	0.35	5	44	10	1.28	297.57	0.26	5	44	8
10-3-l-3-1	298.57	5.14	90.69	12,319	43,746	55	0	1.32	163	530	4	0.27	299.57	0.14	11	77	4
10-3-l-2-2	540.36	34.07	0.75	89	419	6,235	1	0.09	1	28	108	0.07	NA	NA	1	28	16
10-3-l-3-2	541.36	5.31	1.13	159	640	93	0.18	0.65	81	302	59	0.40	541.36	0.12	25	140	20
10-3-h-2-1	741.87	12.91	36.19	5,977	20,835	3,052	0.27	0.06	3	36	9	0.09	741.87	0.08	3	36	9
10-3-h-3-1	742.87	3.26	75.98	11,297	40,067	40	0	0.32	37	175	4	0.17	742.87	0.15	9	68	4
10-3-h-2-2	806.21	12.92	0.46	65	274	5,729	0.4	0.04	1	28	38	0.04	NA	NA	1	28	16
10-3-h-3-2	807.21	3.38	1.29	185	715	114	0.12	0.99	121	497	85	0.31	807.21	0.11	33	164	24
10-6-l-2-1	693.09	22.08	TL	65,493	224,668	32,801	0.72	12.24	517	1,933	312	8.75	694.09	0.70	235	1,069	142
10-6-l-3-1	694.09	4.86	TL	63,164	203,678	16,163	0	24.22	731	2,995	13	10.58	694.09	0.27	305	1,402	13
10-6-l-2-2	1133.75	21.9	634.71	32,217	121,235	31,746	0.84	0.14	1	47	478	0.15	NA	NA	1	47	196
10-6-l-3-2	1134.88	4.96	878.05	38,017	140,336	19,077	0.1	9.18	441	1,686	256	5.49	1134.88	0.49	227	944	137
10-6-h-2-1	1577.94	9.45	TL	65,198	236,962	32,647	0.31	4.13	127	599	98	2.45	1577.94	0.49	69	305	51
10-6-h-3-1	1578.55	3.26	TL	58,549	201,738	12,348	0	23.20	871	3,168	24	4.91	1579.63	0.68	121	582	13
10-6-h-2-2	1662.94	9.33	420.72	24,579	93,941	29,442	0.34	0.60	27	135	553	0.28	1662.94	0.22	7	53	100
10-6-h-3-2	1664.94	3.37	657.85	29,683	114,457	15,004	0.09	34.96	1,783	6,201	1,003	1.05	1664.94	0.41	35	187	34
20-3-l-2-1	483.37	27.14	TL	22,027	216,142	11,030	0	0.45	1	86	4	0.45	NA	NA	1	86	4
20-3-l-3-1	483.37	2.33	TL	17,590	213,916	2,585	0.13	0.45	1	83	7	0.43	NA	NA	1	83	7
20-3-l-2-2	911.74	27.14	287.68	5,073	41,010	8,811	0	0.57	7	94	5	0.64	911.74	0.30	7	94	5
20-3-l-3-2	911.74	2.33	279.48	4,755	39,586	2,422	0.13	0.49	7	78	7	0.54	911.74	0.28	7	78	7
20-3-h-2-1	1371.55	8.76	TL	23,421	247,274	11,721	0	0.37	1	73	4	0.37	NA	NA	1	73	4
20-3-h-3-1	1371.55	1.11	TL	21,367	229,590	476	0.02	0.40	1	77	11	0.43	NA	NA	1	77	7
20-3-h-2-2	1545.19	8.76	50.03	893	7,346	10,697	0	0.90	13	135	6	0.43	1545.19	0.32	5	57	6
20-3-h-3-2	1545.19	1.11	51.98	889	7,717	472	0.02	1.26	21	182	17	0.51	1545.19	0.33	7	60	9
20-6-l-2-1	1142.65	19.6	TL	4,547	64,630	2,300	0.27	104.43	233	2,310	132	101.38	1144.83	3.42	207	2,027	119
20-6-l-3-1	1142.65	2.96	TL	4,453	56,259	4,940	0	148.13	275	3,016	31	134.25	1145.49	3.88	281	2,749	12
20-6-l-2-2	1890.95	19.59	TL	9,832	78,531	2,265	0.27	8.64	29	312	176	1.98	1891.53	1.42	5	87	173
20-6-l-3-2	1890.95	2.96	TL	8,793	77,765	4,447	0	3.12	13	149	18	1.94	1891.53	1.31	5	89	12
20-6-h-2-1	2915.81	7.68	TL	5,109	59,806	2,581	0.08	23.97	51	515	43	27.44	2915.81	2.33	39	392	34
20-6-h-3-1	2916.81	2.02	TL	5,052	58,087	5,173	0.02	105.37	181	2,043	22	84.71	2917.22	3.91	153	1,775	19
20-6-h-2-2	3159.48	7.71	TL	10,269	78,471	2,563	0.12	2.56	13	104	116	3.53	3159.48	1.09	11	92	101
20-6-h-3-2	3159.48	2.02	TL	9,746	79,403	4,894	0.02	1.96	11	89	20	2.21	3159.69	1.05	11	89	20

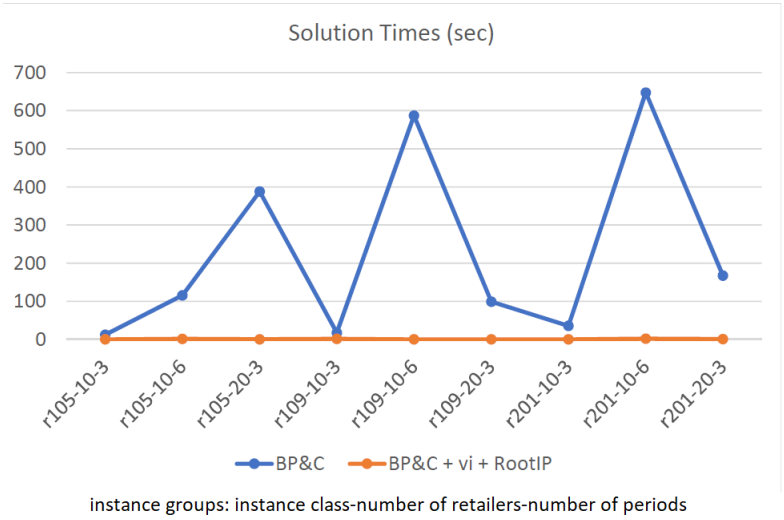

Figure 3: Average solution times of BP&C with and without the improvements

As a result of using the valid inequalities (14)–(15) and seeking stronger upper bounds via solving the rootIP, both the best lower bound and the best upper bound at the root node are significantly improved. This enables finding optimal solutions in smaller computation times by exploring fewer nodes and performing fewer column generation and cutting plane iterations, as can be observed from Tables 2–4: the numbers of nodes explored, pricing and cutting iterations are all reduced by 91-92% on the average. Note that the smallest improvements are again observed in the R109 instances, which is consistent with our observation in Section 5.1 on their difficulty level.

In the next section, we use a set of larger problem instances to test the fastest performing BP&C implementation based on our observations regarding the small instances.

5.3 Results for Larger Problem Instances

In this part, we present the results of our second experiment, conducted on a set of medium and large problem instances with 30, 50, or 100 retailers, and 3 or 6 periods, consistent with the existing literature on IRP such as Avella et al. (2017), Desaulniers et al. (2016), Archetti et al. (2014). Different from before, here we set the time limit to two hours for BP&C, and consider the instances with $m = 4$, i.e. vehicles have smaller capacities, when $|V| \in \{50, 100\}$. Based on our first experiment, we allow CPLEX to run at most 1800 seconds to solve the rootIP with the hope of attaining better upper bounds early on in the search.

The results of this experiment are reported in Tables 5–7. In addition to rTime, we also provide the solution time of rootIP itself when solved to optimality within the time limit under the column "Time" under rootIP, otherwise we use "TL" to indicate that CPLEX hits the time limit of 1800 seconds allocated to solve the rootIP.

We are able to solve all 30 retailer-6 period, 50 retailer-3 period instances of R105 and R201 to

optimality, the average solution times are 1186s and 613s for R105, and 774s and 97s for R201, respectively. We observe that it takes longer to solve the instances with low inventory holding costs and smaller stock capacities, or high inventory holding costs and larger stock capacities. Note that all R201-50-3 instances with $ib = 1$, i.e. high inventory upper bounds, are solved at the root node without the need to solve the rootIP.

Among the large R105 and R201 instances, six of them (three in each group) involving 50 retailers and 6 periods can be solved optimally, with average solution times of 1258s and 207s, respectively. The average final gap is 2.22% across the other instances in the two groups (0.84% for R105, and 3.61% for R201 instances).

Out of all R109 instances (medium and large), only six of them can be solved to optimality within the time limit. However, for the remaining 38 instances for which the algorithm terminated upon reaching the time limit, we observe that the final gap is less than 2% in most cases, and 2.17% on the average. The largest percentage gap values are observed in the 100 retailer instances with low inventory holding costs, which can be regarded as quite large instances considering the state of the art in the IRP literature. The average final gaps for the instances that cannot be solved to optimality are given in Figure 4.

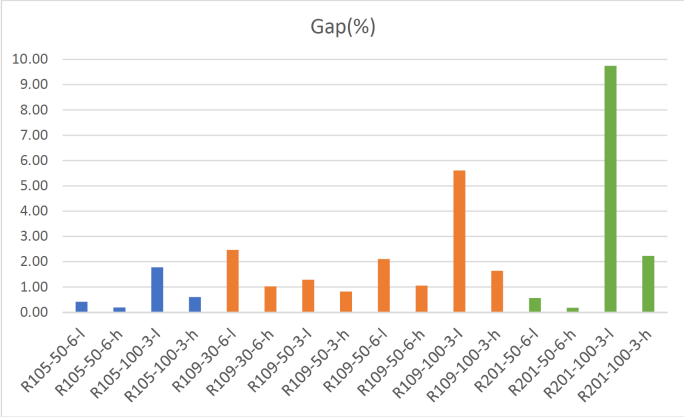

Figure 4: Average final gaps for the instances terminated due to the time limit

One of the key factors in the success of our algorithm is the high quality of the solutions obtained at the root node by solving the rootIP. According to our computational results, the average gap between the optimal values and the upper bounds produced by the rootIP is around 0.18% considering the instances for which an optimal solution is found and the rootIP is solved during the search. A similar result applies also to the instances that could not be solved to optimality within two hours, in 66% of all cases, the best upper bound reported is found by solving the rootIP. For these instances, the average time spent after the root node of the solution tree is around 4500 seconds, and the root gap is 2.87% on the average. The main takeaway here is that solutions of acceptable quality can in fact be obtained reasonably quickly using our BP&C algorithm.

We observe that the difficulty of a problem instance depends on its parameter configurations. Our analyses show that the problem instances in R109 class are harder compared to the other two classes.

We believe that this is due to higher time window density of this instance group which is discussed in Section 5.1. As one might expect, the problem difficulty increases with the number of retailers and the number of periods. We observe that the performance of our algorithm is affected more by the number of retailers than it is by the number of periods. The inventory upper bound and initial inventory settings of the retailers also have an impact on the problem difficulty. As discussed in Section 5.1, in our problem setting, an instance with $ib = 1$ can be reduced to an $n - 1$ period instance, where n represents the number of periods in the original problem. This observation implies that instances with $ib = 2$ –the more general setting we introduced– may be more difficult than the instances with $ib = 1$. On the other hand, when $ib = 2$, if there exists a retailer i with $u_i = d_i$, then d_i should be sent to i in each period, i.e. $q_{it} = d_i$ for all t. In other words, the lot sizing problem for i is trivial, but still, a route that visits i should be determined for each period. According to the solution times and the root gaps reported in Tables 2–7, despite a few exceptional cases, instances with $ib = 2$ are harder than the ones with $ib = 1$ in general.

Another important parameter is the level of inventory holding costs. We observe that for the instances with lower inventory holding costs, the solution times (when the problem is solved to optimality) or the final gaps (when the algorithm terminates due to the time limit) are larger in general. When the inventory holding costs are low, the trade-off between inventory and transportation costs may be amplified, which in turn may cause difficulties in proving optimality. In our solutions, when the inventory costs are high, the delivery schedule involves a greater number of routes in total, which is not surprising since delivering in small quantities frequently to a retailer may be cheaper than delivering large quantities less often and increasing the inventory levels at the retailer. Finally, we note that the optimal values do not change with the vehicle capacities, which implies that the time windows of the retailers are typically more restrictive than the vehicle capacities for our larger test instances.

6 Conclusion

We have introduced a new formulation for the IRPTW that includes both arc and route based variables, strengthened our formulation using valid inequalities derived for the lot sizing problem, and developed an efficient branch-price-and-cut algorithm to solve it. To the best of our knowledge, ours is the first exact approach specifically designed for solving the IRPTW. The results of our computational experiments clearly demonstrate the efficacy of our solution approach, which is able to obtain optimal or fairly good solutions to medium and large problem instances in reasonable computation times.

We observe that the existing lot sizing inequalities are very effective in improving the formulation, thereby, boosting the algorithmic performance. Hence, new valid inequalities for the IRPTW may be worth investigating in the future. Another direction for further research may be to work on adapting the proposed BP&C algorithm to solve the production routing problem with time windows, a generalization of the IRPTW, where production quantities at the supplier are also decision variables.

Table 5: Results for larger R105 instances

Instance	Gap	Time	BUB	rootIP		rTime	Node	Pricing	Cutting
				BUB	Time				(cap; vi)
30-6-l-2-1	0.57	1527.03	1605.45	1605.45	0.68	5.97	1,833	14,574	933 (0; 245)
30-6-l-3-1	0.57	1221.26	1605.45	1605.45	0.57	8.28	1,367	9,586	799 (110; 238)
30-6-l-2-2	0.39	4599.88	2330.47	2341.47	7.57	12.76	8,979	61,570	5019 (0; 163)
30-6-l-3-2	0.39	1596.56	2330.47	2331.05	1.9	6.98	3,079	20,774	1639 (20; 163)
30-6-h-2-1	0.14	335.18	4363.24	4363.24	0.45	4.75	447	3,475	239 (3; 206)
30-6-h-3-1	0.14	116.56	4363.24	4363.24	0.99	7.25	165	1,186	132 (45; 212)
30-6-h-2-2	0.16	49.23	4447.41	4448.41	1.35	5.79	73	799	50 (0; 139)
30-6-h-3-2	0.16	41.20	4447.41	4448.41	1.33	5.78	63	684	44 (1; 139)
50-3-l-2-1	0.97	168.36	1065.99	1066.99	1.26	14.04	159	2,423	91 (0;97)
50-3-l-3-1	0.97	270.53	1065.99	1065.99	0.77	12.14	245	3,950	131 (0;97)
50-3-l-4-1	0.97	230.58	1065.99	1067.99	0.98	14.22	243	3,281	144 (11;99)
50-3-l-2-2	0.72	1985.00	1878.47	1878.86	0.48	8.62	2,255	26,469	1,145 (0;50)
50-3-l-3-2	0.72	1458.82	1878.47	1878.75	0.46	9.00	1,555	19,931	799 (0;52)
50-3-l-4-2	0.72	2135.50	1878.47	1888.93	0.62	8.30	2,385	29,936	1,241 (0;51)
50-3-h-2-1	0.30	317.00	3421.65	3422.65	1.33	12.60	325	4,680	176 (0;78)
50-3-h-3-1	0.30	287.27	3421.65	3421.65	0.92	10.62	271	3,966	148 (2;83)
50-3-h-4-1	0.30	330.12	3421.65	3423.65	1.14	26.31	351	4,228	203 (17;78)
50-3-h-2-2	0.10	46.74	3615.14	3616.14	0.32	7.77	41	692	25 (0;39)
50-3-h-3-2	0.10	71.94	3615.14	3617.14	0.48	8.45	65	908	42 (0;40)
50-3-h-4-2	0.10	55.84	3615.14	3616.14	0.31	8.20	53	756	32 (0;39)
50-6-l-2-1	0.58	TL	2614.15	2617.52	97.1	150.26	3,039	20,864	1,546 (0;366)
50-6-l-3-1	0.48	TL	2611.54	2611.54	40.86	82.46	3,385	23,219	1,715 (2;381)
50-6-l-4-1	0.68	TL	2616.65	2616.65	71.23	130.26	2,317	18,744	1,233 (76;381)
50-6-l-2-2	0.30	TL	3884.94	3884.94	8.18	33.60	4,478	30,772	2,277 (0;236)
50-6-l-3-2	0.23	TL	3881.96	3881.96	15.78	41.83	3,885	26,050	1,959 (0;245)
50-6-l-4-2	0.20	TL	3880.96	3880.96	14.81	43.21	3,939	25,765	1,997 (10;244)
50-6-h-2-1	0.26	TL	7316.38	7316.38	36.02	75.73	3,061	22,008	1,555 (0;356)
50-6-h-3-1	0.29	TL	7318.38	7318.38	46.52	88.34	3,089	24,443	1,573 (7;346)
50-6-h-4-1	0.28	TL	7317.53	7317.53	65.51	109.32	2,180	17,776	1,217 (122;355)
50-6-h-2-2	0.10	1425.07	7364.06	7366.06	7.96	36.85	719	5,311	391 (0;185)
50-6-h-3-2	0.10	1240.09	7364.06	7366.06	5.14	37.62	657	4,375	353 (0;196)
50-6-h-4-2	0.10	1109.44	7364.06	7366.06	2.12	29.89	607	4,504	336 (0;199)
100-3-l-2-1	1.37	TL	1681.47	1727.38	321.54	1571.84	572	13,051	311 (0;181)
100-3-l-3-1	1.37	TL	1681.47	1727.38	334	1575.54	570	12,948	311 (0;181)
100-3-l-4-1	1.37	TL	1681.47	1727.38	318.43	1584.91	575	13,134	312 (0;181)
100-3-l-2-2	2.18	TL	2822.39	2822.39	93.94	569.55	429	9,009	245 (0;119)
100-3-l-3-2	2.18	TL	2822.39	2822.39	95.86	590.93	430	9,063	245 (0;119)
100-3-l-4-2	2.18	TL	2822.39	2822.39	92.47	559.93	439	9,215	252 (0;119)
100-3-h-2-1	0.37	TL	6392.08	6430.08	356.91	1131.85	671	15,438	378 (0;153)
100-3-h-3-1	0.37	TL	6392.08	6430.08	357.03	1116.09	650	15,027	368 (0;153)
100-3-h-4-1	0.37	TL	6392.08	6430.08	357.64	1107.30	643	14,860	364 (0;153)
100-3-h-2-2	0.83	TL	6097.67	6097.67	122.44	584.75	413	9,066	234 (0;102)
100-3-h-3-2	0.83	TL	6097.67	6097.67	122.08	582.41	386	8,537	218 (0;102)
100-3-h-4-2	0.83	TL	6097.67	6097.67	121.85	578.57	397	8,777	225 (0;102)

Table 6: Results for larger R109 instances

Instance	Gap	Time	BUB	rootIP		rTime	Node	Pricing	Cutting
				BUB	Time				(cap; vi)
30-6-l-2-1	1.83	TL	1463.39	1469.39	25.57	39.238	5,230	49,665	2,652 (10;222)
30-6-l-3-1	1.78	TL	1462.61	1466.21	24.2	41.043	3,360	30,750	1,872 (226;224)
30-6-l-2-2	3.13	TL	2104.44	2104.44	7.39	16.253	7,225	77,631	3,639 (0;161)
30-6-l-3-2	3.13	TL	2104.44	2104.44	3.54	12.281	5,479	71,776	2,773 (7;161)
30-6-h-2-1	0.43	2848.52	4218.96	4223.8	6.19	17.14	2,063	16,044	1,047 (3;211)
30-6-h-3-1	0.64	TL	4227.8	4227.97	12.52	27.933	3,243	33,824	1,823 (214;206)
30-6-h-2-2	1.05	TL	4208.41	4221.48	4.6	13.249	7,282	64,110	3,670 (0;145)
30-6-h-3-2	1.37	TL	4221.96	4221.96	5.28	13.941	801	22,443	13,635 (20;145)
50-3-l-2-1	1.19	3796.01	927.99	932.99	6.63	88.248	1,259	21,775	637 (0;79)
50-3-l-3-1	1.19	3852.46	927.99	932.99	6.24	87.101	1,203	21,213	613 (3;79)
50-3-l-4-1	1.19	6768.9	927.99	946.99	6.61	83.374	1,795	37,670	946 (40;85)
50-3-l-2-2	1.29	TL	1644.15	1664.93	6.79	42.25	1,914	25,176	1,002 (0;70)
50-3-l-3-2	1.29	TL	1644.15	1664.93	6.63	42.033	1,867	24,611	977 (0;70)
50-3-l-4-2	1.29	TL	1644.15	1664.93	6.64	41.506	1,975	25,808	1031 (0;70)
50-3-h-2-1	0.33	1573.37	3283.65	3288.65	4.63	63.051	591	8,867	303 (0;71)
50-3-h-3-1	0.33	1523.61	3283.65	3288.65	4.61	62.862	591	8,867	303 (0;71)
50-3-h-4-1	0.91	TL	3302.65	3308.01	15.79	261.862	1787	42,088	996 (90;79)
50-3-h-2-2	0.82	TL	3392.04	3405.48	1.55	31.494	2,238	29,568	1,164 (0;61)
50-3-h-3-2	0.82	TL	3392.04	3405.48	1.56	32.154	2,206	29,162	1,150 (0;61)
50-3-h-4-2	0.82	TL	3392.04	3405.48	1.54	32.87	2,213	29,255	1,153 (0;61)
50-6-l-2-1	1.82	TL	2259.34	2259.34	413.17	626.719	1,042	12,073	548 (0;353)
50-6-l-3-1	1.82	TL	2259.34	2259.34	253.84	437.308	1,195	12,668	625 (8;356)
50-6-l-4-1	1.93	TL	2262.02	2262.02	640.36	992.558	707	7,961	445 (157;361)
50-6-l-2-2	2.16	TL	3397.62	3397.62	27.49	116.36	1,346	14,550	710 (0;219)
50-6-l-3-2	2.28	TL	3401.7	3401.7	65.74	155.948	1,314	14,789	696 (3;219)
50-6-l-4-2	2.63	TL	3413.88	3413.88	267.34	360.483	1,288	13,479	685 (20;219)
50-6-h-2-1	0.67	TL	6971.14	6971.14	457.73	622.649	1,163	11,685	612 (0;340)
50-6-h-3-1	0.55	TL	6962.44	6965.04	295.49	524.236	950	10,161	516 (26;331)
50-6-h-4-1	0.88	TL	6985.61	6985.61	TL	2144.529	437	5,911	355 (230;336)
50-6-h-2-2	1.28	TL	6904.37	6904.37	30.73	138.717	1,232	15,479	662 (0;195)
50-6-h-3-2	1.28	TL	6904.37	6904.37	56.16	165.955	1,109	12,325	599 (2;195)
50-6-h-4-2	1.49	TL	6918.48	6918.48	143.16	252.926	1,023	12,558	563 (27;195)
100-3-l-2-1	4.72	TL	1497.47	1497.47	586.39	6196.169	10	1,843	13 (0;156)
100-3-l-3-1	4.72	TL	1497.47	1497.47	659.11	6218.141	9	1,778	12 (0;156)
100-3-l-4-1	4.72	TL	1497.47	1497.47	619.66	6254.332	9	1,794	12 (0;156)
100-3-l-2-2	6.5	TL	2465.97	2465.97	TL	4202.564	32	1,611	37 (0;140)
100-3-l-3-2	6.5	TL	2465.97	2465.97	TL	4325.815	31	1,608	36 (0;140)
100-3-l-4-2	6.5	TL	2465.97	2465.97	TL	4286.594	31	1,606	36 (0;140)
100-3-h-2-1	1.14	TL	6208.08	6208.08	257.18	5258.251	19	1,530	23 (0;126)
100-3-h-3-1	1.14	TL	6208.08	6208.08	259.88	5644.868	20	1,532	24 (0;126)
100-3-h-4-1	1.14	TL	6208.08	6208.08	279.58	5523.729	21	1,540	25 (0;126)
100-3-h-2-2	2.14	TL	5720.27	5720.27	1082.23	4072.555	32	1,648	37 (0;107)
100-3-h-3-2	2.14	TL	5720.27	5720.27	1006.87	4158.464	33	1,660	38 (0;107)
100-3-h-4-2	2.14	TL	5720.27	5720.27	1003.17	4049.972	32	1,652	37 (0;107)

Table 7: Results for larger R201 instances

Instance	Gap	Time	BUB	rootIP		rTime	Node	Pricing	Cutting
				BUB	Time				(cap; vi)
30-6-l-2-1	0.55	1625.15	1577.79	1584.45	3.97	14.27	1,307	12,810	681 (0; 227)
30-6-l-3-1	0.55	3132.50	1577.79	1581.45	3.71	13.19	2,603	25,843	1,319 (5; 226)
30-6-l-2-2	0.11	77.59	2296.47	2298.14	1.36	9.54	101	710	71 (0; 164)
30-6-l-3-2	0.11	809.23	2296.47	2306.39	3.96	11.83	1,021	8,820	546 (0; 157)
30-6-h-2-1	0.23	29.86	4341.48	4345.68	2.49	10.82	17	245	19 (0; 211)
30-6-h-3-1	0.23	502.81	4341.48	4345.68	2.01	11.48	471	4,633	257 (5; 215)
30-6-h-2-2	0.00	8.49	4410.99	NA	NA	NA	1	119	9 (0; 134)
30-6-h-3-2	0.00	7.57	4410.99	NA	NA	NA	1	122	9 (0; 133)
50-3-l-2-1	0.00	75.72	1039.99	NA	NA	NA	1	558	4 (0;93)
50-3-l-3-1	0.00	74.38	1039.99	NA	NA	NA	1	558	4 (0;93)
50-3-l-4-1	0.00	73.21	1039.99	NA	NA	NA	1	558	4 (0;93)
50-3-l-2-2	0.48	125.24	1857.23	1859.89	1.29	38.75	37	709	28 (0;55)
50-3-l-3-2	0.48	109.07	1857.23	1859.89	1.33	37.93	37	709	28 (0;55)
50-3-l-4-2	0.48	107.86	1857.23	1859.89	1.32	38.15	37	709	28 (0;55)
50-3-h-2-1	0.00	70.11	3395.65	NA	NA	NA	1	485	4 (0;72)
50-3-h-3-1	0.00	82.90	3395.65	NA	NA	NA	1	485	4 (0;72)
50-3-h-4-1	0.00	69.85	3395.65	NA	NA	NA	1	485	4 (0;72)
50-3-h-2-2	0.15	123.55	3594.45	3600.45	1.12	33.03	41	778	27 (0;43)
50-3-h-3-2	0.15	120.97	3594.45	3600.45	1.13	33.21	41	778	27 (0;43)
50-3-h-4-2	0.15	124.78	3594.45	3600.45	1.09	32.91	41	778	27 (0;43)
50-6-l-2-1	0.77	TL	2572.14	2586.17	170.56	359.27	1,047	12,964	592 (0;382)
50-6-l-3-1	0.77	TL	2572.14	2586.17	173.37	363.13	1,046	12,961	591 (0;382)
50-6-l-4-1	0.68	TL	2569.62	2586.17	171.32	363.00	1,106	13,060	644 (3;382)
50-6-l-2-2	0.38	TL	3845.46	3848.85	26.37	119.76	1,998	20,908	1,050 (0;211)
50-6-l-3-2	0.38	TL	3845.46	3848.85	26.24	121.73	2,005	20,975	1,053 (0;211)
50-6-l-4-2	0.38	TL	3845.46	3848.85	25.85	119.01	1,965	20,558	1,031 (0;211)
50-6-h-2-1	0.17	TL	7268.4	7276.72	24.76	247.04	1,273	12,411	657 (0;337)
50-6-h-3-1	0.17	TL	7268.4	7276.72	24.91	251.78	1,266	12,326	654 (0;337)
50-6-h-4-1	0.20	TL	7270.58	7277.58	31.66	254.71	1,003	12,827	607 (34;337)
50-6-h-2-2	0.02	211.91	7317.22	7317.22	2.7	87.64	27	574	29 (0;209)
50-6-h-3-2	0.02	207.76	7317.22	7317.22	2.68	90.25	27	574	29 (0;209)
50-6-h-4-2	0.02	200.01	7317.22	7317.22	2.72	87.78	27	574	29 (0;209)
100-3-l-2-1	8.47	TL	1558.81	1558.81	TL	7805.11	1	1,797	4 (0;156)
100-3-l-3-1	8.40	TL	1557.7	1557.7	TL	7704.58	1	1,797	4 (0;156)
100-3-l-4-1	8.47	TL	1558.81	1558.81	TL	7741.34	1	1,797	4 (0;156)
100-3-l-2-2	9.11	TL	2536.88	2536.88	TL	4752.67	25	1,641	33 (0;141)
100-3-l-3-2	12.76	TL	2643.01	2643.01	TL	4743.19	24	1,639	32 (0;141)
100-3-l-4-2	11.25	TL	2598.12	2598.12	TL	4637.30	28	1,669	36 (0;141)
100-3-h-2-1	2.03	TL	6264.48	6264.48	TL	6395.58	5	1,581	9 (0;125)
100-3-h-3-1	2.03	TL	6264.48	6264.48	TL	6260.01	6	1,583	10 (0;125)
100-3-h-4-1	2.03	TL	6264.48	6264.48	TL	6080.79	6	1,588	10 (0;125)
100-3-h-2-2	2.44	TL	5738.16	5738.16	TL	4992.40	23	1,704	27 (0;106)
100-3-h-3-2	2.44	TL	5738.16	5738.16	TL	4966.08	21	1,678	25 (0;106)
100-3-h-4-2	2.41	TL	5736.16	5736.16	TL	5029.22	21	1,681	25 (0;106)

References

Adulyasak, Y., Cordeau, J.-F. and Jans, R. (2013), 'Formulations and branch-and-cut algorithms for multivehicle production and inventory routing problems', *INFORMS Journal on Computing* **26**(1), 103–120.

Andersson, H., Christiansen, M. and Fagerholt, K. (2010), Transportation planning and inventory management in the lng supply chain, *in* 'Energy, Natural Resources and Environmental Economics', Springer, pp. 427–439.

Archetti, C., Bertazzi, L., Laporte, G. and Speranza, M. G. (2007), 'A branch-and-cut algorithm for a vendor-managed inventory-routing problem', *Transportation Science* **41**(3), 382–391.

Archetti, C., Bianchessi, N., Irnich, S. and Speranza, M. G. (2014), 'Formulations for an inventory routing problem', *International Transactions in Operational Research* **21**(3), 353–374.

Atamtürk, A. and Küçükyavuz, S. (2005), 'Lot sizing with inventory bounds and fixed costs: Polyhedral study and computation', *Operations Research* **53**(4), 711–730.

Avella, P., Boccia, M. and Wolsey, L. A. (2015), 'Single-item reformulations for a vendor managed inventory routing problem: Computational experience with benchmark instances', *Networks* **65**(2), 129–138.

Avella, P., Boccia, M. and Wolsey, L. A. (2017), 'Single-period cutting planes for inventory routing problems', *Transportation Science* **52**(3), 497–508.

Barany, I., Van Roy, T. J. and Wolsey, L. A. (1984), 'Strong formulations for multi-item capacitated lot sizing', *Management Science* **30**(10), 1255–1261.

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

Christiansen, M. (1999), 'Decomposition of a combined inventory and time constrained ship routing problem', *Transportation Science* **33**(1), 3–16.

Coelho, L. C. and Laporte, G. (2013), 'The exact solution of several classes of inventory-routing problems', *Computers & Operations Research* **40**(2), 558–565.

Coelho, L. C. and Laporte, G. (2014), 'Improved solutions for inventory-routing problems through valid inequalities and input ordering', *International Journal of Production Economics* **155**, 391–397.

Dauzère-Pérès, S., Nordli, A., Olstad, A., Haugen, K., Koester, U., Per Olav, M., Teistklub, G. and Reistad, A. (2007), 'Omya Hustadmarmor optimizes its supply chain for delivering calcium carbonate slurry to European paper manufacturers', *Interfaces* **37**(1), 39–51.

Desaulniers, G. (2010), 'Branch-and-price-and-cut for the split-delivery vehicle routing problem with time windows', *Operations Research* **58**(1), 179–192.

Desaulniers, G., Rakke, J. G. and Coelho, L. C. (2016), 'A branch-price-and-cut algorithm for the inventory-routing problem', *Transportation Science* **50**(3), 1060–1076.

Eppen, G. and Martin, R. (1987), 'Solving multi-item lot-sizing problems using variable definition', *Operations Research* **35**, 832–848.

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

Kinable, J., Hadka, D. and Lelieveld, F. (2016), 'jORLib–Java Operations Research Library', http://www.coin-or.org/projects/jORLib.xml.

Klempous, R., Nikodem, J., Jacak, W. and Chaczko, Z., eds (2014), *Advanced Methods and Applications in Computational Intelligence*, Vol. 6 of *Topics in Intelligent Engineering and Informatics*, Springer. **URL:** *http://link.springer.com/chapter/10.1007/978-3-319-01436-4_10*

Liu, S.-C. and Lee, W.-T. (2011), 'A heuristic method for the inventory routing problem with time windows', *Expert Systems with Applications* **38**(10), 13223–13231.

Mahmutoğulları, Ö. and Yaman, H. (2019), A branch and cut algorithm for the inventory routing problem, Technical report, Department of Industrial Engineering, Bilkent University, Ankara, Turkey.

Mercer, A. and Tao, X. (1996), 'Alternative inventory and distribution policies of a food manufacturer', *Journal of the Operational Research Society* **47**(6), 755–765.

Michail, D., Kinable, J., Naveh, B. and Sichi, J. V. (2020), 'JGraphT - A Java library for graph data structures and algorithms', *ACM Transactions on Mathematical Software* **46**(2), 1–29.

Ozbaygin, G., Karasan, O. E., Savelsbergh, M. and Yaman, H. (2017), 'A branch-and-price algorithm for the vehicle routing problem with roaming delivery locations', *Transportation Research Part B: Methodological* **100**, 115–137.

Pochet, Y. and Wolsey, L. A. (1994), 'Polyhedra for lot-sizing with wagner—whitin costs', *Mathematical Programming* **67**(1-3), 297–323.

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

Solyalı, O. and Süral, H. (2011), 'A branch-and-cut algorithm using a strong formulation and an a priori tour-based heuristic for an inventory-routing problem', *Transportation Science* **45**(3), 335–345.

Van Roy, T. J. and Wolsey, L. A. (1986), 'Valid inequalities for mixed 0–1 programs', *Discrete Applied Mathematics* **14**(2), 199–213.