Enhanced solution representations for vehicle routing problems with split deliveries

Enhanced Solution Representations for Vehicle Routing Problems with Split Deliveries

Wenbin Zhu[a], Zhuoran Ao[a], Roberto Baldacci[b], Hu Qin[*,c], Zizhen Zhang[d]

[a] *School of Business Administration, South China University of Technology, 510640 Guangzhou, China*

[b] *College of Science and Engineering (CSE), Hamad Bin Khalifa University (HBKU), Doha P.O. Box 5825, Qatar*

[c] *School of Management, Huazhong University of Science and Technology, No. 1037, Luoyu Road, Wuhan, China*

[d] *School of Data and Computer Science, Sun Yat-Sen University, Guangzhou 510275, PR China*

Abstract

In this paper, we investigate a forest-based solution representation for split delivery vehicle routing problems (SDVRPs), which find several practical applications and are among the most difficult vehicle routing problems. The new solution representation fully reflects the nature of split delivery, and can help reduce the search space when being used in heuristic algorithms. Based on the forest structure, we devise three neighborhood search operators, and to highlight the effectiveness of this solution representation, we integrate these operators into a standard tabu search framework.

We conduct extensive experiments on three main vehicle routing problems with split deliveries addressed in the literature, namely, the basic SDVRP, the multi-depot SDVRP (MDSDVRP) and the SDVRP with time windows (SDVRPTW). The experimental results show that the new forest-based solution representation is particularly effective in designing and implementing neighborhood operators, and that the new algorithm is highly competitive or outperforms the state-of-the-art methods on standard data sets.

Key words: vehicle routing; split delivery; multi-depot; time windows; tabu search

[*]Corresponding author

Email addresses: i@zhuwb.com (Wenbin Zhu), 423740893@qq.com (Zhuoran Ao), r.baldacci@unibo.it (Roberto Baldacci), tigerqin1980@qq.com (Hu Qin), zhangzizhen@gmail.com (Zizhen Zhang)

1. Introduction

The solution of a vehicle routing problem (VRP) calls for the design of a set of routes, each performed by a vehicle starting and ending at the depot, such that all customers are serviced, a set of operational constraints are satisfied, and the total route cost is minimized.

In this paper, we address an important member of the VRP family, the Split Delivery Vehicle Routing problem (SDVRP), in which an unlimited number of homogeneous and capacitated vehicles are dispatched to serve a set of customers, each with a non-negative demand, and where each customer can be served by more than one vehicle. The objective is to construct a set of *delivery patterns*, starting from and ending at the depot, to satisfy the demands of all the customers and such that the total routing cost is minimized. A *delivery pattern* is defined as a sequence of customers (i.e., a route) together with the quantity delivered to each customer. Obviously, when the demand of a customer is greater than the vehicle capacity, it has to be split and the customer has to be visited more than once. As shown in Dror and Trudeau (1989), when all customer demands are less than or equal to the vehicle capacity, split delivery can lead to substantial cost saving.

Also in this paper, we consider two relevant generalizations of the basic SDVRP, the multi-depot SDVRP (MDSDVRP) and the SDVRP with time windows (SDVRPTW). The MDSDVRP generalizes the SDVRP by dispatching vehicles from multiple depots, while the SDVRPTW imposes that each customer is to be visited within a specified time interval, called *time window*.

In literature, several variants of the basic SDVRP have been studied. Some of these variants use additional assumptions in order to simplify the problem to certain extents. For example, Jin et al. (2007) study a variant of SDVRP in which the number of vehicles is fixed to the *minimum possible number*. Another variant, called the k-SDVRP (Archetti et al., 2006), where k denotes the maximum quantity delivered in each route, requires that the vehicle capacity, all customer demands and the quantity delivered to each customer must be *integral numbers*. Our work considers more general assumptions, in which the vehicle capacity and customer demands are not required to be integers, the number of vehicles is not limited to the minimum possible number, and the customer demands may exceed the vehicle capacity. In the following, the term SDVRP is used to refer to this latter problem variant.

SDVRPs are among the most difficult VRPs, because additional decision variables have to be included to decide the delivery quantity at each customer and for each vehicle. Computing optimal solutions for large-size SDVRP instances (e.g., instances with more than 100 customers) is difficult (Archetti et al., 2011b, 2014). Therefore, most of the studies in the literature adopt heuristic (or metaheuristic) methods to obtain near-optimal solutions of SDVRPs. The key to designing an efficient heuristic relies on the *solution representation*, i.e., on the method used to represent feasible solutions of the problem. In contrast to previous heuristic approaches using a natural *delivery-pattern-based* representation and thus under utilizing the properties of SDVRPs, our approaches combine a *route-based* and *forest-based* representations of a solution such that the delivery quantity at each customer is not explicitly specified, but can be efficiently computed by means of the *forest-based* representation.

1.1. Contributions of this paper

The contributions of this paper are as follows.

i) We design a novel way to represent solution of SDVRPs, which uses a *forest* structure to detect the feasibility of a solution;

ii) Based on the new solution representation, we introduce new neighborhood search operators to improve the solution quality. The new operators can significantly reduce the time complexity in exploring the solution space;

iii) We propose a simple tabu search framework for solving three main SDVRP variants, the basic SDVRP, the MDSDVRP, and the SDVRPTW. The computational experiments show that such standard approach suffices to yield excellent results compared with several state-of-the-art methods.

What is more, we collect several data sets of SDVRPs, and we provide detailed benchmark results which can serve as baselines for future research.

The rest of this paper is organized as follows. The next section gives an overview of the literature related to SDVRPs. Section 2 formally describes the SDVRPs addressed in this paper, and

reviews some well-known properties of this class of problems. Section 3 introduces the solution representation, and three classes of neighborhood operators, followed by the details of the main components of our solution approaches reported in Section 4. The computational studies are given in Section 5. Finally, Section 6 concludes the paper and indicates future research directions.

1.2. Literature review

The SDVRP has been addressed by many articles in the literature, most of them dealing with heuristic techniques, and surveys on the SDVRP and related problems can be found in Archetti and Speranza (2008) and in Irnich et al. (2014).

The SDVRP with the minimum possible number of vehicles was solved by a dynamic programming method by Lee et al. (2006) and an iterative two-stage algorithm with valid inequalities by Jin et al. (2007). The most recent exact algorithms for the SDVRP have been proposed by Archetti et al. (2011a) and Archetti et al. (2014). In Archetti et al. (2011a), a branch-and-price-and-cut (BPC) method was described for both the case where the fleet of vehicles is unlimited and the case where the fleet is limited to the minimum possible number of vehicles. Instances with up to 48 customers were consistently solved to optimality and an instance with 144 customers was also solved to optimality. Archetti et al. (2014) described two exact branch-and-cut (BC) algorithms based on two relaxed formulations and on procedures to obtain feasible solutions to the SDVRP from feasible solutions of the relaxed formulations. The authors reported results showing that one of the two proposed BC algorithms was capable to solve most of the instances with up to 50 customers and two instances with 75 and 100 customers. Desaulniers (2010) described a BPC algorithm to solve the SDVRPTW. Archetti et al. (2011b) enhanced the BPC algorithm of Desaulniers (2010) by introducing a tabu search algorithm to heuristically solve the pricing problem, several classes of valid inequalities and a new separation algorithm for the k-path inequalities. Instances with up to 100 customers were solved to optimality within a one-hour time limit. Recently, Luo et al. (2017) investigated a BPC algorithm to solve an extension of the SDVRPTW, called the SDVRPTW with linear weight-related cost (SDVRPTWL), in which the travel cost per unit distance is charged based on a linear function of the load weight. The algorithm was tested on both SDVRPTW and SDVRPTWL instances, and instances with up to 100 customers for both

the two variants were solved to optimality within a one-hour time limit. The k-SDVRP with time windows and k-SDVRP with the minimum possible number of vehicles were solved by a BPC algorithm by Salani and Vacca (2011) and by a cutting-plane algorithm by Belenguer et al. (2000), respectively.

More research efforts have been spent in designing heuristics for SDVRPs. Dror and Trudeau (1989) designed a local search algorithm which employs two types of problem-specific local search operators, namely the *k-split interchange* and *route addition*. Archetti et al. (2006) proposed a tabu search algorithm to solve the k-SDVRP that can also deal with the more general SDVRP. Further, Archetti et al. (2008) designed an optimization-based heuristic based on the tabu search algorithm of Archetti et al. (2006). In this heuristic, the tabu search algorithm is first executed to solve the SDVRP. The solutions computed by the tabu search procedure are then used to guide the algorithm toward promising feasible regions. Finally, a mixed integer programming (MIP) model is solved to generate high-quality solutions. Zhang et al. (2015) also developed a tabu search algorithm for the SDVRP based on alternative solution representations. Our work is inspired by the work of Zhang et al., which it extends in several directions, i.e., enhanced solution representations, improved neighborhood search operators, and extension to other SDVRPs than the basic SDVRP. Chen et al. (2007) developed a heuristic that combines an MIP model and a record-to-record travel algorithm. Their MIP model is used to re-allocate the demands of "endpoints", where an endpoint is referred to as the first or the last non-depot vertex visited by the vehicle. After the re-allocation, an endpoint may be removed from its current route by allocating its demand to other routes. In Derigs et al. (2010), the authors described four local search operators adapted from the inter-tour and intra-tour exchange operators for the classical VRP. The authors integrated these operators into five metaheuristic frameworks, namely, Simulated Annealing, Threshold Accepting, Record-to-Record Travel, the Attribute Based Hill Climber heuristic and the Attribute Based Local Beam Search heuristic. Their experimental results show that the Attribute Based Hill Climber heuristic outperforms the other heuristics. Heuristic algorithms for the SDVRP with the minimum possible number of vehicles are the scatter search algorithm by Mota et al. (2007), the two column generation based heuristics by Jin et al. (2008) and Archetti et al. (2011a), the adaptive memory algorithm by Aleman et al. (2010), and the tabu search algorithm with vocabulary building approach by

Aleman and Hill (2010). Berbotto et al. (2013) proposed a randomized granular tabu search algorithm for the k-SDVRP with the minimum possible number of vehicles, whereas Ho and Haugland (2004) designed a tabu search algorithm for the SDVRPTW. Gulczynski et al. (2011) developed an integer programming based heuristic for the MDSDVRP, and Ray et al. (2014) investigated an integer linear programming model and used an heuristic search method for solving the same problem.

2. The SDVRP, MDSDVRP and SDVRPTW: problems definitions

The SDVRP is defined on a complete and undirected graph $G = (V, E)$, where $V = \{v_0, v_1, \ldots, v_n\}$ is a vertex set and $E = \{(i, j) : i, j \in V, i \neq j\}$ is an edge set. Node v_0 represents the depot and the set of remaining vertices $V_c = \{v_1, \ldots, v_n\}$ denotes the set of n customers. Each customer $i \in V_c$ has a positive demand d_i and each edge (i, j) has a non-negative traveling or routing cost $c_{i,j}$, where the cost matrix $[c_{i,j}]$ satisfies the triangle inequality. An unlimited number of homogeneous vehicles are available, each with capacity Q.

A *route* $r = (v_0, i_1, \ldots, i_h, v_0)$ is defined as a simple cycle of graph G passing through depot v_0 and visiting a set of customers $\{i_1, \ldots, i_h\} \subseteq V_c$, and the cost of a route is computed as the sum of the costs of the edge set traversed by the route. A *delivery pattern* p is represented by an underlying route $r = (v_0, i_1, \ldots, i_h, v_0)$ and associated delivery quantity $\delta_{p,i}$ at each visited customer i. A delivery pattern is feasible if its total delivery quantity is less than or equal to the vehicle capacity Q, i.e., $\sum_{i \in \{i_1, \ldots, i_h\}} \delta_{p,i} \leq Q$. The SDVRP consists of designing a set of delivery patterns such that all customers are fully served and the sum of the corresponding route costs is minimized.

The MDSDVRP is the generalization of the SDVRP where the vertex set V is replaced with set $W \cup V_c$, where $W = \{w_1, \ldots, w_m\}$ is a set of depots. In the MDSDVRP, a vehicle route starts from any depot of the set W, and ends at the same depot. The SDVRPTW also generalizes the SDVRP by associating with each customer $i \in V_c$ a prescribed time interval or time window $[e_i, l_i]$. The service for customer i must start within its time window, and if the vehicle arrives at customer i before e_i, the service is delayed to time e_i.

It is worth noting that a solution of the SDVRP (also of the MDSDVRP and the SDVRPTW) is a set of delivery patterns while the corresponding solution cost, namely, the total routing cost,

is only determined by the underlying set of routes. The delivery quantity at each visited customer does not affect the solution value.

According to Dror and Trudeau (1989), the following properties hold for the SDVRP.

Property 1. *If the cost matrix* $[c_{i,j}]$ *satisfies the triangle inequality, then there exists an optimal solution in which no two vehicles have more than one split customer in common.*

A second property is based on the following definition of k-split cycle.

Definition 1. *Given* k *customers* $\{i_1, i_2, \ldots, i_k\}$ *and* k *delivery patterns* $\{p_1, p_2, \ldots, p_k\}$*, these* k *delivery patterns contain a* k*-split cycle if* p_1 *includes* i_1 *and* i_2*,* p_2 *includes* i_2 *and* i_3*, ...,* p_{k-1} *includes* i_{k-1} *and* i_k *and* p_k *includes* i_k *and* i_1*.*

Property 2. *If the cost matrix* $[c_{i,j}]$ *satisfies the triangle inequality, then there exists an optimal solution that does not include* k-split cycle *for any* $k \geq 2$*.*

It is easy to prove that Properties 1 and 2 also hold for MDSDVRP, because the optimal solution to MDSDVRP can be decomposed into a set of m SDVRP optimal solutions (Azad et al., 2017), each of which satisfies these two properties. According to Ho and Haugland (2004), the properties also hold for SDVRPTW.

3. Description of the solution representation and definition of neighborhood structures

In this section, we address two important components in designing efficient and effective heuristic algorithms for SDVRPs, namely, the solution representation and the neighborhood structures.

In literature, neighborhood operators originally designed for VRPs have been modified to handle SDVRPs, see, for example, Dror and Trudeau (1989); Ho and Haugland (2004); Derigs et al. (2010); Aleman et al. (2010); Berbotto et al. (2013). These neighborhood operators represent a feasible solution by a set of delivery patterns, as shown by the example reported in Figure 1. In the figure, six delivery patterns are represented for an SDVRP instance involving 15 customers, and each value δ_p represents the total delivery quantity of pattern p.

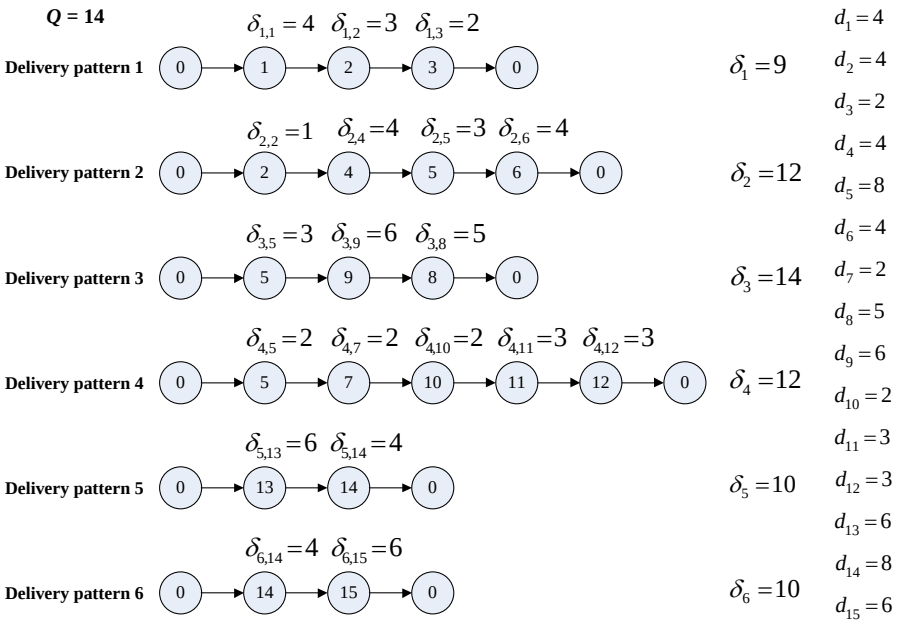


Figure 1: An example of solution involving six delivery patterns.

To show the limits of the representation, consider, for example, the operator *relocate* used in Berbotto et al. (2013) and *2-split interchange* used in Dror and Trudeau (1989). In Figure 1, if $c_{2,1} + c_{1,4} - c_{2,4} < c_{0,1} + c_{1,2} - c_{0,2}$, then relocating customer 1 from delivery pattern 1 to pattern 2 in the position between customers 2 and 4, can reduce the total traveling cost. However, this results in an infeasible solution since the vehicle capacity restriction would be violated. We can make this relocation possible by re-assigning the delivery quantities of the split customers. For example, we can move from delivery pattern 2 two units of customer 5's demand to delivery pattern 4 and then relocate customer 1 to delivery pattern 2. Analogously, allocating customer 8's demand into delivery patterns 2 and 4 by using the 2-split interchange operation also results in an infeasible solution due to the violation of vehicle capacity. We can make this 2-split interchange operation feasible by first moving from delivery pattern 2 one unit of customer 2's demand to delivery pattern 1.

The above observations suggest that if we have a mechanism to re-allocate split customers' demands automatically and freely among all involved delivery patterns, most of the previously proposed operators would be capable of exploring a larger solution space. This can be achieved,

for example, by replacing the traditional *delivery-pattern-based* representation with a *forest-based* solution representation as described in the following section.

3.1. A forest-based solution representation

A solution S, as defined in Section 2, consists of a set of delivery patterns in which a vertex sequence of each route and delivery quantity at each customer are specified. In SDVRPs, changing the delivery quantities may affect the feasibility of a solution but has no impact on the solution cost. Therefore, given the vertex sequence of each route of solution S, if we can find an appropriate delivery quantity at each customer such that the demand of each customer can be fulfilled and the capacity of each vehicle is not exceeded, then S is feasible and the corresponding solution value can be computed as the sum of the route costs of solution S. This observation implies that a solution representation where delivery quantities are not explicitly maintained may be better for SDVRPs. We therefore consider a new solution representation where a *route-based* representation modeling the route sequences is coupled with a *forest-based* representation modeling delivery quantities.

Given the customer set V_c and an index set P of delivery patterns with unknown delivery quantities in solution S, the following model defines a feasible region of possible delivery quantities:

$$\sum_{j \in V_c} \delta_{i,j} \leq Q, \forall\ i \in P, \tag{1a}$$

$$\sum_{i \in P} \delta_{i,j} \geq d_j, \forall\ j \in V_c, \tag{1b}$$

$$\delta_{i,j} \geq 0, \forall\ i \in P, j \in V_c. \tag{1c}$$

Constraints (1a) state that the total delivery quantities in a vehicle route must not exceed the vehicle capacity. Constraints (1b) ensure that the demand of each customer has to be satisfied. It can be easily seen that the coefficient matrix of the above set of inequalities is *totally unimodular*. Hence, if values Q and d_j are integral, the feasible region is an integral polyhedron. Indeed, an integral feasible solution (if any) of the set of inequalities (1) can be determined by solving a

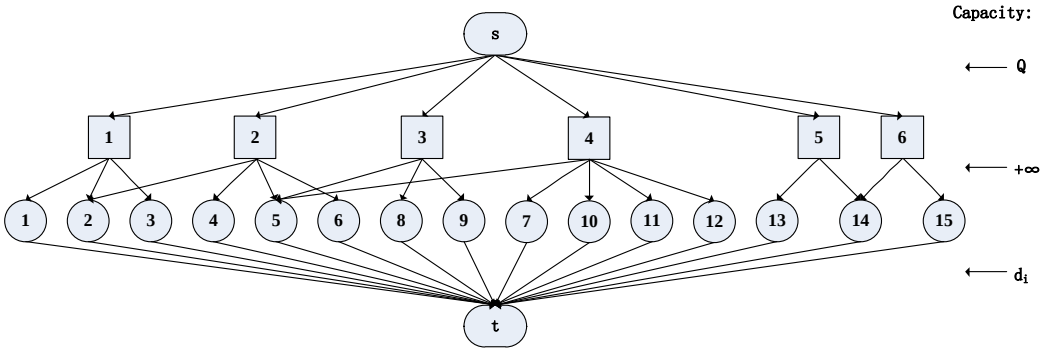


Figure 2: Flow network for the solution of Figure 1, where the squares and circles represent the route and customer nodes, respectively.

maximum flow problem (Ahuja et al., 1993), as also shown below.

Given a solution S, we first construct a bipartite graph $G(S) = (N(S), E(S))$. The node set $N(S)$ is composed of two sets of nodes, namely, a route node set $N_r(S)$ and a customer node set $N_c(S)$, where node $i \in N_r(S)$ corresponds to route index $i \in S$ and node $j \in N_c(S)$ corresponds to customer $j \in V_c$. The edge set $E(S)$ contains an edge (i, j) only if customer j is served by route i. The corresponding *flow network* is then defined by a capacitated directed graph obtained from graph $G(S)$ as follows.

1) The set of nodes is composed of the sets of nodes $N_r(S)$, $N_c(S)$, and two additional nodes, a source node s and a sink node t;

2) The set of arcs is defined as follows.

   (a) An arc (s, i) with capacity Q for each $i \in N_r(S)$;

   (b) An arc (j, t) with capacity d_j for each $j \in N_c(S)$;

   (c) An arc (i, j) with capacity equal to $+\infty$ for each undirected edge $(i, j) \in E(S)$.

Note that the maximum flow in the flow network is limited by value $\sum_{i \in V_c} d_i$. Figure 2 shows the flow network corresponding to the solution of Figure 1. In the figure, the squares and circles represent the route and customer nodes, respectively.

Based on the flow network defined above, model (1) admits a feasible solution if and only if the maximum s-t flow in the flow network is equal to $\sum_{j \in V_c} d_j$. If a feasible solution exists, the

delivery quantities $\delta_{i,j}$, $\forall i \in N_r(S)$ and $\forall j \in N_c(S)$, of the model are equal to the flow values on the corresponding arcs (i, j). Although the maximum s-t flow problem can be solved in polynomial time (e.g., in $O(|N(S)||E(S)|^2)$ by using the Ford-Fulkerson's algorithm), it is not efficient enough to be used in practice since, once used within an heuristic framework for SDVRPs, it must be executed many times. Below we investigate structures of the optimal SDVRP solutions that can help accelerate the problem of checking the feasibility of a route-based representation.

Let Ω be the set of all feasible solutions of the SDVRP. If the cost matrix $[c_{i,j}]$ satisfies the triangle inequality, then all solutions that do not satisfy properties 1 and 2 can be removed from set Ω, and the bipartite graph $G(S)$ associated with an optimal solution $S \in \Omega$ is a forest, as shown by the following proposition.

Proposition 1. *Given an optimal solution $S \in \Omega$, the bipartite graph $G(S) = (N(S), E(S))$ is a forest, i.e., $G(S)$ is an acyclic graph.*

Proof. We prove it by contradiction, by showing that graph $G(S)$ is acyclic. Suppose that $G(S)$ contains a cycle $C = (s_1, s_2, \ldots, s_k = s_1)$, where $k \geq 3$ is the length of the cycle. Value k must also be odd, because $G(S)$ is a bipartite graph. Moreover cycle C alternates between customer and route nodes. Without loss of generality, we can assume that $s_1 (= s_k)$ is a customer node. Then C has the form $(i_1, j_1, i_2, j_2, \ldots, i_{(k-1)/2}, j_{(k-1)/2}, i_1)$, where vertices in odd and even positions corresponds to customer and route nodes, respectively. This contradicts Property 2. □

Proposition 1 also indicates that graph $G(S)$ consists of one or more connected components, each of which is a tree. For the sake of convenience, we represent $G(S)$ as $G(S) = \{T_1, T_2, \ldots, T_g\}$ where T_k, $1 \leq k \leq g$, is a tree. For the example of solution given in Figure 1, $G(S)$ is composed of two trees T_1 and T_2, as shown by Figure 3.

To reduce the computation of checking the feasibility of a route-based representation, below we describe a *greedy* procedure (GP) with the aim of quickly computing a feasible delivery pattern (if any) associated with graph $G(S)$, instead of directly finding the maximum flow on the corresponding flow network. The procedure works as follows.

Let $residual(i)$ and $residual(j)$ be the *residual capacity* of route i and the *residual demand* of customer j, respectively. Initially, we set $residual(i) = Q$, $\forall i \in N_r(S)$, and $residual(j) = d_j$,

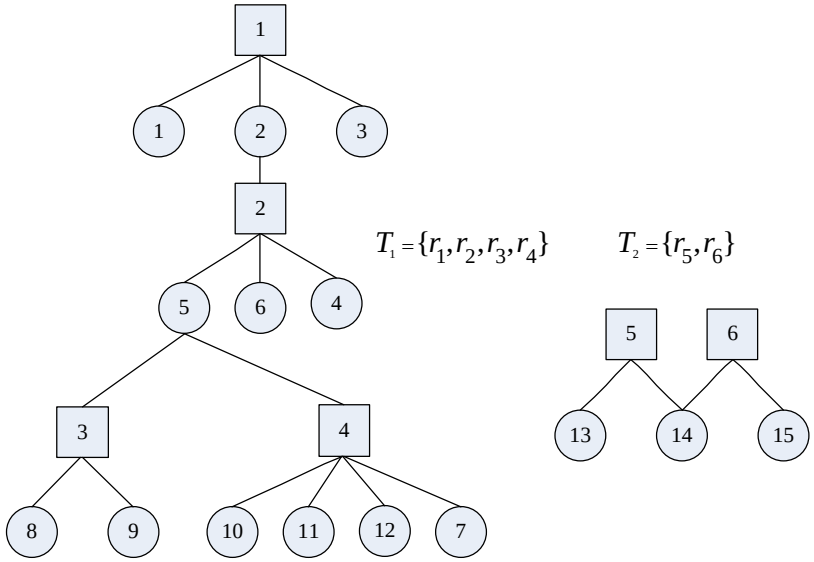


Figure 3: Graph $G(S)$ corresponding to the solution S shown in Figure 1.

$\forall j \in N_c(S)$. Procedure GP performs the following steps.

Let *residual*(i) and *residual*(j) be the *residual capacity* of route i and the *residual demand* of customer j, respectively. Initially, we set *residual*(i) = Q, $\forall i \in N_r(S)$, and *residual*(j) = d_j, $\forall j \in N_c(S)$. Procedure GP performs the following steps.

1) Set $\delta_{i,j} = 0$, for all arcs (i, j) of the flow network.

2) Select a "non-processed" tree T from the forest $G(S)$. If no tree T can be determined, the procedure ends with a feasible solution represented by the delivery quantities $\{\delta_{i,j}\}$.

3) If T consists of only one node $i \in N_r(S)$, then set $T = \emptyset$ (i.e., T has been processed), and go to step 2.

4) If T consists of only one node $j \in N_c(S)$, we have the following two cases.

   a) *residual*(j) > 0: the procedure terminates without having found a feasible solution.

   b) *residual*(j) = 0: set $T = \emptyset$ and go to step 2.

5) Select a leaf node i of T and its father node j. Note that a node is a leaf node if its degree is equal to 1.

6) If node $i \in N_c(S)$ and node $j \in N_r(S)$, we have the following two cases.

   a) $residual(i) \leq residual(j)$: set $\delta_{i,j} = residual(i)$, $residual(j) = residual(j) - residual(i)$ and $residual(i) = 0$. Then, remove node i from T.

   b) $residual(i) > residual(j)$: the procedure terminates without having found a feasible solution.

7) If node $i \in N_r(S)$ and node $j \in N_c(S)$, we have the following two cases.

   a) $residual(i) \leq residual(j)$: set $\delta_{j,i} = residual(i)$, $residual(j) = residual(j) - residual(i)$ and $residual(i) = 0$. Remove node i from T.

   b) $residual(i) > residual(j)$: set $\delta_{j,i} = residual(j)$, $residual(i) = residual(i) - residual(j)$ and $residual(j) = 0$. Then, remove node i from T.

8) Go to step 3.

Concerning tree T_1 of the example shown by Figure 3, procedure GP works as follows.

(i) Customers 10, 11, 12 and 7 are served by vehicle 4; the residual capacity of vehicle 4 is equal to 4.

(ii) Customers 8 and 9 are served by vehicle 3; the residual capacity of vehicle 3 is equal to 3.

(iii) Vehicle 3 delivers 3 units and vehicle 4 delivers 4 units to customer 5. Then the residual capacities of vehicles 3 and 4 are both equal to zero, and the residual demand of customer 5 is equal to 1.

(iv) Vehicle 2 fully serves customers 2, 4, 6 and delivers 1 unit to customer 5. Then, its residual capacity becomes equal to 1.

(v) Customers 1 and 3 are served by vehicle 1 whose residual capacity is 8.

The following theorem proves the correctness of procedure GP, i.e., that procedure GP is capable of computing the maximum *s-t* flow value.

Theorem 1. *Procedure GP returns a feasible solution if and only if the maximum flow value in the flow network corresponding to graph $G(S)$ is equal to $\sum_{j\in V_c} d_j$.*

Proof. Proof. First, if the greedy procedure terminates with a feasible solution at step 2, the residual demand of each customer $j \in N_c(S)$ is equal to 0, hence $\sum_{i\in N_r(S)} \sum_{j\in N_c(S)} \delta_{i,j} = \sum_{j\in V_c} d_j$, i.e., the maximum flow in the flow network is equal to $\sum_{j\in V_c} d_j$. In this case, the greedy procedure essentially performs a series of *augmenting path* steps in the flow network.

Second (by contradiction), assume that the network has maximum flow value equal to $\sum_{j\in V_c} d_j$ with flow value $\delta^*_{i,j}$ associated with each arc (i, j) $(i \in N_r(S), j \in N_c(S))$, and the greedy procedure terminates without having found a feasible solution. Before terminating, the greedy procedure iteratively removes an arc (i, j) $(i \in N_r(S), j \in N_c(S))$ from some tree associated with delivery quantity $\delta_{i,j}$. Let sequence of the arcs removed be $(i_1, j_1), (i_2, j_2), \ldots, (i_l, j_l)$, and assume that the greedy procedure terminates immediately after checking node $j \in N_c(S)$ at step 6. We have two cases:

(i) If $\delta^*_{i_1,j_1} = \delta_{i_1,j_1}, \cdots, \delta^*_{i_l,j_l} = \delta_{i_l,j_l}$, the residual demand of customer j cannot be fulfilled by the greedy procedure but can be satisfied according to the maximum flow, hence we have a contradiction.

(ii) There must exist an index $k \leq l$ such that $\delta^*_{i_1,j_1} = \delta_{i_1,j_1}, \cdots, \delta^*_{i_{k-1},j_{k-1}} = \delta_{i_{k-1},j_{k-1}}$ and $\delta^*_{i_k,j_k} \neq \delta_{i_k,j_k}$. According to our algorithm, $\delta^*_{i_k,j_k} < \delta_{i_k,j_k}$ holds because δ_{i_k,j_k} fully uses the residual capacity of nodes i_k and j_k. Considering customer node j_k in graph $G(S)$, it is connected with a set of route nodes $\{i_k, i', i'', \ldots\}$, as shown in Figure 4. We increase $\delta^*_{i_k,j_k}$ to δ_{i_k,j_k}, while decreasing $\{\delta^*_{i',j_k}, \delta^*_{i'',j_k}, \ldots\}$ of the same amount. Hence, the new δ^* values also correspond to a maximum flow of value equal to $\sum_{j\in V_c} d_j$ and, iteratively, solution δ^*_{ij} can be transformed such that $\delta^*_{i_1,j_1} = \delta_{i_1,j_1}, \cdots, \delta^*_{i_l,j_l} = \delta_{i_l,j_l}$, thus obtaining case (i).

□

Procedure GP can be executed in $O(|N(S)| + |E(S)|)$ time by using a *forest traversal* method from leaves to the root, e.g., a *depth-first* traversal or a *level-by-level* traversal. Therefore, it is very efficient for checking the feasibility of a route-based solution. In the following, we describe

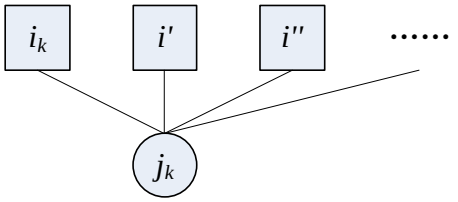


Figure 4: Example where a customer node is connected with a set of route nodes.

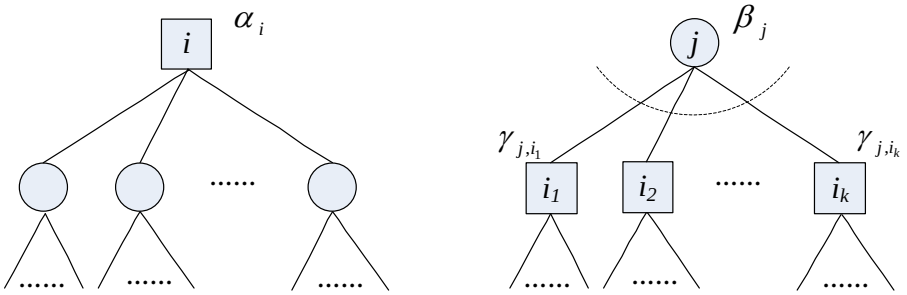


Figure 5: Computing the maximum residual capacity, the maximum supporting capacity and the maximum available capacity.

a route-based heuristic method that incorporates the auxiliary graph $G(S)$ to represent a solution $S \in \Omega$.

The forest-based solution representation can be easily adapted to deal with the MDSDVRP by considering a set of forests for each depot used in a solution.

3.2. Definition of the neighborhood structures

In this section, we describe three operators for SDVRPs, namely, *relocate, exchange* and *split*, to define the neighborhoods of a solution. The first two operators are adapted from classic VRP operators whereas the third operator is based on the *k-split interchange* operator proposed in Dror and Trudeau (1989). All of them are specifically tailored to the forest-based solution representation. To speed up the computation, the following attributes of a solution, also illustrated by Figure 5, are used by the different operators.

- *Maximum residual capacity* α_i of a route i. Let i be the node representing the given route in graph $G(S)$. Value α_i can be computed by processing node i as the last node in procedure GP.

- *Maximum supporting capacity* $\gamma_{j,i}$ of route i before serving customer j. Suppose that customer j is served by a set of routes $\{i_1, i_2, \ldots, i_k\}$. Value γ_{j,i_h} $(1 \leq h \leq k)$ can be computed by first removing edge $\{i_h, j\}$ from graph $G(S)$, and then by finding the maximum residual capacity α_i of route i_h.

- *Maximum available capacity* β_j of customer j. Value β_j can be computed as $\sum_{h=1}^{k} \gamma_{j,i_h}$, which is equal to the total maximum supporting capacity from the set of routes serving customer j.

Given the index i, α_i can be computed in $O(|N(S)|)$ time by applying procedure GP with i as root node. Similarly, $\gamma_{j,i}$ and β_j can be obtained in $O(|N(S)|)$ time by setting j as the root node. Therefore, given a solution, values α_i, $\gamma_{j,i}$ and β_j can be computed in $O(|N(S)|^2)$ time.

3.2.1. Relocate operator

The *relocate operator* selects a customer j from a route r_1 and relocates it into a route r_2. We assume that routes r_1 and r_2 are contained in trees T_1 and T_2, respectively. The cost of the resulting solution can be easily recomputed, whereas checking the feasibility depends on the structures of the trees, as described by the two cases below.

- $T_1 \neq T_2$, *External relocate operation*. Consider the example shown by Figure 6. Suppose that customer j (visited by routes 1 and 3) is moved from route 1 into route 2 and inserted between customers 2 and 3, as shown in figures (a) and (b). The resulting solution S' and graph $G(S')$ are depicted in figures (c) and (d), respectively. The operation is feasible if $\beta_j - \gamma_{j,r_1} + \alpha_{r_2} \geq d_j$.

- $T_1 = T_2$, *Internal relocate operation*. In this case, the move producing solution S' can lead to a cycle in the corresponding graph $G(S')$ as shown by Figure 7. Indeed, if customer 2 is moved from route 2 into route 4, a cycle $(4, 3, 1, 2, 4)$ arises, and the resulting operation is not admissible due to Property 2. However, if customer 3 is moved from route 1 to route 2, as shown in figure (b), the resulting graph $G(S')$ is acyclic, and thus this operation is potentially admissible. In this case, we apply procedure GP to T_1 to check the feasibility of the operation.

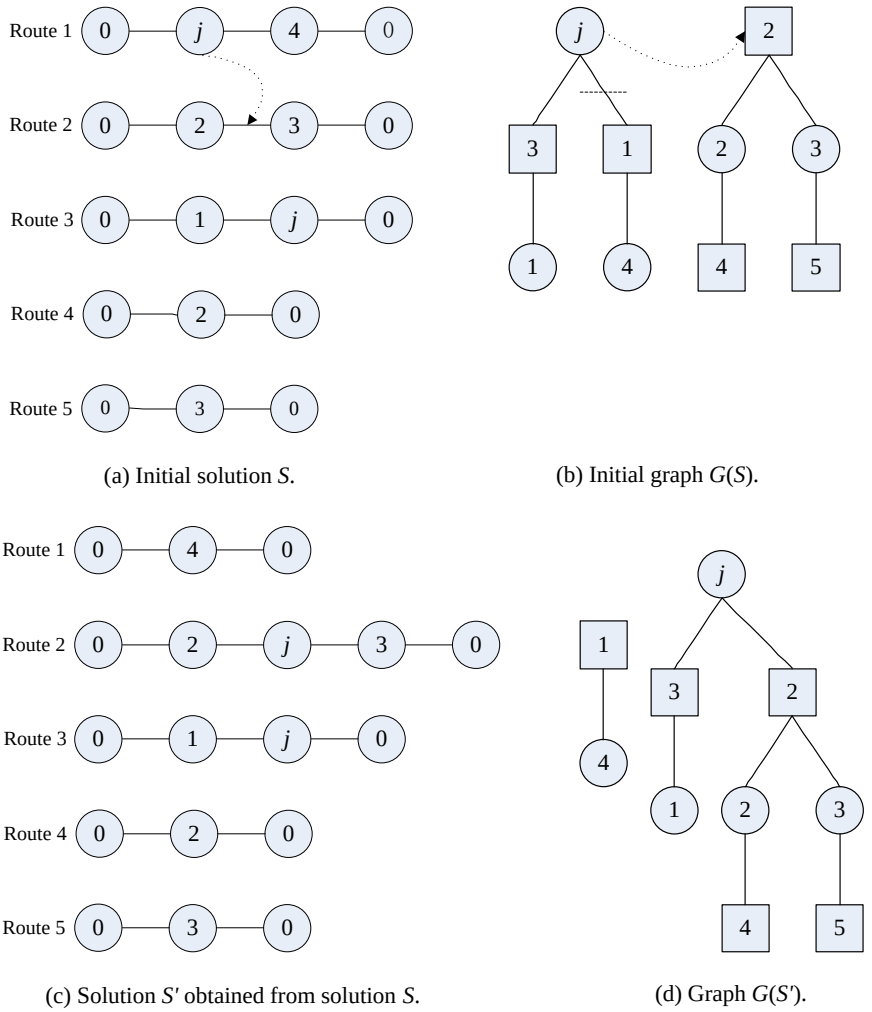


Figure 6: An example of *external* relocate operation.

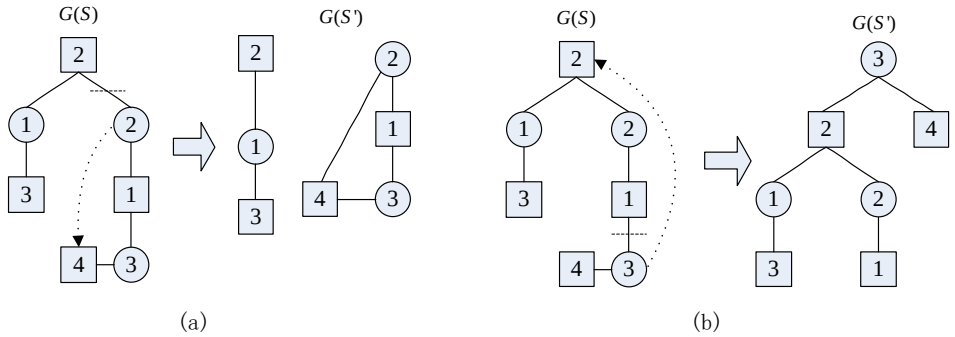


Figure 7: Examples of *internal* relocate operation.

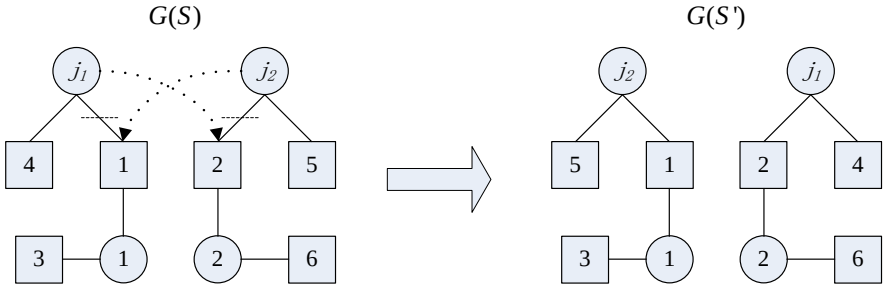


Figure 8: An example of *external* exchange operation.

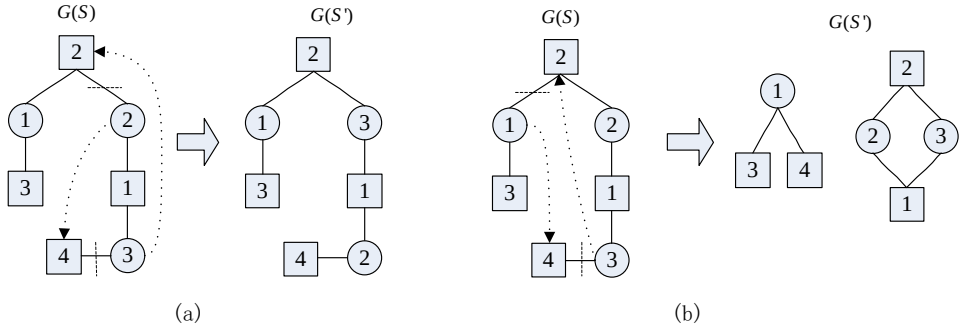


Figure 9: Examples of *internal* exchange operation.

3.2.2. Exchange operator

The *exchange operator* selects two customers from two different routes, and exchanges their corresponding positions. Let i_1 and i_2 be the two selected customers, route r_1 (resp., r_2) be the route containing i_1 (resp., i_2), and T_1 (resp., T_2) be the tree containing route r_1 (resp., route r_2). Similar to the relocate operator, we have the following two cases.

- $T_1 \neq T_2$, *External exchange operation*. Figure 8 shows an example of the exchange of customer j_1 of route 1 with customer j_2 of route 2, and the resulting graph $G(S')$. In this case, the operation is feasible if the following conditions hold: $\beta_{j_1} - \gamma_{j_1,r_1} + \gamma_{j_2,r_2} \leq d_{j_1}$ and $\beta_{j_2} - \gamma_{j_2,r_2} + \gamma_{j_1,r_1} \leq d_{j_2}$.

- $T_1 = T_2$, *Internal exchange operation*. In this case, a cycle in the resulting graph $G(S')$ can occur (see, for example, Figure 9(b)). When the resulting graph $G(S')$ is acyclic, procedure GP is applied to T_1 to verify the feasibility of the operation.

3.2.3. Split operator

The *split operator* is based on the *k-split interchange* operator proposed by Dror and Trudeau (1989). The operator tries to replace a set of routes serving customer j (denoted by R_j) with a new set of routes (denoted by R'_j) such that the total routing cost is reduced.

Given a customer j, the operator first computes the saving $SAV_j(i)$ corresponding to the removal of j from route $i \in R_j$ as $SAV_j(i) = c_{j^-,j} + c_{j,j^+} - c_{j^-,j^+}$, where j^- and j^+ are the immediate predecessor and successor of customer j in route i. The operator then computes the routing cost $CI_j(i)$ for inserting the customer j into every route i in the solution. If $i \notin R_j$, each position in i is examined and the position between customers p and p^+ with the minimum insertion cost is selected, namely, $CI_j(i) = c_{p,j} + c_{j,p^+} - c_{p,p^+}$. If $i \in R_j$, $CI_j(i)$ is set to $SAV_j(i)$. Finally, customer j is removed from every route $i \in R_j$ and reinserted into the routes in R'_j as described below.

The set R'_j is determined by solving a Disjunctively Constrained Knapsack Problem (DCKP) or Knapsack Problem with Conflicts (see Yamada et al., 2002), which is defined as follows. Customer j is regarded as the *knapsack* with capacity d_j. Each route i in the solution is regarded as an *item*, with weight equal to $\gamma_{j,i}$ if $i \in R_j$, or α_i otherwise, i.e, $i \notin R_j$. The value or profit of the item is $CI_j(i)$ – see Figure 10 for an example. The disjunctive constraints require that two conflicting items must not be selected at the same time. Two items i and i' are in conflict, if assigning customer j to both routes i and i' results in a cycle in the resulting graph $G(S')$.

The DCKP is $\mathcal{NP}$-complete, and solving it to optimality can be time consuming. To speed up the computation, we use a greedy heuristic for its solution which works as follows. The heuristic first sorts all the items for decreasing values of the value per unit weight. Then, each item with the associated route i is sequentially checked. If assigning customer j to route i does not generate cycles, customer j is inserted into the best position in route i. Otherwise, the next item is checked. When all items have been checked and the demand of customer j is not fully served (i.e., the knapsack is not full), we create a new route to serve customer j.

Note that for the MDSDVRP, the depot of a newly created route is set to the one closest to customer j. For the SDVRPTW, the insertion of customer j to route i must also consider the time window constraints.

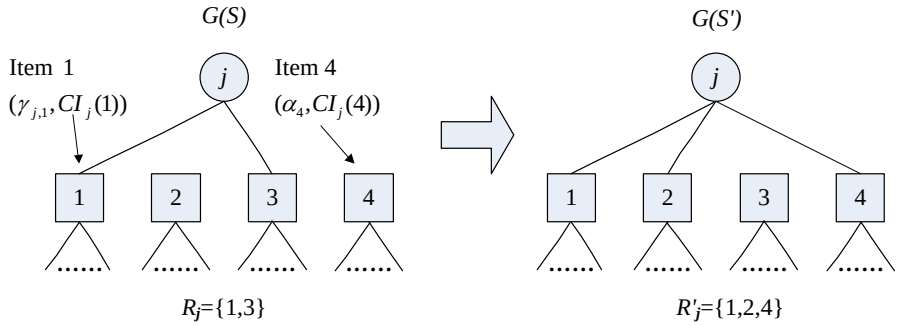


Figure 10: An example of the split operator.

4. A unified tabu search framework for SDVRPs

In this section, we describe a unified tabu search framework for SDVRPs, which is used to solve the three problems considered in this paper, namely, the SDVRP, the MDSDVRP, and the SDVRPTW. The tabu search algorithm (Glover and Laguna, 1998) is a well-known metaheuristic that has been successfully applied to a wide variety of VRPs, such as classic VRP (Gendreau et al., 1994), the SDVRP (Archetti et al., 2006), the Vehicle Routing Problem with Time Windows (VRPTW) (Potvin et al., 1996), the SDVRPTW (Ho and Haugland, 2004) and the three-dimensional loading VRP (Wei et al., 2014), to name a few. The neighborhood operators described in the previous section are integrated into the proposed tabu search framework, and their effectiveness in shown by the computational experiments reported in Section 5. It is worth noting that, the neighborhood operators described in this paper may be also applied within other metaheuristic frameworks. Algorithm 1 presents the tabu search framework, which consists of an initialization phase (step 1), a neighborhood search phase (steps 3-4) and a diversification phase (steps 5-7). Below, the different steps of the algorithm are described in details.

4.1. Neighborhood search and tabu list

Our tabu search algorithm uses a neighborhood search procedure to iteratively move from the current solution S to a neighbor S'. In our implementation, a tabu list management strategy similar to strategies already proposed in the literature (see Cordeau et al., 1997, for example) is included in the neighborhood search procedure to avoid revisiting previously explored solutions.

Algorithm 1 The tabu search framework.

```
S ← initial solution;
while no termination criterion is reached do
    S ← best feasible neighbor S′;
    Update tabu list and aspiration conditions;
    if diversification criterion holds then
        diversification;
    end if
end while
```

The neighborhood search procedure is given in Algorithm 2. We use $f(S)$ to denote the objective function of a solution S, i.e., the total routing cost. Operators *relocate*, *exchange* and *split* are used. The tabu list, denoted by L, is a two-dimensional array, where $L_{i,j}$ records the index of the iteration in which customer j is inserted into the i-th route. We use multiple tabu tenures stored in a two-dimensional array T. $T_{i,j}$ is randomly drawn in the range $[\xi_l, \xi_u]$, where ξ_l and ξ_u are user-defined parameters. Let t be the index of the current iteration. If $t - L_{i,j} \leq T_{i,j}$ and customer j is removed from the route i, then the corresponding operation is forbidden. However, such operation is still available if the best solution S^* found so far can be improved, i.e., an *aspiration criterion* is used.

Algorithm 2 Neighborhood search procedure.

```
S_1 ← relocate(S, S*, t, L, T);
S_2 ← exchange(S, S*, t, L, T);
S_3 ← split(S, S*, t, L, T);
S ← argmin_{S′∈{S_1,S_2,S_3}} f(S′);
t ← t + 1;
Update L;
```

4.2. Diversification mechanism

Diversification helps to prevent the search from trapping in local optima. In our implementation, the diversification process is invoked whenever the best solution currently found is not improved for η iterations, where η is a user-defined parameter. We simply use the multiple-k-split operator proposed by Penna et al. (2013) and Silva et al. (2015) to perturb the solution. Here, k is also a user-defined parameter.

The multiple-k-split operator first randomly selects k customers, with k randomly selected in the range $[k_l, k_u]$. Next, these customers are removed from the solution. Finally, the *split* operator (see Section 3.2.3) is applied to the removed customers in order to reinsert them into the solution. After the solution is perturbed, the tabu list L is cleared and the iteration t is set to 0.

4.3. Termination criteria

The algorithms terminates whenever one of the two following conditions is met: the number of perturbations reaches a pre-defined parameter ρ or the running time exceeds a prescribed time limit τ.

5. Computational experiments

This section reports on the computational results of the tabu search algorithm described in Section 4, hereafter called Forest-based Tabu Search (FTS). All the experiments were conducted on a personal computer equipped with an Intel i7-4790 4.00 GHz CPU, 8 GB RAM, and running under Windows 10 operating system.

Algorithm FTS was tested on sets of instances already proposed in the literature for the SDVRP, the MDSDVRP, and the SDVRPTW. For each instance, as generally done in the literature, algorithm FTS was executed 10 times with different random seeds.

5.1. Parameters calibration

We performed preliminary experiments to produce suitable values for the parameters of FTS. Table 1 gives the final parameter settings used by FTS. In our preliminary experiments, we decided to set the time limit τ equal to 400 seconds whereas the remaining parameters were computed as follows.

We first grouped the parameters into three groups $[\xi_l, \xi_u]$, $[k_l, k_u]$, and (η, ρ), and defined the following ranges for the parameters:

- $[\xi_l, \xi_u]$: $[0.01n, 0.05n]$, $[0.5n, 0.1n]$, $[0.1n, 0.2n]$ and $[0.2n, 0.3n]$;
- $[k_l, k_u]$: $[0, 0]$, $[1, 3]$, $[3, 5]$, $[5, 7]$, $[7, 10]$ and $[10, 15]$ – interval $[0, 0]$ means that no perturbation is used;

Table 1: Parameter settings of algorithm FTS.

Description	Parameter	Values
Range of the tabu tenure	$[\xi_l, \xi_u]$	$[0.05n, 0.1n]$
Range of k in multiple-k-split	$[k_l, k_u]$	$[3, 5]$
No. of non-improved steps	η	20000
No. of perturbations	ρ	10
Time limit	τ	400 (seconds)

- (η, ρ): $(40000, 5)$, $(20000, 10)$ and $(10000, 20)$.

For the sake of the preliminary experiments, we selected the seven SDVRP instances named "p05", and we run algorithm FTS for all combinations of the different parameters. For each combination, we compute the average of the corresponding solution values, and the parameters are defined based on the combination having minimum value.

5.2. Results on the SDVRP

The SDVRP has been extensively studied in the literature, and several benchmark instances have been generated by different authors. We conducted experiments on the following three sets of SDVRP benchmark instances.

- **Set 1**. This set contains the 14 instances proposed by Belenguer et al. (2000) generated by using the generation scheme described by Dror and Trudeau (1989). The vehicle capacity Q of each instance is equal to 160, and the customer demands were randomly generated from six different intervals, namely, $[\lceil 0.01Q \rceil, \lfloor 0.1Q \rfloor]$, $[\lceil 0.1Q \rceil, \lfloor 0.3Q \rfloor]$, $[\lceil 0.1Q \rceil, \lfloor 0.5Q \rfloor]$, $[\lceil 0.1Q \rceil, \lfloor 0.9Q \rfloor]$, $[\lceil 0.3Q \rceil, \lfloor 0.7Q \rfloor]$, and $[\lceil 0.7Q \rceil, \lfloor 0.9Q \rfloor]$.
- **Set 2**. This set contains the 21 instances proposed by Chen et al. (2007). The number of customers in these instances ranges from 8 up to 288. The customer demands are either 60 or 90 units, and the vehicle capacity Q is set equal to 100 for each instance. In each instance, customers are located in concentric circles around the depot.
- **Set 3**. This set of instances is derived from the capacitated VRP (CVRP) benchmark instances used in Gendreau et al. (1994). For each CVRP instance, Archetti et al. (2006)

varied the customer demands according to one of the six demand intervals used by Set 1. We consider six CVRP instances (*p01 – p05* and *p11*), for a total of 42 SDVRP instances.

Algorithm FTS was compared with the results obtained by the following algorithms already proposed in the literature for the SDVRP and evaluated on the above sets of instances.

- SPLITABU: the tabu search algorithm of Archetti et al. (2006).
- VRTR+EMIP: the Variable length Record-To-Record travel algorithm with an Endpoint Mixed Integer Programming algorithm of Chen et al. (2007).
- OH: the Optimization-based Heuristic of Archetti et al. (2008).
- ICA+VND: the Variable Neighborhood Descent algorithm of Aleman et al. (2010), where an Iterated Constructive Algorithm is employed to create the initial solution.
- TSVBA: the tabu Search with Vocabulary Building Approach of Aleman and Hill (2010).
- ABHC: the Attribute Based Hill Climber heuristic of Derigs et al. (2010).
- BPCH: the Branch-and-Price-and-Cut based Heuristic of Archetti et al. (2011a).
- RGTS: the Randomized Granular tabu Search of Berbotto et al. (2013).
- SplitILS: the Iterated Local Search heuristic of Silva et al. (2015).

Table 5 of the e-companion to this paper summarizes the programming languages and experimental environments of the above algorithms.

Table 2 reports the results obtained by the different algorithms on the three sets of SDVRP instances. For each algorithm, the table reports the average of the best solution values computed over the 10 runs, and the corresponding average running time in seconds. If a value is marked with an "-", the set has not been executed by the corresponding algorithm. In the table, the row labelled "BKS" gives the averages of the best solution values found by the different algorithms proposed in the literature (see Silva et al., 2015). Tables 6, 7, and 8 of the e-companion report detailed results about the SDVRP.

Table 2 shows that FTS is, on average, the second best algorithm for the SDVRP. The average percentage gaps of the solutions produced by FTS, SplitILS, BPCH, ICA+VND, RGTS and

Table 2: Summary results on the SDVRP.

	Set 1		Set 2		Set 3	
	Cost	Time	Cost	Time	Cost	Time
BKS	1367.5	-	9002.0	-	2799.5	-
SplitILS	1367.6	114	9006.2	592	2800.7	1063
FTS	1369.8	66	9024.3	176	2814.3	178
BPCH	1390.1	-	9067.2	-	-	-
RGTS	1405.8	25	-	-	2865.4	201
TSVBA	1414.0	134	9043.3	183	-	-
ICA+VND	1445.7	14	9091.6	47	-	-
ABHC	-	-	9092.8	1557	-	-
VRTR+EMIP	-	-	9179.1	2116	-	-
SPLITABU	-	-	-	-	2903.9	-

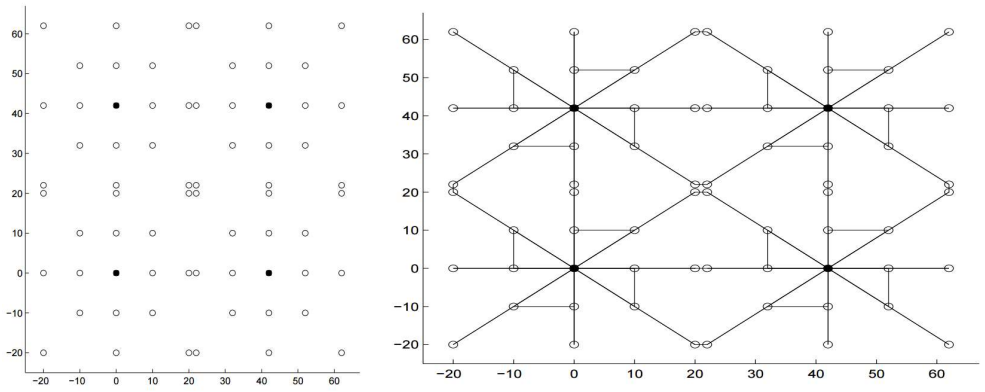
TSVBA with respect to the best values BKS are equal to 0.17%, 0.01%, 1.65%, 5.72%, 2.80% and 3.40%, respectively, thus showing the hight quality of the solutions produced by FTS. The detailed results reported in the e-companion also show that improved solutions can be computed by FTS with respect to SplitILS for some large SDVRP instances, such as instance SD21 involving 288 customers, the largest instances of Set 2 instances.

5.3. Results on the MDSDVRP

Algorithm FTS was tested on the following two sets of benchmark instances proposed in the literature.

- **Set 1**. This set contains 12 instances generated by Gulczynski (2010) having a special geographical structure. As an example, Figure 11 shows the layout of one instance in this set and a corresponding feasible solution computed by Gulczynski (2010).
- **Set 2**. This set contains 30 instances proposed by Gulczynski et al. (2011). All instances were generated from 10 Multi-Depot Vehicle Routing Problem instances generated by Christofides and Eilon (1969) and Gillett and Johnson (1976). The demand of a customer is randomly generated from three intervals $[0.1Q, 0.9Q]$, $[0.3Q, 0.7Q]$ and $[0.7Q, 0.9Q]$.

For all the two set of instances, the number of depots ranges from 2 to 5.



(a) (b)

Figure 11: Layouts of MDSDVRP instance SQ3 and of the corresponding solution computed by Gulczynski (2010).

Table 3: Summary results on the MDSDVRP.

	Set 1		Set 2	
	Cost	Time	Cost	Time
FTS	6731.9	375	7465.7	382
VES	6735.1	-	-	-
HP	6762.4	103	-	-
IDH	6767.4	760	7578.8	777

Algorithm FTS was compared with the results obtained by the following algorithms already proposed in the literature for the MDSDVRP.

- VES: the Visually Estimated Solutions algorithm proposed by Gulczynski et al. (2011);
- IDH: the Inter-Depot Heuristic algorithm proposed by Gulczynski et al. (2011);
- HP: the Heuristic Procedure algorithm proposed by Ray et al. (2014).

Table 9 of the e-companion reports additional details about the algorithms listed above.

Table 3 summarizes the results obtained whereas detailed computational results are given in Tables 10 and 11 of the e-companion. For each algorithm, Table 3 reports the average of the best solution costs computed over the 10 runs, and the corresponding average running time in seconds. If a value is marked with "-", the set has not been executed by the corresponding algorithm. Table 3 clearly shows that on average FTS outperforms all other algorithms. The detailed results reported in the e-companion show that FTS produces the best solutions for almost all the instances.

5.4. Results on the SDVRPTW

SDVRPTW instances were generated by different authors based on instances proposed for the VRPTW (Solomon, 1987). The following two sets of SDVRPTW were tested.

- **Set 1**. This set of instances was proposed by Solomon (1987) and contains 56 VRPTW instances which are divided into six classes (C1, RC1, and R1 with tight time windows and strict vehicle capacity, and C2, RC2, and R2 with wide time windows and loose vehicle capacity). We considered all 100-customer instances.
- **Set 2**. This set of instances was proposed by Ho and Haugland (2004) and derived from VRPTW instances where the original customer demands have been modified as follows. The customer demand d'_i is computed as $d'_i = lQ + Q(u - l)(d_i - d_{min})/(d_{max} - d_{min})$, where d_i is the original VRPTW customer demand, d_{max} and d_{min} are respectively the maximum and minimum customer demands, l and u ($l < u$) are parameters in the interval $(0, 1]$. Ho and Haugland (2004) used the following four combinations of (l, u) values: (0.01,0.05), (0.02,1.00), (0.50,1.00), (0.70,1.00).

Algorithm FTS is compared with the Delivery-pattern-based Tabu Search Heuristic proposed by Ho and Haugland (2004), called algorithm TSH, that was coded in C++ and executed on a Sun Ultra 10 (UltraSPARC-IIi 440 MHz) workstation. Since the best VRPTW solutions provide valid heuristic solutions for the SDVRPTW, we also compare the solutions computed by FTS with the best VRPTW solutions found in the literature (Baldacci et al., 2012). We denote by VRPTW-EA the set of the best solutions found for the VRPTW.

Table 4 summarizes the results obtained on the SDVRPTW. Detailed computational results can be found in Table 12 reported in the e-companion. For each algorithm, Table 4 reports the average of the best solution costs computed over the 10 runs, and the corresponding average running time in seconds. If a value is marked with "-", the set has not been executed by the corresponding algorithm.

The table shows that FTS outperforms on average algorithm TSH. Further, the detailed tables reported in the e-companion show that algorithm FTS was capable of computing improved solu-

Table 4: Summary results on the SDVRPTW.

		FTS		VRPTW-EA	TSH
		Cost	Time	Cost	Cost
Set 1	R1	1190.0	38	1209.9	1247.2
	C1	828.9	34	828.4	833.8
	RC1	1369.2	34	1384.2	1431.9
	R2	905.1	48	951.9	962.2
	C2	600.0	43	589.9	592.6
	RC2	1032.6	50	1119.4	1146.3
Set 2	R1	2961.9	305	-	3096.3
	C1	2847.5	281	-	3073.2
	RC1	4044.8	319	-	4213.1
	R2	2939.7	321	-	3095.5
	C2	2947.6	309	-	3171.0
	RC2	4041.6	328	-	4266.9

tions with respect to existing VRPTW solutions (column VRPTW-EA) which is not the case of algorithm TSH.

6. Conclusions and future research

In this paper, we designed new heuristic algorithms for SDVRPs including the classic Split Delivery Vehicle Routing Problem (SDVRP), the Multi-Depot SDVRP (MDSDVRP) and the SDVRP with Time Windows (SDVRPTW). These problems found several practical applications and are among the most difficult vehicle routing problems.

The main difficulty of SDVRPs lies in how to determine the delivery quantity at each customer. To overcome this difficulty, we proposed a novel method to represent feasible SDVRPs solutions. The method makes use of a forest-based structure that is used to quickly check the feasibility of a solution. Also in this paper, we devised three new forest-based neighborhood operators, namely, *relocate*, *exchange* and *split* operators, and integrated them into a standard tabu search framework.

The new algorithm was tested on several classes of SDVRP, MDSDVRP, and SDVRPTW instances and the results compared with those obtained with state-of-the-art algorithms. The algorithm proposed in this paper turns out to be highly competitive on SDVRP instances, and was capable of computing new best solutions for some large-sized SDVRP instances. Regarding the

MDSDVRP and SDVRPTW, the algorithm on average outperforms the state-of-the-art algorithms for both the two problems.

Based on our forest-based solution representation, future research will develop more advanced neighborhood operators for solving SDVRPs. What is more, future work will generalize the proposed methods to address other variants of vehicle routing problems with split demands that also find several practical applications, such as the class of inventory routing problems and arc routing problems Tang et al. (2017); Shang et al. (2016).

References

Ahuja, R. K., Magnanti, T. L., Orlin, J. B., 1993. Network flows: theory, algorithms, and applications. Prentice hall.

Aleman, R. E., Hill, R. R., 2010. A tabu search with vocabulary building approach for the vehicle routing problem with split demands. International Journal of Metaheuristics 1 (1), 55 – 80.

Aleman, R. E., Zhang, X., Hill, R. R., 2010. An adaptive memory algorithm for the split delivery vehicle routing problem. Journal of Heuristics 16 (3), 441 – 473.

Archetti, C., Bianchessi, N., Speranza, M. G., 2011a. A column generation approach for the split delivery vehicle routing problem. Networks 58 (4), 241 – 254.

Archetti, C., Bianchessi, N., Speranza, M. G., 2014. Branch-and-cut algorithms for the split delivery vehicle routing problem. European Journal of Operational Research 238 (3), 685 – 698.

Archetti, C., Bouchard, M., Desaulniers, G., 2011b. Enhanced branch and price and cut for vehicle routing with split deliveries and time windows. Transportation Science 45 (3), 285 – 298.

Archetti, C., Speranza, M., Hertz, A., 2006. A tabu search algorithm for the split delivery vehicle routing problem. Transportation Science, 64–73.

Archetti, C., Speranza, M., Savelsbergh, M., 2008. An optimization-based heuristic for the split delivery vehicle routing problem. Transportation Science 42 (1), 22–31.

Archetti, C., Speranza, M. G., 2008. The split delivery vehicle routing problem: A survey. In: Golden, B. L., Raghavan, S., Wasil, E. A. (Eds.), The Vehicle Routing Problem: Latest Advances and New Challenges. Springer, New York, pp. 103 – 122.

Azad, A. S., Islam, M., Chakraborty, S., 2017. A heuristic initialized stochastic memetic algorithm for mdpvrp with interdependent depot operations. IEEE Transactions on Cybernetics 47 (12), 4302–4315.

Baldacci, R., Mingozzi, A., Roberti, R., 2012. Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. European Journal of Operational Research 218 (1), 1–6.

Belenguer, J. M., Martinez, M. C., Mota, E., 2000. A lower bound for the split delivery vehicle routing problem. Operations Research 48 (5), 801 – 810.

Berbotto, L., García, S., Nogales, F. J., 2013. A randomized granular tabu search heuristic for the split delivery vehicle routing problem. Annals of Operations Research In press.

Chen, S., Golden, B., Wasil, E., 2007. The split delivery vehicle routing problem: Applications, algorithms, test problems, and computational results. Networks 49 (4), 318–329.

Christofides, N., Eilon, S., 1969. An algorithm for the vehicle-dispatching problem. Journal of the Operational Research Society 20 (3), 309–318.

Cordeau, J. F., Gendreau, M., Laporte, G., 1997. A tabu search heuristic for periodic and multi–depot vehicle routing problems. Networks 30, 105–119.

Derigs, U., Li, B., Vogel, U., 2010. Local search-based metaheuristics for the split delivery vehicle routing problem. Journal of the Operational Research Society 61, 1356 – 1364.

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

Dror, M., Trudeau, P., 1989. Savings by split delivery routing. Transportation Science 23 (2), 141 – 145.

Gendreau, M., Hertz, A., Laporte, G., 1994. A tabu search heuristic for the vehicle routing problem. Management Science, 1276 – 1290.

Gillett, B. E., Johnson, J. G., 1976. Multi-terminal vehicle-dispatch algorithm. Omega 4 (6), 711–718.

Glover, F., Laguna, M. (Eds.), 1998. Tabu Search. Kluwer Academic Publishers, 101 Philip Drive, Assinippi Park, Norwell, Massachusetts 02061, USA.

Gulczynski, D., Golden, B., Wasil, E., 2011. The multi-depot split delivery vehicle routing problem: An integer programming-based heuristic, new test problems, and computational results. Computers & Industrial Engineering 61 (3), 794–804.

Gulczynski, D. J., 2010. Integer programming-based heuristics for vehicle routing problems. University of Maryland, College Park.

Ho, S. C., Haugland, D., 2004. A tabu search heuristic for the vehicle routing problem with time windows and split deliveries. Computers & Operations Research 31 (12), 1947–1964.

Irnich, S., Schneider, M., Vigo, D., 2014. Four variants of the vehicle routing problem. In: Toth, P., Vigo, D. (Eds.), Vehicle Routing. SIAM, Ch. 9, pp. 241–271.

Jin, M., Liu, K., Bowden, R. O., 2007. A two-stage algorithm with valid inequalities for the split delivery vehicle routing problem. International Journal of Production Economics 105 (1), 228 – 242.

Jin, M., Liu, K., Eksioglu, B., 2008. A column generation approach for the split delivery vehicle routing problem. Operations Research Letters 36 (2), 265 – 270.

Lee, C.-G., Epelman, M. A., White III, C. C., Bozer, Y. A., 2006. A shortest path approach to the multiple-vehicle routing problem with split pick-ups. Transportation Research Part B 40 (4), 265 – 284.

Luo, Z., Qin, H., Zhu, W., Lim, A., 2017. Branch and price and cut for the split-delivery vehicle routing problem with

time windows and linear weight-related cost. Transportation Science 51 (2), 668 – 687.

Mota, E., Campos, V., Corberán, A., 2007. A new metaheuristic for the vehicle routing problem with split demands. In: van Hemert, J., Cotta, C. (Eds.), Evolutionary Computation in Combinatorial Optimization, Lecture Notes in Computer Science 4446. Springer, Berlin, pp. 121 – 129.

Penna, P. H. V., Subramanian, A., Ochi, L. S., 2013. An iterated local search heuristic for the heterogeneous fleet vehicle routing problem. Journal of Heuristics, 1–32.

Potvin, J.-Y., Kervahut, T., Garcia, B.-L., Rousseau, J.-M., 1996. The vehicle routing problem with time windows part I: Tabu search. INFORMS Journal on Computing 8 (2), 158 – 164.

Ray, S., Soeanu, A., Berger, J., Debbabi, M., 2014. The multi-depot split-delivery vehicle routing problem: Model and solution algorithm. Knowledge-Based Systems 71, 238–265.

Salani, M., Vacca, I., 2011. Branch and price for the vehicle routing problem with discrete split deliveries and time windows. European Journal of Operational Research 213 (3), 470 – 477.

Shang, R., Dai, K., Jiao, L., Stolkin, R., 2016. Improved memetic algorithm based on route distance grouping for multiobjective large scale capacitated arc routing problems. IEEE Transactions on Cybernetics 46 (4), 1000–1013.

Silva, M. M., Subramanian, A., Ochi, L. S., Jan. 2015. An iterated local search heuristic for the split delivery vehicle routing problem. Computers & Operations Research 53, 234–249.

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

Tang, K., Wang, J., Li, X., Yao, X., 2017. A scalable approach to capacitated arc routing problems based on hierarchical decomposition. IEEE Transactions on Cybernetics 47 (11), 3928–3940.

Wei, L., Zhang, Z., Lim, A., 2014. An adaptive variable neighborhood search for a heterogeneous fleet vehicle routing problem with three-dimensional loading constraints. IEEE Computational Intelligence Magazine 9 (4), 18–30.

Yamada, T., Kataoka, S., Watanabe, K., 2002. Heuristic and exact algorithms for the disjunctively constrained knapsack problem. Information Processing Society of Japan Journal 43 (9).

Zhang, Z., He, H., Luo, Z., Qin, H., Guo, S., 2015. An efficient forest-based tabu search algorithm for the split-delivery vehicle routing problem. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence. AAAI Press, pp. 3432–3438.

This e-companion reports detailed computational results of algorithm FTS on the benchmark instances of the Split Delivery Vehicle Routing Problem SDVRP, the multi-depot SDVRP (MDS-DVRP) and the SDVRP with time windows (SDVRPTW).

EC.1: Detailed results on the SDVRP

In this section, we report the detailed computational results about the SDVRP.

Table 5 reports details about the programming languages and the machines used by the different algorithms proposed in the literature for the SDVRP. Column "SuperPi (1M)" gives a performance evaluation in seconds of the corresponding machine according to the SuperPi (1M) benchmark (`http://www.superpi.net/`). The computing time of SuperPi is an estimate of the single-thread speed of a CPU — the less computing time that a CPU takes, the faster the CPU is. The SuperPi (1M) score of the machine used in our experiments is equal to about 9 seconds, hence our machine is about as fast as the machine used for SplitILS (Silva et al., 2015).

Tables 6 – 8 give detailed results about the three set of SDVRP instances, respectively. These tables show the following columns: the number of customers ("n"), the cost of the best known solution ("BKS", Silva et al. (2015)), and for each algorithm, the average cost of the solutions computed over the ten runs ("Cost"), and the corresponding average computing time ("Time").

The solution values marked with bold numbers are about values less than or equal to the BKS values - an asterisk "*" indicates that the value reported is strictly less than the BKS value.

Table 5: Details of state-of-the-art algorithms for the SDVRP.

Algorithm	Reference	Language	Machine	SuperPi (1M)
SPLITABU	Archetti et al. (2006)	C++	Pentium IV 2.4 GHz + 256 MB RAM	90
VRTR+EMIP	Chen et al. (2007)	C++	Pentium IV 1.7 GHz + 512 MB RAM	120
OH	Archetti et al. (2008)	C++	–	–
ICA+VND	Aleman et al. (2010)	C#	Pentium IV 2.8 GHz	66
TSVBA	Aleman and Hill (2010)	C#	Pentium IV 2.8 GHz	66
ABHC	Derigs et al. (2010)	–	CPU 3.0 GHz + 2 GB RAM	44
BPCH	Archetti et al. (2011a)	C++	Intel Dual Core 2.4 GHz + 3 GB RAM	25
RGTS	Berbotto et al. (2013)	C++	CPU 2.1 GHz + 4 GB RAM	100
SplitILS	Silva et al. (2015)	C++	Intel Core i7 2.93 GHz + 8 GB RAM	10

Table 6: SDVRP: detailed results on Set 1 instances.

Instance	n	BKS	FTS		SplitILS		BPCH	ICA+VND		RGTS		TSVBA	
			Cost	Time	Cost	Time	Cost	Cost	Time	Cost	Time	Cost	Time
S51D1	51	459.5	**459.5**	24	**459.5**	1	**459.5**	473.2	5	**459.5**	12	468.8	14
S51D2	51	708.4	708.5	28	709.3	11	717.2	732.4	4	724.0	1	718.7	32
S51D3	51	948.0	948.2	31	948.1	16	960.4	1001.2	3	970.7	3	969.8	19
S51D4	51	1561.3	1562.8	53	1562.0	56	1569.9	1708.0	3	1614.1	15	1628.2	20
S51D5	51	1333.7	1333.9	42	**1333.7**	37	1339.4	1404.5	2	1381.7	4	1362.2	15
S51D6	51	2169.1	2186.2	90	**2169.1**	63	2182.1	2230.1	2	2213.9	2	2236.2	14
S76D1	51	598.9	**598.9**	56	**598.9**	5	633.8	610.2	64	629.6	98	613.7	252
S76D2	51	1087.4	1087.5	53	**1087.4**	69	1104.6	1169.8	8	1113.4	9	1128.2	60
S76D3	51	1427.8	1428.2	64	1427.9	97	1435.1	1490.1	12	1460.0	13	1472.9	51
S76D4	51	2079.8	2083.5	90	**2079.8**	188	2106.6	2220.9	7	2103.1	13	2180.1	54
S101D1	51	726.6	727.8	69	**726.6**	16	791.1	765.5	10	791.2	119	749.9	860
S101D2	51	1378.4	1378.6	66	**1378.4**	152	1426.2	1445.0	26	1415.9	24	1409.0	220
S101D3	51	1874.8	1876.6	110	**1874.8**	317	1911.1	1990.3	28	1907.9	21	1947.6	132
S101D5	51	2790.7	2797.5	147	2791.2	572	2824.2	2999.3	18	2896.0	14	2910.7	131
Average	51	1367.5	1369.8	66	1367.6	114	1390.1	1445.7	14	1405.8	25	1414.0	134

Table 7: SDVRP: detailed results on Set 2 instances.

Instance	n	BKS	FTS		SplitILS		BPCH	ICA+VND		VRTR+EMIP		TSVBA		RGTS		ABHC	
			Cost	Time	Cost	Time	Cost	Cost	Time	Cost	Time	Cost	Time	Cost	Time	Cost	Time
SD1	8	228.3	**228.3**	0	**228.3**	0	**228.3**	**228.3**	1	**228.3**	1	**228.3**	5	**228.3**	0	**228.3**	300
SD2	16	708.3	**708.3**	2	**708.3**	1	**708.3**	**708.3**	0	714.4	54	**708.3**	0	**708.3**	0	**708.3**	300
SD3	16	430.4	430.6	2	430.6	1	**430.4**	430.6	0	430.6	67	430.6	0	430.6	0	430.6	300
SD4	24	630.6	631.1	9	631.1	2	**630.6**	635.8	1	631.1	400	631.0	2	634.0	0	631.1	300
SD5	32	1389.9	1390.6	23	1390.6	6	**1389.9**	1390.6	1	1408.1	403	1390.6	1	1401.3	3	1390.6	300
SD6	32	830.9	831.2	20	831.2	6	**830.9**	831.2	1	831.2	408	831.2	4	846.2	2	831.2	300
SD7	40	3640.0	**3640.0**	35	**3640.0**	14	**3640.0**	**3640.0**	1	3714.4	403	**3640.0**	1	**3640.0**	3	**3640.0**	300
SD8	48	5068.3	**5068.3**	47	**5068.3**	25	**5068.3**	**5068.3**	2	5200.0	404	**5068.3**	1	**5068.3**	2	**5068.3**	300
SD9	48	2042.9	2044.2	46	2044.2	39	**2042.9**	2071.0	3	2059.8	404	2071.0	11	2044.7	1	2067.8	300
SD10	64	2683.7	2684.9	65	2684.9	101	**2683.7**	2747.8	4	2749.1	400	2747.3	11	2701.5	6	2784.2	300
SD11	80	13280.0	**13280.0**	192	**13280.0**	152	**13280.0**	**13280.0**	4	13612.1	400	**13280.0**	5	**13280.0**	15	**13280.0**	300
SD12	80	7213.6	7217.2	155	**7213.6**	211	7239.6	7280.0	4	7399.1	408	**7213.6**	18	**7213.6**	19	7220.4	300
SD13	96	10105.9	10110.6	213	10110.6	189	**10105.9**	10110.6	6	10367.1	405	10110.6	9	10129.5	61	10277.8	300
SD14	120	10717.5	10735.6	330	**10717.5**	480	10725.4	10893.5	15	11023.0	5022	10802.9	79	10783.0	41	10790.6	3600
SD15	144	15094.5	15122.2	400	**15094.5**	732	15129.7	15168.3	18	15271.8	5042	15153.5	50	15151.1	110	15152.9	3600
SD16	144	3379.3	3381.3	157	3381.3	931	**3379.3**	3635.3	40	3449.1	5015	3446.4	238	3481.2	54	3381.3	3600
SD17	160	26493.6	26555.6	400	26496.1	577	26533.4	26559.9	17	26665.8	5024	**26493.6**	88	26512.5	130	26536.1	3600
SD18	160	14202.5	14217.5	400	**14202.5**	835	14283.5	14440.6	40	14546.6	5029	14323.0	213	14293.5	61	14469.1	3600
SD19	192	19995.7	20094.6	400	**19995.7**	1525	20374.7	20191.2	28	20559.2	5034	20157.1	190	20131.9	310	20420.1	3600
SD20	240	39635.5	39866.7	400	**39635.5**	1563	40265.3	39813.5	63	40408.2	5053	39722.9	388	39702.0	560	40368.6	3600
SD21	288	11271.1	**11271.0***	400	11345.7	5035	11440.5	11799.6	738	11491.7	5051	11458.8	2533	11365.2	371	11271.1	3600
Average	–	9002.0	9024.3	176	9006.2	592	9067.2	9091.6	47	9179.1	2116	9043.3	183	9035.6	83	9092.8	1557

Table 8: SDVRP: detailed results on Set 3 instances.

Instance	n	BKS	FTS		SplitILS		ABHC	BPCH	RGTS		SPLITABU	OH
			Cost	Time	Cost	Time	Cost	Cost	Cost	Time	Cost	Cost
p01	50	524.6	**524.6**	23	**524.6**	2	**524.6**	–	529.2	12	530.8	527.7
p01_D1	50	459.5	**459.5**	21	**459.5**	1	–	**459.5**	466.9	17	462.9	–
p01_D2	50	756.7	**756.7**	26	757.2	13	776.4	770.2	784.6	4	765.3	758.2
p01_D3	50	1005.8	**1005.8**	27	**1005.8**	21	1012.6	1017.2	1025.0	1	1039.1	1021.0
p01_D4	50	1488.4	**1487.6***	46	1488.6	43	1489.6	1489.4	1503.3	1	1512.0	1497.2
p01_D5	50	1481.7	1481.9	44	**1481.7**	44	1488.3	1499.3	1503.2	8	1503.9	1502.0
p01_D6	50	2155.8	2166.2	79	2156.1	78	2174.5	2166.3	2195.7	4	2173.6	2166.8
p02	75	823.9	**823.9**	45	**823.9**	30	829.9	–	864.6	11	854.3	853.6
p02_D1	75	617.9	618.3	40	**617.9**	6	–	652.9	629.1	19	623.9	–
p02_D2	75	1109.6	1109.7	43	**1109.6**	53	1124.0	1121.8	1146.2	8	1134.1	1122.9
p02_D3	75	1502.1	1503.7	49	**1502.1**	108	1508.7	1514.4	1550.4	13	1556.7	1548.5
p02_D4	75	2298.3	2298.8	98	2298.6	207	2340.1	2318.3	2398.4	17	2338.7	2337.8
p02_D5	75	2219.5	2222.3	94	2220.0	265	2243.9	2237.2	2240.0	12	2293.5	2263.1
p02_D6	75	3223.4	3236.1	185	**3223.4**	379	3266.8	3258.2	3259.4	10	3285.4	3250.4
p03	100	826.1	827.4	56	**826.1**	41	**826.1**	–	846.0	31	841.4	840.1
p03_D1	100	760.0	760.2	58	**760.0**	22	–	788.2	805.0	123	771.5	–
p03_D2	100	1458.5	1458.8	80	**1458.5**	180	1478.6	1477.4	1491.8	29	1515.2	1505.5
p03_D3	100	1996.8	1998.4	99	**1996.8**	362	2035.9	2040.9	2062.5	53	2054.1	2024.6
p03_D4	100	3085.7	3101.2	188	**3085.7**	737	3145.3	3127.1	3171.6	52	3155.2	3136.3
p03_D5	100	2988.4	2994.8	158	2989.3	743	3014.1	3030.7	3091.3	49	3070.9	3055.5
p03_D6	100	4387.3	4431.0	324	**4387.3**	675	4447.5	4467.6	4465.0	84	4470.7	4452.6
p04	150	1023.7	1025.8	116	1023.9	234	1028.4	–	1059.7	488	1070.9	1055.1
p04_D1	150	921.5	923.5	119	921.9	162	–	984.7	979.7	433	947.1	–
p04_D2	150	2016.9	2017.8	161	2017.0	1110	2055.2	2066.5	2093.2	135	2101.8	2093.3
p04_D3	150	2849.7	2852.3	227	**2849.7**	1519	2912.1	2917.8	2943.5	179	2991.6	2977.0
p04_D4	150	4543.2	4573.1	400	4545.5	2410	4638.7	4678.5	4652.1	214	4674.1	4659.9
p04_D5	150	4334.7	4362.2	400	**4334.7**	2358	4436.0	4438.8	4460.2	255	4496.9	4465.5
p04_D6	150	6393.2	6479.5	400	6395.4	1926	6467.2	6523.2	6511.5	198	6482.2	6462.8
p05	199	1283.3	1287.9	198	1289.9	1355	1302.9	–	1368.8	654	1340.4	1338.4
p05_D1	199	1074.2	1078.5	199	**1074.2**	626	–	1268.8	1158.1	1,011	1148.3	–
p05_D2	199	2477.4	2484.4	310	2478.4	2661	2540.1	2596.9	2571.0	313	2585.8	2582.6
p05_D3	199	3471.3	3489.3	400	3471.4	3015	3581.7	3568.2	3592.8	247	3624.2	3594.0
p05_D4	199	5520.7	5578.9	400	5521.6	4350	5669.3	5673.2	5798.4	232	5715.8	5710.2
p05_D5	199	5403.4	5452.1	400	5409.8	4524	5541.1	5560.3	5556.0	201	5571.1	5549.8
p05_D6	199	8188.2	8307.5	400	8192.0	3258	8297.7	8410.4	8319.4	223	8392.1	8355.5
p11	120	1037.9	1046.0	99	**1037.9**	85	1042.1	–	1043.9	1,121	1057.0	1057.0
p11_D1	120	1042.9	1043.7	105	1043.2	110	–	1071.6	1099.3	1,201	1055.3	–
p11_D2	120	2898.5	2899.4	129	**2898.5**	895	2913.1	2983.8	2939.4	104	3060.5	3017.9
p11_D3	120	4218.5	4221.7	216	4219.0	1957	4270.4	4259.9	4301.5	139	4502.6	4476.4
p11_D4	120	6854.0	6876.1	297	6854.1	3442	6890.4	6995.9	6967.5	304	7350.1	7117.2
p11_D5	120	6652.6	6670.9	304	6674.0	2354	6671.0	6822.3	6770.1	178	7168.3	7126.8
p11_D6	120	10132.5	10262.7	400	10204.8	2280	10233.4	10376.9	**10132.5**	39	10673.3	10429.8
Average	–	2799.5	2814.3	178	2800.7	1063	–	–	2865.4	201	2903.9	–

EC.2: Detailed results on the MDSDVRP

In this section, we report the detailed computational results about the MDSDVRP. Table 9 reports details about the programming languages and the machines used by the different algorithms proposed in the literature for the MDSDVRP. Tables 10 and 11 give detailed results about the two set of MDSDVRP instances, respectively. These tables show the following columns: the number of customers ("n"), the number of depots ("m"), and for each algorithm the average cost of the solutions computed over the ten runs ("Cost"), and the corresponding average computing time ("Time") (when available). For our algorithm, the tables also give the number of vehicles used in the solutions ("NV"). The solution values marked with bold numbers are about the best values computed by the algorithms.

Table 9: Details of state-of-the-art algorithms for the MDSDVRP.

Algorithm	Reference	Language	Machine
VES	Gulczynski et al. (2011)	C++	Pentium IV 3.0 GHz + 512 MB RAM
IDH	Gulczynski et al. (2011)	C++	Pentium IV 3.0 GHz + 512 MB RAM
HP	Ray et al. (2014)	–	–

Table 10: MDSDVRP: detailed results on Set 1 instances.

Instance	n	m	FTS			VES	IDH		HP	
			Cost	NV	Time	Cost	Cost	Time	Cost	Time
SQ1	32	2	1056.6	25	185.6	1057.7	1063.1	638.2	**1048**	1.0
SQ2	48	3	1588.2	37	312.2	1588.5	1601.0	634.5	**1588**	1.1
SQ3	64	4	2123.2	49	400.0	2131.4	2142.1	559.6	**2116**	2.6
SQ4	80	5	**2653.9**	62	400.0	2662.2	2684.0	604.3	2665	5.6
SQ5	64	2	**3407.1**	49	400.0	3422.2	3434.7	646.8	3446	8.6
SQ6	96	3	**5117.4**	73	400.0	5135.3	5142.1	653.1	5153	24.4
SQ7	128	4	**6835.0**	98	400.0	6860.4	6969.1	924.8	6929	57.9
SQ8	160	5	8582.8	122	400.0	**8573.5**	8600.6	928.2	8638	101.6
SQ9	96	2	**7035.1**	73	400.0	7050.6	7109.7	696.7	7047	41.1
SQ10	144	3	10599.0	110	400.0	**10577.9**	10586.5	939.8	10587	127.8
SQ11	192	4	14128.2	147	400.0	**14117.2**	14135.8	947.9	14152	302.8
SQ12	240	5	17656.2	184	400.0	**17644.6**	17739.6	940.9	17780	566.8
Average	–	–	6731.9	–	374.8	6735.1	6767.4	759.6	6762.4	103.4

Table 11: MDSDVRP: detailed results on Set 2 instances.

Instance	n	m	Demand interval	FTS			IDH	
				Cost	NV	Time	Cost	Time
MDSD1	50	4	[0.1,0.9]	**995.84**	29	237.8	1018.22	635.0
			[0.3,0.7]	**956.54**	27	260.3	990.85	614.9
			[0.7,0.9]	**1332.06**	44	400.0	1344.99	614.6
MDSD2	75	5	[0.1,0.9]	**1271.68**	43	285.7	1289.06	687.6
			[0.3,0.7]	**1204.76**	38	288.8	1223.57	682.0
			[0.7,0.9]	1712.09	65	400.0	**1705.98**	680.5
MDSD3	100	2	[0.1,0.9]	**2571.20**	48	400.0	2624.41	654.6
			[0.3,0.7]	**2526.27**	50	400.0	2558.33	657.8
			[0.7,0.9]	**3867.67**	81	400.0	3878.34	660.6
MDSD4	100	2	[0.1,0.9]	**2360.43**	53	400.0	2393.23	639.7
			[0.3,0.7]	**2320.90**	50	400.0	2336.65	651.3
			[0.7,0.9]	**3477.85**	82	400.0	3525.24	645.5
MDSD5	100	3	[0.1,0.9]	**1935.72**	52	400.0	1963.13	656.1
			[0.3,0.7]	**1855.37**	50	385.4	1871.47	665.3
			[0.7,0.9]	**2739.25**	82	400.0	2772.58	649.4
MDSD6	100	4	[0.1,0.9]	**1913.47**	54	400.0	1963.68	657.1
			[0.3,0.7]	**1827.27**	52	400.0	1887.48	689.1
			[0.7,0.9]	**2674.72**	85	400.0	2696.47	664.1
MDSD7	249	2	[0.1,0.9]	**15910.21**	122	400.0	16096.91	953.6
			[0.3,0.7]	**16024.68**	124	400.0	16136.07	944.1
			[0.7,0.9]	**25034.38**	204	400.0	25502.49	937.6
MDSD8	249	3	[0.1,0.9]	**13159.39**	122	400.0	13258.26	969.7
			[0.3,0.7]	**13279.96**	125	400.0	13444.18	948.8
			[0.7,0.9]	**20639.76**	204	400.0	20915.02	987.4
MDSD9	249	4	[0.1,0.9]	**11703.12**	123	400.0	11959.27	960.8
			[0.3,0.7]	**12050.49**	125	400.0	12176.61	942.4
			[0.7,0.9]	**18478.00**	203	400.0	18844.77	987.1
MDSD10	249	5	[0.1,0.9]	**11150.65**	124	400.0	11377.3	938.8
			[0.3,0.7]	**11546.80**	125	400.0	11831.52	980.6
			[0.7,0.9]	**17450.08**	203	400.0	17777.76	961.7
Average	–	–	–	7465.69	–	381.9	7578.79	777.2

EC.3: Detailed results on the SDVRPTW

In this section, we report the detailed computational results about the SDVRPTW. Tables 12 and 13 gives detailed results about the two sets of SDVRPTW instances, respectively. The tables show for each algorithm the average cost of the solutions computed over the ten runs ("Cost"), and the corresponding average computing time ("Time") (when available). For all the algorithms, the tables also give the number of vehicles used in the solutions ("NV"). The solution values marked with bold numbers are about the best values computed by the algorithms.

Table 12: SDVRPTW: detailed results on Set 1 instances.

R1								R2							
Instance	VRPTW-EA		FTS			TSH		Instance	VRPTW-EA		FTS			TSH	
	Cost	NV	Cost	NV	Time	Cost	NV		Cost	NV	Cost	NV	Time	Cost	NV
R101	1645.79	19	**1643.9**	20	38	1648.96	20	R201	1252.37	4	**1153.93**	9	52	1236.93	5
R102	1486.12	17	**1475.4**	18	33	1506.08	18	R202	1191.7	3	**1055.67**	7	49	1138.91	4
R103	1292.68	13	**1239.89**	15	36	1276.58	14	R203	939.54	3	**926.47**	5	43	911.87	4
R104	1007.24	9	**993.78**	11	32	1059.42	11	R204	825.52	2	**816.28**	3	50	844.13	4
R105	1377.11	14	**1364.78**	15	39	1448.89	15	R205	994.42	3	**974.43**	5	46	1035.57	4
R106	**1251.98**	12	1252.38	13	45	1282.5	14	R206	906.14	3	**900.48**	4	45	990.42	3
R107	1104.66	10	**1090.57**	12	39	1229.47	11	R207	893.33	2	**816.72**	3	50	909.18	3
R108	**960.88**	9	969.35	10	37	962.33	10	R208	726.75	2	**719.71**	3	44	766.37	3
R109	1194.73	11	**1154**	13	41	1241.21	13	R209	909.16	3	**878.9**	5	52	966.11	3
R110	1118.59	10	**1072.41**	12	40	1212.11	13	R210	939.34	3	**938.2**	5	57	950.99	4
R111	1096.72	10	**1053.5**	12	41	1098.37	12	R211	892.71	2	**775.66**	4	44	833.47	3
R112	982.14	9	**969.87**	10	33	1000.12	11								
Average	1209.89	11.9	1189.99	13.4	38	1247.17	13.5	Average	951.91	2.7	905.13	4.8	48	962.18	3.6

C1								C2							
Instance	VRPTW-EA		FTS			TSH		Instance	VRPTW-EA		FTS			TSH	
	Cost	NV	Cost	NV	Time	Cost	NV		Cost	NV	Cost	NV	Time	Cost	NV
C101	**828.94**	10	**828.94**	10	29	**828.94**	10	C201	**591.56**	3	**591.56**	3	39	**591.56**	3
C102	**828.94**	10	**828.94**	10	34	840.3	10	C202	**591.56**	3	**591.56**	3	43	**591.56**	3
C103	**828.06**	10	**828.06**	10	42	834.56	10	C203	**591.17**	3	600.21	3	40	**591.17**	3
C104	**824.78**	10	829.66	10	38	855.3	10	C204	**590.6**	3	614.95	3	45	612.51	3
C105	**828.94**	10	**828.94**	10	32	**828.94**	10	C205	**588.88**	3	609.36	3	46	**588.88**	3
C106	**828.94**	10	**828.94**	10	29	**828.94**	10	C206	**588.49**	3	**588.49**	3	43	**588.49**	3
C107	**828.94**	10	**828.94**	10	31	**828.94**	10	C207	**588.29**	3	615.31	4	41	**588.29**	3
C108	**828.94**	10	**828.94**	10	34	**828.94**	10	C208	**588.32**	3	**588.32**	3	45	588.35	3
C109	**828.94**	10	**828.94**	10	39	**828.94**	10								
Average	828.38	–	828.92	–	34	833.76	–	Average	589.86	–	599.97	–	43	592.60	–

RC1								RC2							
Instance	VRPTW-EA		FTS			TSH		Instance	VRPTW-EA		FTS			TSH	
	Cost	NV	Cost	NV	Time	Cost	NV		Cost	NV	Cost	NV	Time	Cost	NV
RC101	1696.94	14	**1646.96**	16	33	1743.32	16	RC201	1406.91	4	**1278.22**	8	44	1375.94	5
RC102	1554.75	12	**1537.56**	15	37	1564.88	15	RC202	1367.09	3	**1151.49**	6	48	1283.78	4
RC103	**1261.67**	11	1290	12	33	1303.4	12	RC203	1049.62	3	**974.42**	5	49	1142.61	4
RC104	**1135.48**	10	1156.2	10	36	1203.85	11	RC204	**798.41**	3	802.25	4	47	867.72	3
RC105	1629.44	13	**1558.53**	16	33	1629.84	17	RC205	1297.19	4	**1204.34**	8	54	1345.87	5
RC106	1424.73	11	**1392.02**	13	34	1459.59	13	RC206	1146.32	3	**1060.78**	6	50	1121.4	4
RC107	**1230.48**	11	1233.52	12	32	1282.58	12	RC207	1061.14	3	**1002.28**	6	48	1045.55	4
RC108	1139.82	10	**1138.93**	11	33	1268.03	11	RC208	828.14	3	**787.17**	5	61	987.41	3
Average	1384.16	–	1369.22	–	34	1431.94	–	Average	1119.35	–	1032.62	–	50	1146.29	–

Table 13: SDVRPTW: detailed results on Set 2 instances.

	R1						R2					
	FTS			TSH			FTS			TSH		
(l, u)	Cost	NV	Time	Cost	NV	Gap(%)	Cost	NV	Time	Cost	NV	Gap(%)
(0.01, 0.05)	**1429.33**	18.8	73.3	1471.49	18.3	2.9	**1362.58**	18.1	91.8	1430.62	18.0	5.0
(0.02, 1.00)	**2233.39**	36.1	348.3	2291.46	35.0	2.6	**2206.56**	36.0	392.5	2318.04	35.0	5.1
(0.50, 1.00)	**3789.02**	68.2	400.0	4040.67	67.0	6.6	**3786.63**	68.4	400.0	4059.26	68.0	7.2
(0.70, 1.00)	**4396.01**	81.4	400.0	4581.54	79.0	4.2	**4402.91**	81.7	400.0	4574.17	80.0	3.9
	C1						C2					
	FTS			TSH			FTS			TSH		
(l, u)	Cost	NV	Time	Cost	NV	Gap(%)	Cost	NV	Time	Cost	NV	Gap(%)
(0.01, 0.05)	**1053.54**	12.1	50.8	1182.12	12.2	12.2	**1065.21**	11.3	84.6	1174.29	11.1	10.2
(0.02, 1.00)	**1730.70**	22.9	271.7	2168.57	22.2	25.3	**1740.10**	22.3	351.0	1995.59	22.0	14.7
(0.50, 1.00)	**3836.91**	61.0	400.0	3979.78	61.0	3.7	**4038.31**	61.0	400.0	4268.02	61.0	5.7
(0.70, 1.00)	**4768.78**	77.1	400.0	4962.28	77.0	4.1	**4946.82**	77.4	400.0	5246.12	77.0	6.1
	RC1						RC2					
	FTS			TSH			FTS			TSH		
(l, u)	Cost	NV	Time	Cost	NV	Gap(%)	Cost	NV	Time	Cost	NV	Gap(%)
(0.01, 0.05)	**1895.39**	21.6	74.8	1965.05	20.1	3.7	**1873.90**	21.0	113.2	1946.07	20.0	3.9
(0.02, 1.00)	**3258.18**	42.0	400.0	3339.20	40.0	2.5	**3267.68**	42.0	400.0	3419.85	39.0	4.7
(0.50, 1.00)	**5130.66**	71.1	400.0	5453.10	70.0	6.3	**5134.80**	71.4	400.0	5546.20	71.0	8.0
(0.70, 1.00)	**5894.94**	83.4	400.0	6095.20	81.0	3.4	**5889.84**	83.5	400.0	6155.49	82.0	4.5