Front. Eng. Manag. 2023, 10(3): 483–498
https://doi.org/10.1007/s42524-023-0259-z

RESEARCH ARTICLE

Wenbin ZHU, Zhuoran AO, Roberto BALDACCI, Hu QIN, Zizhen ZHANG

Enhanced solution representations for vehicle routing problems with split deliveries

Abstract In this study, we investigate a forest-based solution representation for split delivery vehicle routing problems (SDVRPs), which have 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 used in heuristic algorithms. Based on the forest structure, we devise three neighborhood search operators. 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 SDVRPs addressed in the literature: The basic SDVRP, the multidepot SDVRP, and the SDVRP with time windows. The experimental results show that the new forest-based solution representation is particularly effective in designing and implementing neighborhood operators, and that our new approach outperforms state-of-the-art algorithms on standard datasets.

Keywords vehicle routing, multidepot, time windows, tabu search, split delivery

Received Apr. 17, 2022; revised Jan. 2, 2023; accepted Jan. 16, 2023

Wenbin ZHU
School of Business Administration, South China University of Technology, Guangzhou 510640, China

Zhuoran AO
Thrust of Intelligent Transportation, The Hong Kong University of Science and Technology (Guangzhou), Guangzhou 511466, China

Roberto BALDACCI
College of Science and Engineering (CSE), Hamad Bin Khalifa University (HBKU), Doha 5825, Qatar

Hu QIN (✉)
School of Management, Huazhong University of Science and Technology, Wuhan 430074, China
E-mail: tigerqin1980@qq.com

Zizhen ZHANG
School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510275, China

1 Introduction

The vehicle routing problem (VRP) aims to design a set of routes, each of which is performed by a vehicle, such that all customer demands are fulfilled, all constraints are respected, and the cost of all routes is minimized.

In this study, we address a VRP variant, called the split delivery vehicle routing problem (SDVRP), in which an unlimited number of homogeneous and capacitated vehicles are dispatched to serve a number of customers, each with a nonnegative demand, and where each customer is allowed to be served by multiple vehicles. The objective of this problem is to construct a set of delivery patterns to satisfy the demands of all customers such that the cost of all routes is minimized. A delivery pattern is defined as a sequence of customers together with a specified delivery quantity for each customer. When the vehicle capacity is less than the demand of a customer, split delivery is unavoidable and the customer has to be served by more than one vehicle. Dror and Trudeau (1989) showed that even when the vehicle capacity is greater than or equal to all customer demands, substantial cost savings can be achieved through split delivery.

In this study, we also consider two relevant extensions of the basic SDVRP, namely, the multidepot SDVRP (MDSDVRP) and the SDVRP with time windows (SDVRPTW). The MDSDVRP generalizes the SDVRP by dispatching vehicles from multiple depots, whereas the SDVRPTW requires that each customer be served within a specified time interval.

In the literature, researchers have studied variants of the basic SDVRP, some of which use additional assumptions to simplify the problem to a certain extent. For example, Jin et al. (2007) studied an SDVRP variant in which the number of available vehicles is fixed to the minimum possible number. Another variant, called k-SDVRP (Archetti et al., 2006), where k denotes the maximum quantity delivered in each route, requires that all

customer demands, vehicle capacity, and delivery quantity must be integers. Our work considers more general assumptions, in which customer demands and vehicle capacity can be real numbers, the number of available vehicles can be greater than the minimum possible number, and vehicle capacity can be less than customer demand. In the following, the term SDVRP is used to refer to the latter problem variant.

SDVRPs are among the most difficult VRPs because additional decision variables must be included to determine the quantity delivered to each customer by each vehicle. Computing optimal solutions for large SDVRP instances (e.g., instances with more than 100 customers) is difficult (Luo et al., 2017; Bianchessi and Irnich, 2019). Therefore, the majority of studies in the literature adopt heuristic methods to obtain near-optimal solutions for SDVRPs. The key to designing an efficient heuristic relies on the method used to represent feasible solutions of the problem (i.e., solution representation). In contrast to previous heuristic approaches using a natural delivery-pattern-based representation and thus utilizing the properties of SDVRPs, our approach combines 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

The contributions of this paper are as follows.

(1) We design a novel way to represent the solution of SDVRPs, which uses a forest structure to detect the feasibility of a solution.

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

(3) We propose a simple tabu search framework for solving three main SDVRP variants: The basic SDVRP, the MDSDVRP, and the SDVRPTW. Computational experiments show that such a standard approach yields excellent results compared with several existing methods.

Moreover, we collect several datasets of SDVRPs and provide detailed benchmark results that can be used as baselines for future research.

The rest of this paper is organized as follows. The next subsection provides an overview of the literature on SDVRPs. Section 2 formally describes the SDVRPs addressed in this study and reviews some well-known properties of this class of problems. Section 3 introduces the solution, representation and three classes of neighborhood operators. Section 4 details the components of our solution approaches. Section 5 presents the computational results, and Section 6 concludes the paper and presents several future research directions.

1.2 Literature review

The SDVRP has been addressed by many articles in the literature, most of which deal with heuristic techniques. Surveys on the SDVRP and related problems can be found in Archetti and Speranza (2008), Toth and Vigo (2014), and Qin et al. (2021).

The SDVRP with the minimum possible number of vehicles was solved using a dynamic programming algorithm (Lee et al., 2006) and iterative two-stage heuristic with several valid inequalities (Jin et al., 2007). Archetti et al. (2011a; 2014) proposed two exact algorithms for the SDVRP. Archetti et al. (2011a) designed a branch-and-price-and-cut (BPC) algorithm for cases with an unlimited number of vehicles and with the minimum possible number of vehicles. Instances with up to 48 customers were consistently solved to optimality, and an instance containing 144 customers was optimally solved. Archetti et al. (2014) described two exact branch-and-cut (BC) algorithms based on two relaxed formulations. The authors reported that one of the two proposed BC algorithms was capable of solving two instances with 75 and 100 customers, respectively, and the majority of the instances with less than or equal to 50 customers. Ozbaygin et al. (2018) presented a new vehicle-indexed flow formulation for the SDVRP and presented computational results including optimal solutions for instances with up to 76 customers. Munari and Savelsbergh (2022) proposed three compact formulations for the SDVRP and developed a BC algorithm that solved 91 instances with up to 80 customers to prove optimal solutions. Desaulniers (2010) described a BPC algorithm to solve the SDVRPTW. Archetti et al. (2011b) improved the BPC algorithm of Desaulniers (2010) by incorporating a tabu search heuristic to heuristically deal with the pricing problem, several valid inequalities, and a separation approach for k-path inequalities. Instances with less than or equal to 100 customers were optimally solved within one hour. Recently, Luo et al. (2017) investigated a BPC algorithm to solve an SDVRPTW with a linear weight-related cost (SDVRPTWL), where the travel cost is charged based on the traveling distance and vehicle weight. The algorithm was tested on both SDVRPTW and SDVRPTWL instances and instances with up to 100 customers for both variants were optimally solved within one hour. The k-SDVRP with time windows and k-SDVRP considering the minimum 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. Recently, Li et al. (2020) proposed a VRP variant in which split delivery was considered, demands were integers, service time was proportional to the quantity of products delivered, and multiple time windows were available, and devised a BPC algorithm to solve their problem.

More research articles in the literature have made

efforts to design heuristic approaches for SDVRPs. Dror and Trudeau (1989) designed a local search algorithm with two problem-specific search operators, namely route addition and k-split interchange. Archetti et al. (2006) designed a tabu search algorithm for the k-SDVRP, which is able to handle the more general SDVRP. Archetti et al. (2008) further designed a math-heuristic based on the tabu search algorithm proposed by Archetti et al. (2006). In this math-heuristic, the tabu search procedure 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 SDVRP based on alternative solution representations. Our work is based on the work of Zhang et al. (2015), which extends in several directions, i.e., enhanced solution representations, improved neighborhood search operators, and extension to SDVRPs other than the basic SDVRP. In Derigs et al. (2010), the authors described four local search operators adapted from the intratour and intertour exchange operators for the classical VRP. The authors integrated these operators into five metaheuristic frameworks: Attribute-based local beam search heuristic, attribute-based hill climber heuristic, record-to-record travel, threshold acceptance, and simulated annealing. Their numerical results show that the attribute-based hill climber heuristic outperforms other heuristics. Heuristic algorithms for the SDVRP with a minimum number of vehicles can be found in Mota et al. (2007), Jin et al. (2008), Aleman et al. (2010), Aleman and Hill (2010) and Archetti et al. (2011a). Berbotto et al. (2014) designed a randomized granular tabu search algorithm for the k-SDVRP with a minimum number of vehicles, whereas Ho and Haugland (2004) designed a tabu search algorithm for the SDVRPTW. Gulczynski et al. (2011) developed a math-heuristic for the MDSDVRP, and Ray et al. (2014) presented an MIP model and used a heuristic search method for the same problem. Han and Chu (2016) studied an SDVRP with minimum delivery amounts (SDVRP-MDA) in which the demand of a customer can be delivered to the customer by multiple vehicles, while each delivery quantity must exceed a given amount. They proposed a new multistart two-phase variable neighborhood descent heuristic to solve the SDVRP-MDA. Chen et al. (2017) provided a simple, effective, and efficient method for the SDVRP using an a priori splitting strategy in which the demand of each customer is split into several small demands. Shi et al. (2018) proposed a particle swarm optimization approach that incorporates a local search to solve the SDVRP. Bortfeldt and Yi (2020) studied an SDVRP with three-dimensional loading constraints. They proposed a hybrid heuristic consisting of a genetic algorithm and a local search, in which several construction heuristics for packing are embedded.

2 The SDVRP, MDSDVRP and SDVRPTW: Problem definitions

The SDVRP is defined on an undirected and complete network $G = (V, E)$, where $V = \{v_0, v_1, ..., v_n\}$ is the vertex set and $E = \{(i, j) : i, j \in V, i \neq j\}$ is the edge set. Node v_0 denotes the depot and $V_c = \{v_1, ..., v_n\}$ denotes the customer set. Each customer $i \in V_c$ has demand d_i and each edge (i, j) has a traveling or routing cost $c_{i,j}$, where the cost matrix $[c_{i,j}]$ satisfies the triangle inequality. We have a sufficient number of homogeneous vehicles, and the vehicle capacity is Q. Figure 1 shows an SDVRP instance with three customers and two vehicles.

A route $r = (v_0, i_1, ..., i_h, v_0)$ is the cycle of network G passing through v_0 and visiting a set of customers $\{i_1, ..., i_h\} \subseteq V_c$ and its cost is computed as the total cost of the edge set in the route. A delivery pattern p is represented by an underlying route $r = (v_0, i_1, ..., i_h, v_0)$ and associated delivery quantity $\delta_{p,i}$ at each visited customer i. A delivery pattern is feasible if its total delivery quantity does not exceed vehicle capacity Q, i.e., $\sum_{i \in \{i_1, ..., i_h\}} \delta_{p,i} \leqslant Q$. The SDVRP consists of designing a set of delivery patterns such that all customer demands are fully satisfied, and the total route costs are minimized.

The MDSDVRP generalizes the SDVRP by replacing vertex set V with set $W \cup V_c$, where $W = \{w_1, ..., w_m\}$ denotes the depot set. In MDSDVRP, a vehicle route starts from any depot of set W and ends at the same depot. An example of MDSDVRP instance can be found in Fig. 2, where two depots, four customers and three vehicles are involved. 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 the service starts from time e_i if the vehicle visits customer i before e_i. An example SDVRPTW instance is provided in Fig. 3.

Note that an SDVRP (also MDSDVRP and SDVRPTW) solution is a set of delivery patterns, while

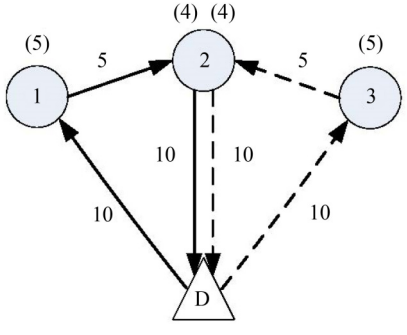

Fig. 1 An example SDVRP instance. Three customers are represented by circles with demands $d_1 = 5$, $d_2 = 8$ and $d_3 = 5$. The traveling distance between two vertices is marked beside the corresponding edge. The number in parentheses is the quantity delivered by the corresponding vehicle.

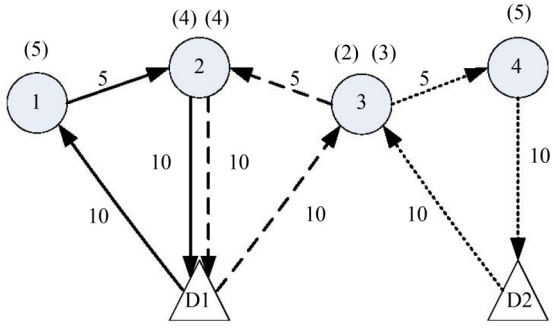

Fig. 2 An example MDSDVRP instance. Four customers are represented by circles with demands $d_1 = 5$, $d_2 = 8$ and $d_3 = d_4 = 5$. The traveling distance between two vertices is marked beside the corresponding edge. The number in parentheses is the quantity delivered by the corresponding vehicle.

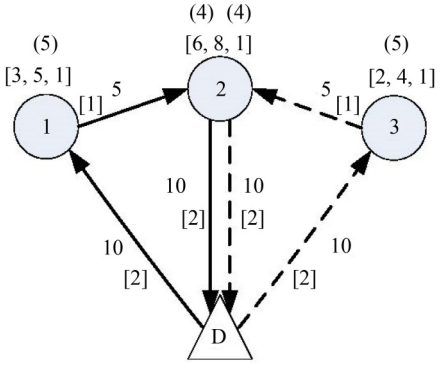

Fig. 3 An example SDVRPTW instance. Three customers are represented by circles with demands $d_1 = 5$, $d_2 = 8$ and $d_3 = 5$. The traveling distance and time (in square brackets) between two vertices are marked beside the corresponding edge. The number in parentheses is the quantity delivered by the corresponding vehicle. The numbers in square brackets $[e, l, s]$ are the ready time, due time and service time of a customer, respectively.

the corresponding solution cost, namely, the total routing cost, is only determined by the underlying routes. The quantity delivered at each customer does not affect the solution value.

Dror and Trudeau (1989) derived the following two properties for the SDVRP.

Property 1. An optimal solution in which no two vehicles have more than one split customer in common must exist if the cost matrix $[c_{i,j}]$ satisfies the triangle inequality.

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

Definition 1. Given k customers $\{i_1, i_2, ..., i_k\}$ and k delivery patterns $\{p_1, p_2, ..., p_k\}$, 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. An optimal solution that does not include a k-split cycle for any $k \geqslant 2$ must exist if the cost matrix $[c_{i,j}]$ satisfies the triangle inequality

Let us use the example shown in Fig. 4 to illustrate Properties 1 and 2. Suppose we have an SDVRP solution with a 2-split cycle shown in Fig. 4(a), where the squares and circles represent the route nodes (r) and customer nodes (c), respectively. Demand is given beside the customer node. For a split customer, the number on a link specifies the quantity of demand served by the vehicle; for a non-split customer, the delivery quantity on the link always equals its demand and is not shown. The number beside the route node provides the current load. The 2-split cycle is $c_2 - r_2 - c_3 - r_1 - c_2$. Then, in Fig. 4(b), we can adjust the quantity of split customers' demand served by the vehicle by setting one of them to zero without violating the feasibility of the solution. If the cost matrix satisfies the triangle inequality, the link with zero quantity is redundant and can be removed (see Fig. 4(c)) to reduce the cost of the current solution or at least not increase it. This implies that there exists an optimal solution that does not include a k-split cycle for any $k \geqslant 2$.

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

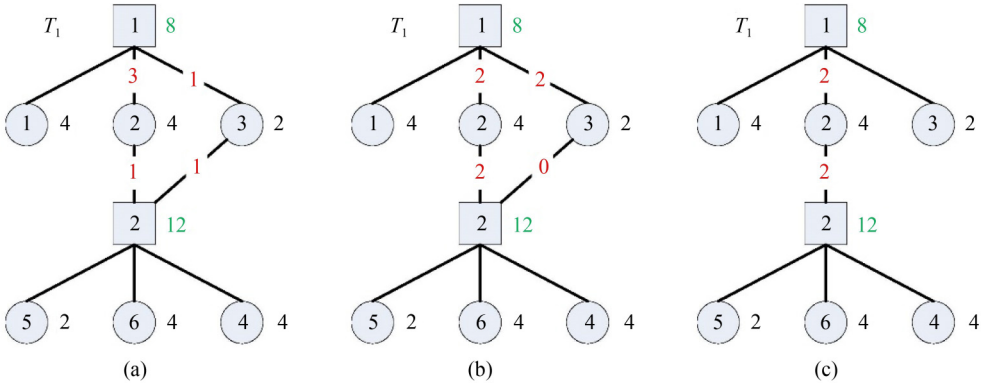

Fig. 4 An example to illustrate Properties 1 and 2.

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 the 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) and Berbotto et al. (2014). These neighborhood operators use a set of delivery patterns to represent a feasible solution, as shown in the example in Fig. 5. 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.

To show the limits of the representation, consider, for example, the operator "2-split interchange" used in Dror and Trudeau (1989) and the operator "relocate" used in Berbotto et al. (2014). From Fig. 5, we can see that moving customer 1 from delivery pattern 1 to delivery pattern 2 in the position between customers 2 and 4 can reduce the total traveling distance if $c_{2,1}+c_{1,4}-c_{2,4}<c_{0,1}+c_{1,2}-c_{0,2}$. However, this results in an infeasible solution, because the vehicle capacity constraint is violated. This relocation can be made possible by reassigning the delivery quantities of customers that have been split. For instance, we can move two units of customer 5's demand from delivery pattern 2 to delivery pattern 4 and then relocate customer 1 to delivery pattern 2. Analogously, by using the 2-split interchange operation, allocating customer 8's demand to delivery patterns 2 and 4 also results in an infeasible solution because of the vehicle capacity constraint. This 2-split interchange operation can be made feasible by first moving one unit of customer 2's demand from delivery pattern 2 to delivery pattern 1.

The above observations suggest that most previously proposed operators can search for a larger solution space if we can find a mechanism that is capable of reallocating split customers' demands automatically and freely among all delivery patterns involved. This can be achieved, for example, by replacing the traditional delivery-pattern-based representation with a forest-based solution representation.

3.1 A forest-based solution representation

A solution S, as defined in Section 2, includes a set of delivery patterns in which the vertex sequence of each route and delivery quantity at each customer are specified. In SDVRPs, changing the delivery quantities may influence the solution's feasibility, but does not impact the solution cost. Therefore, given each route (i.e., vertex sequence) of solution S, if there exists an appropriate delivery quantity at each customer such that all customer demands can be fully satisfied and the vehicle capacity is respected, then S must be feasible, and the corresponding solution cost can be computed as the sum of all route costs. This observation implies that a solution representation in which the delivery quantities are not explicitly maintained may be better for SDVRPs. Therefore, we consider a new solution representation in which 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,

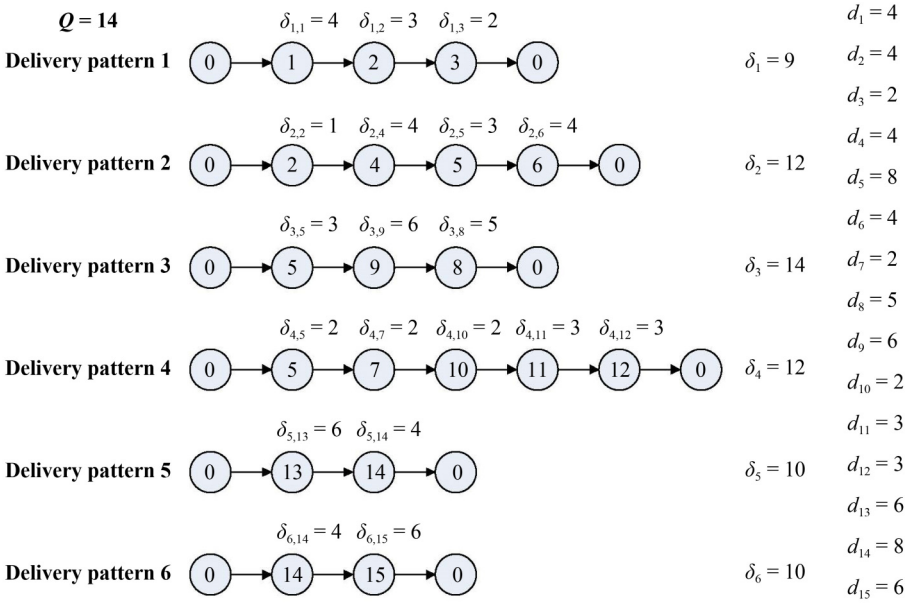

Fig. 5 An example of solution involving six delivery patterns.

the following model defines a feasible region of possible delivery quantities:

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

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

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

Constraint (1a) states that the total delivery quantities in a vehicle route cannot exceed Q. Constraint (1b) ensures that each customer must be fully served. It can be easily seen that the coefficient matrix for the above set of inequalities is totally unimodular. Hence, if Q and d_j are integrals, 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 the maximum flow problem (Ahuja et al., 1993), as shown below.

We first construct a bipartite graph $G(S)=(N(S), E(S))$ based on the information provided by a given solution S. Node set $N(S)$ is composed of two sets of nodes: A route node set $N_r(S)$ and a customer node set $N_c(S)$. Node $i\in N_r(S)$ corresponds to route index $i\in P$ and node $j\in N_c(S)$ corresponds to customer $j\in V_c$. The edge set $E(S)$ includes an edge (i, j) only if route i contains customer j. We then define the corresponding flow network $\overrightarrow{G}(S)=(\overrightarrow{N}(S), \overrightarrow{E}(S))$ from graph $G(S)$ as follows.

(1) The node set $\overrightarrow{N}(S)$ is composed of the sets of nodes $N_r(S)$ and $N_c(S)$ and two dummy nodes, namely, a source node s and a sink node t;

(2) The arc set $\overrightarrow{E}(S)$ is defined as:

a) We create an arc (s, i) with capacity Q from source node s to each $i\in N_r(S)$;

b) We create an arc (j, t) with capacity d_j from each $j\in N_c(S)$ to sink node t;

c) We create an arc (i, j) with a capacity equal to $+\infty$ from route node i to customer node j if $(i, j)\in E(S)$.

Note that the maximum flow in the network is limited by value $\sum_{j\in V_c}d_j$. Figure 6 shows the flow network corresponding to the solution in Fig. 5, where the circles and squares denote the customer and route 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, then 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 of the corresponding arcs (i, j). Although the maximum s-t flow problem can be solved optimally in polynomial time (e.g., in $O(|N(S)||E(S)|^2)$ by using Ford–Fulkerson's algorithm), it is not efficient enough to be used in practice, because once used within a heuristic framework for SDVRPs, it must be executed many times. Below, we investigate the structures of the optimal SDVRP solutions that can help accelerate the problem of checking the feasibility of route-based representation.

We define Ω as the set of all feasible solutions of the SDVRP. We assume that the cost matrix $[c_{i,j}]$ satisfies the triangle inequality. Therefore, 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 $S\in\Omega$, $G(S)=(N(S), E(S))$ is a forest, i.e., $G(S)$ is an acyclic graph.

Proof. We prove this by contradiction by showing that the graph $G(S)$ is acyclic. Suppose that $G(S)$ contains cycle $C=(s_1, s_2, ..., s_k=s_1)$, where $k\geqslant 3$ denotes the cycle length. The value of k must also be odd because $G(S)$ is a bipartite graph. Moreover, cycle C alternates between customer and route nodes. We can assume that $s_1(=s_k)$ is a customer node. Then, C takes the form $(i_1, j_1, i_2, j_2, ..., i_{(k-1)/2}, j_{(k-1)/2}, i_1)$, where the vertices in the odd and even positions correspond to the customer and route nodes, respectively. This result contradicts Property 2.

Proposition 1 also indicates that graph $G(S)$ comprises one or more trees. For convenience, we represent $G(S)$ as $G(S)=\{T_1, T_2, ..., T_g\}$, where T_k $(1\leqslant k\leqslant g)$ is a tree.

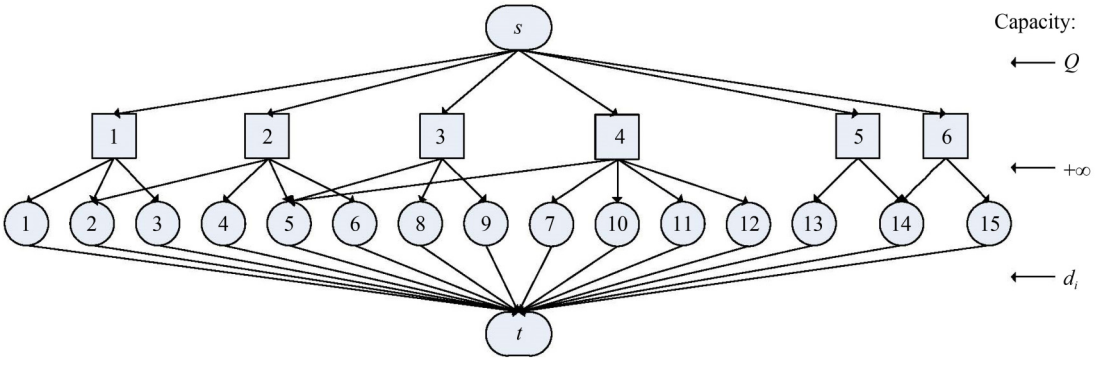

Fig. 6 Flow network for the solution of Fig. 5.

For example, the solution given in Fig. 5, $G(S)$, is composed of two trees, T_1 and T_2, as shown by Fig. 7.

To reduce the computation of checking the feasibility of a route-based representation, below, we propose a greedy procedure (GP) that is able to quickly compute 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 $residual(j) = d_j$, $\forall j \in N_c(S)$. The GP procedure involves the following steps.

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

(2) Select a "nonprocessed" tree T from the forest $G(S)$. If no tree T can be determined, then 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 $T = \varnothing$ (i.e., T has been processed) and go to step 2.

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

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

b) $residual(j) = 0$, set $T = \varnothing$ 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 $j \in N_r(S)$, then we have the following two cases:

a) $residual(i) \leqslant residual(j)$, set $\delta_{i,j} = residual(i)$, $residual(j) = residual(j) - residual(i)$ and $residual(i) = 0$. Then node i is removed from T.

b) $residual(i) > residual(j)$, the procedure terminates without finding a feasible solution.

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

a) $residual(i) \leqslant residual(j)$, set $\delta_{j,i} = residual(i)$, $residual(j) = residual(j) - residual(i)$ and $residual(i) = 0$. Then 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, node j is removed from T.

(8) Go to step 3.

For tree T_1 of the example shown in Fig. 7, the GP procedure works as follows.

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

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

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

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

(5) Customers 1 and 3 are served by vehicle 1, which has a residual capacity of 8.

The following theorem proves the correctness of procedure GP, i.e., 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. First, if the GP 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 GP essentially performs a series of augmenting path steps in the flow network.

Second (by contradiction), assume that the network has a maximum flow value equal to $\sum_{j \in V_c} d_j$ with a flow value $\delta^*_{i,j}$ associated with each arc (i, j) $(i \in N_r(S),\ j \in N_c(S))$

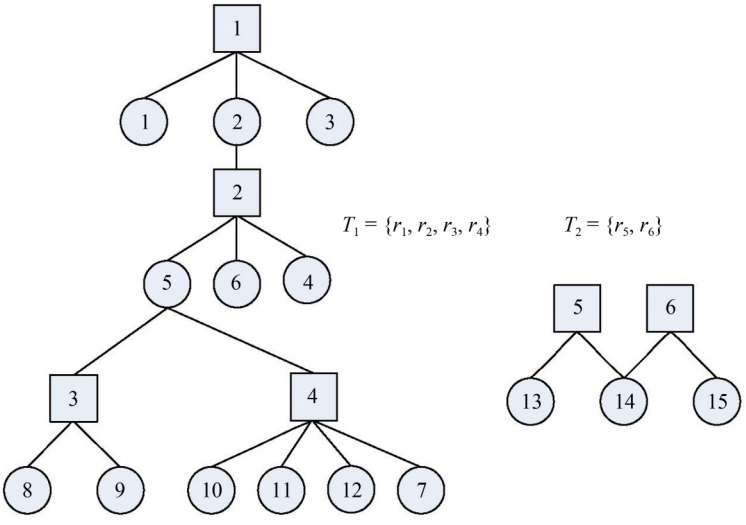

Fig. 7 Graph $G(S)$ corresponding to the solution S shown in Fig. 5.

from a tree associated with the delivery quantity $\delta_{i,j}$. Let the sequence of arcs removed be (i_1, j_1), (i_2, j_2), ..., (i_l, j_l), and assume that the GP terminates immediately after checking node $j \in N_c(S)$ in step 6. We have two cases:

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

(2) There must exist an index $k \leqslant l$ such that $\delta^*_{i_1,j_1} = \delta_{i_1,j_1}$, ..., $\delta^*_{i_{k-1},j_{k-1}} = \delta_{i_{k-1},j_{k-1}}$ 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. Customer node j_k in graph $G(S)$ is connected to a set of route nodes $\{i_k, i', i'', ...\}$, as shown in Fig. 8. We increase $\delta^*_{i_k,j_k}$ to δ_{i_k,j_k} while decreasing $\{\delta^*_{i',j_k}, \delta^*_{i'',j_k}, ...\}$ 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, the solution δ^*_{ij} can be transformed such that $\delta^*_{i_1,j_1} = \delta_{i_1,j_1}$, ..., $\delta^*_{i_l,j_l} = \delta_{i_l,j_l}$, thus obtaining case (1).

Procedure GP can be executed in $O(|N(S)|+|E(S)|)$ time 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 to check the feasibility of a route-based solution. In the following, we describe 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 adapted to handle the MDSDVRP by considering a set of forests for each depot used in the 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 relocate and exchange operators are adapted from their corresponding classic VRP operators, whereas the split operator is based on the k-split interchange operator introduced by Dror and Trudeau (1989). All are specifically tailored to the forest-based solution representation. To speed up the computation, the following attributes of a solution, as illustrated in Fig. 9, are used by different operators.

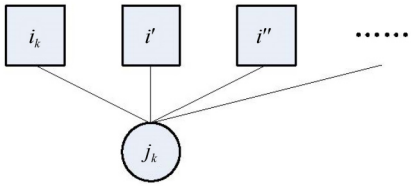

Fig. 8 Example of where a customer node is connected with a set of route nodes.

(1) Maximum residual capacity α_i of 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 the GP procedure.

(2) Maximum supporting capacity $\gamma_{j,i}$ of route i before serving customer j. Suppose that customer j is served by a set of routes $\{i_1, i_2, ..., i_k\}$. The value γ_{j,i_h} $(1 \leqslant h \leqslant k)$ 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.

(3) 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 of the set of routes serving customer j.

Given the index i, α_i can be computed in $O(|N(S)|)$ time by applying GP with i as the 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, the 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 chooses a customer from route r_1 and inserts it into route r_2. Routes r_1 and r_2 are assumed to be 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 tree structures, as described by the two cases below.

(1) $T_1 \neq T_2$, external relocate operation. Consider the example shown in Fig. 10. Suppose that customer j (visited by routes 1 and 3) is moved from route 1 to route 2 and inserted between customers 2 and 3, as shown in Figs. 10(a) and 10(b), respectively. We depict the resulting solution S' and graph $G(S')$, as shown in Figs. 10(c) and 10(d), respectively. The operation is feasible if $\beta_j - \gamma_{j,r_1} + \alpha_{r_2} \geqslant d_j$.

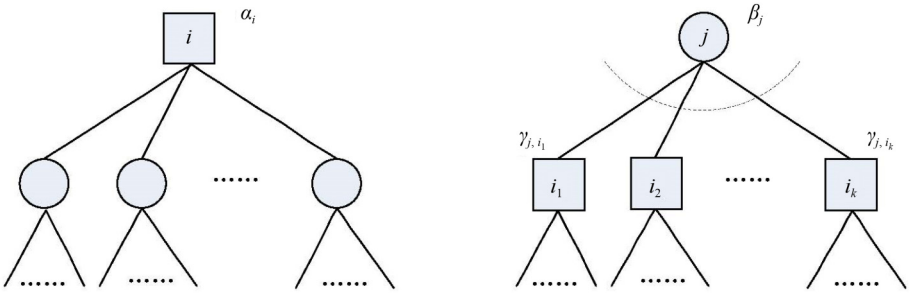

Fig. 9 Computing the maximum residual capacity, the maximum supporting capacity and the maximum available capacity.

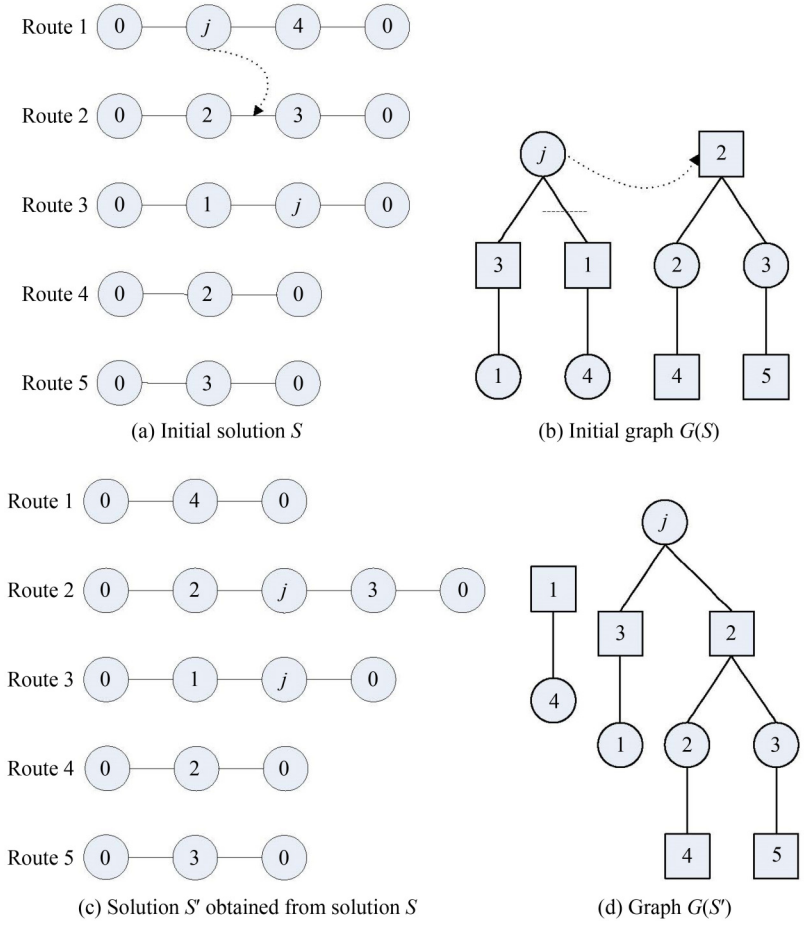

Fig. 10 An example of external relocate operation.

(2) $T_1 = T_2$, internal relocate operation. In this case, the move-producing solution S' can lead to a cycle in the corresponding graph $G(S')$, as shown in Fig. 11. If customer 2 is moved from route 2 to route 4, a cycle (4, 3, 1, 2, 4) arises, and the resulting operation is not admissible due to Property 2. However, as shown in Fig. 11(b), if we move customer 3 from route 1 to route 2, the resulting graph $G(S')$ is acyclic; thus, this operation is potentially admissible. In this case, we apply procedure GP to T_1 to check the feasibility of the operation.

3.2.2 Exchange operator

The exchange operator chooses two customers from two different routes, and exchanges their corresponding positions. Let j_1 and j_2 be the two selected customers, route r_1 (resp., r_2) be the route containing j_1 (resp., j_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.

(1) $T_1 \neq T_2$, external exchange operation. Figure 12 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: $\beta_{j_1} - \gamma_{j_1,r_1} + \gamma_{j_2,r_2} \leqslant d_{j_1}$ and $\beta_{j_2} - \gamma_{j_2,r_2} + \gamma_{j_1,r_1} \leqslant d_{j_2}$.

(2) $T_1 = T_2$, internal exchange operation. In this case, a cycle can occur in the resulting graph $G(S')$ (see Fig. 13(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 introduced by Dror and Trudeau (1989). This operation attempts 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 cost of all new routes is reduced.

Given customer j, the operator first computes 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. Then, the operator calculates the routing cost $CI_j(i)$ for inserting customer j into every route i in the solution. If $i \notin R_j$, the position between customers p and p^+ with the minimum insertion cost is selected, namely,

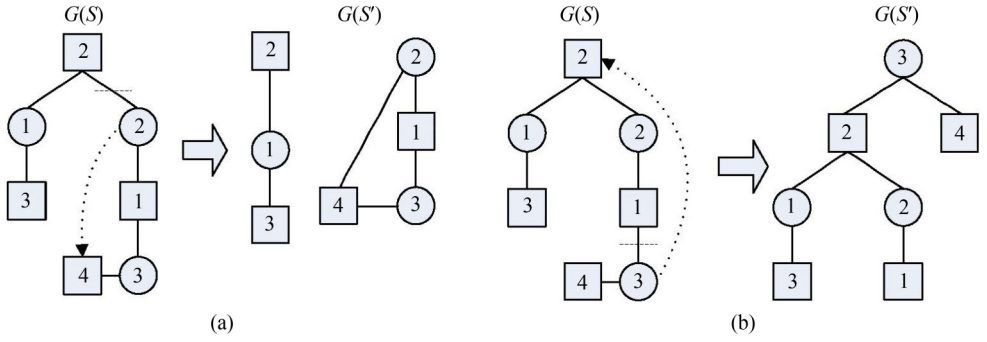

Fig. 11 Examples of internal relocate operation.

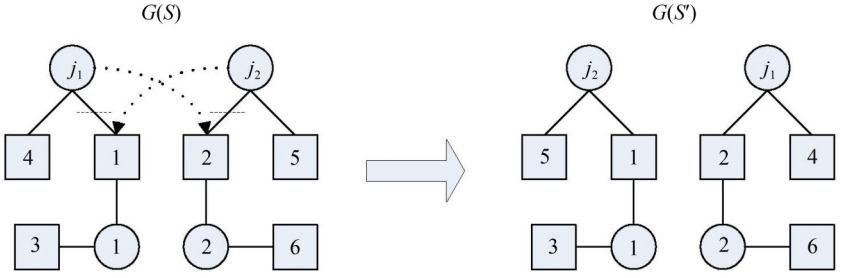

Fig. 12 An example of external exchange operation.

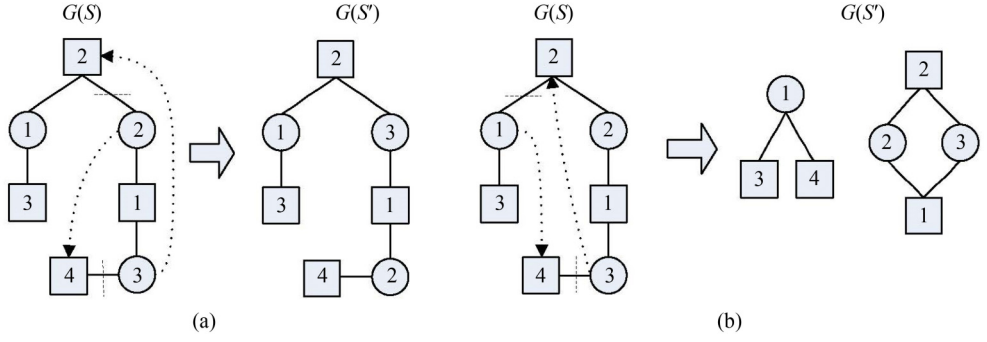

Fig. 13 Examples of internal exchange operation.

$CI_j(i) = c_{p,j} + c_{j,p^+} - c_{p,p^+}$, after examining each position in i. If $i \in R_j$, $CI_j(i)$ is set as $SAV_j(i)$. Finally, customer j is removed from every route $i \in R_j$ and reinserted into the routes in R'_j as described below.

Set R'_j is determined by solving a disjunctively constrained knapsack problem (DCKP) or knapsack problem with conflicts (Yamada et al., 2002), which is defined as follows. Customer j is regarded as a 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 Fig. 14 as an example. Disjunctive constraints require that two conflicting items be not selected simultaneously. 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 Non-deterministic Polynomial (NP)-complete, and solving it to optimality is very time-consuming. To accelerate the computation, we use a greedy heuristic for its solution, which works as follows. The heuristic first sorts all items to decrease the value per unit weight. Then, each item with associated route i is checked sequentially. If assigning customer j to route i does not generate cycles, customer j is inserted into the best position on route i. Otherwise, the next item is checked. When all items have been checked and the demand of customer j is not fully met (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 into route i must also consider the time window constraints.

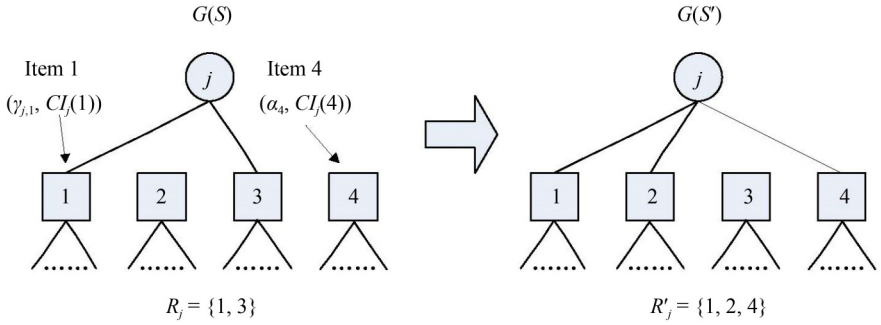

Fig. 14 An example of the split operator.

4 A unified tabu search framework for SDVRPs

We present a unified tabu search framework for SDVRPs that is used to solve the three problems considered in this study: SDVRP, MDSDVRP, and SDVRPTW. The tabu search algorithm (Glover, 1989; 1990) is a famous metaheuristic that has been applied to various VRPs, such as the classic VRP (Gendreau et al., 1994), the SDVRP (Archetti et al., 2006), the 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 our proposed tabu search framework, and their effectiveness is demonstrated by the numerical experiments reported in Section 5. It is worth noting that the neighborhood operators described in this study may also be applied to other metaheuristic frameworks. Algorithm 1 presents a tabu search framework that consists of 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.

Algorithm 1 The tabu search framework

1: $S \leftarrow$ initial solution;
2: **while** no termination criterion is reached **do**
3: $S \leftarrow$ best feasible neighbor S';
4: Update tabu list and aspiration conditions;
5: **if** diversification criterion holds **then**
6: **diversification**;
7: **end if**
8: **end while**

4.1 Neighborhood search and tabu list

Our algorithm employs a neighborhood search procedure to iteratively move from the incremental solution S to one of its neighbors S'. In our implementation, a tabu list management strategy similar to strategies proposed in the literature (e.g., Cordeau et al. (1997)) is included in the neighborhood search procedure to avoid revisiting previously explored solutions.

Algorithm 2 provides the neighborhood search procedure. We denote by $f(S)$ the objective function of 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 search iteration in which customer j is inserted into the i-th route.

$$L = \begin{Bmatrix} L_{1,1} & L_{1,2} & \cdots & L_{1,n} \\ L_{2,1} & L_{2,2} & \cdots & L_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ L_{m,1} & L_{m,2} & \cdots & L_{m,n} \end{Bmatrix}$$

Algorithm 2 Neighborhood search procedure

1: $S_1 \leftarrow$ **relocate** (S, S^*, t, L, T);
2: $S_2 \leftarrow$ **exchange** (S, S^*, t, L, T);
3: $S_3 \leftarrow$ **split** (S, S^*, t, L, T);
4: $S \leftarrow \arg\min_{S' \in \{S_1, S_2, S_3\}} f(S')$;
5: $t \leftarrow t+1$;
6: **Update** L

When inserting customer j into route i, we randomly draw a value in the range $[\xi_l, \xi_u]$ and then assign it to tabu tenure $T_{i,j}$, where ξ_l and ξ_u are user-defined parameters. By this way, we do not set the tabu tenure associated with all insertions to a fixed value, but a random number in a given interval.

$$T = \begin{Bmatrix} T_{1,1} & T_{1,2} & \cdots & T_{1,n} \\ T_{2,1} & T_{2,2} & \cdots & T_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ T_{m,1} & T_{m,2} & \cdots & T_{m,n} \end{Bmatrix}$$

Let t be the index of the current search iteration, $t - L_{i,j} \leqslant T_{i,j}$, removing customer j from route i is not allowed, because it is a tabu operation. However, such an

operation is still admissible if the best solution S^* found thus far can be improved, i.e., an aspiration criterion is used.

4.2 Diversification mechanism

Diversification helps prevent the search process from becoming trapped in the local minima. We invoke a diversification process whenever the best solution currently found cannot be improved for η iterations, where η is a 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 $[k_l, k_u]$. Then, we remove these selected customers from the solution. Finally, we apply the split operator (see Section 3.2.3) to the removed customers in order to reinsert them into the solution. After the solution is perturbed, the tabu list L is cleared and iteration t is set to 0.

4.3 Termination criteria

The algorithms terminate whenever one of the following two conditions is reached: The number of perturbations reaches a predefined parameter ρ or the running time exceeds a prescribed time limit τ.

5 Computational experiments

In this section, we report the computational results of the tabu search algorithm described in Section 4, hereafter called forest-based tabu search (FTS). All algorithms were coded in C++, and all experiments were conducted on a personal computer equipped with an Intel i7-4790 4.00 GHz CPU, 8 GB RAM, and running on the Windows 10 operating system. The computation times reported are in CPU seconds on this machine.

The FTS algorithm was tested on sets of instances already proposed in the literature for SDVRP, MDSD-VRP, and SDVRPTW. For each instance, as generally done in the literature, the FTS algorithm was executed ten times with different random seeds.

5.1 Parameter calibration

We perform preliminary experiments to determine suitable values for the FTS parameters. Table 1 lists the final parameter settings used by FTS. In our preliminary experiments, we set the time limit τ to 400 s, and the remaining parameters are computed as follows.

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

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]
Number of nonimproved steps	η	20000
Number of perturbations	ρ	10
Time limit	τ	400 s

(1) $[\xi_l, \xi_u]$: $[0.01n, 0.05n]$, $[0.05n, 0.1n]$, $[0.1n, 0.2n]$ and $[0.2n, 0.3n]$;

(2) $[k_l, k_u]$: [0, 0], [1, 3], [3, 5], [5, 7], [7, 10] and [10, 15], where interval [0, 0] indicates that no perturbation is used;

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

For the sake of the preliminary experiments, we select 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 with the minimum value.

5.2 Results on the SDVRP

In the literature, the SDVRP has been extensively investigated, and several benchmark instances have been generated by different authors. We conduct experiments on the following three sets of SDVRP benchmark instances.

(1) **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 $[\lceil 0.7Q \rceil, \lfloor 0.9Q \rfloor]$.

(2) **Set 2**. This set contains 21 instances, as proposed by Chen et al. (2007). The number of customers in these instances ranges from 8 to 288. The customer demands are either 60 or 90 units, and the vehicle capacity Q is set to 100 for each instance.

(3) **Set 3**. This set of instances is derived from the capacitated vehicle routing problem (CVRP) instances used by Gendreau et al. (1994). For each CVRP instance, Archetti et al. (2006) varied customer demands according to one of the six demand intervals used in Set 1. We consider six CVRP instances (p01–p05 and p11) for a total of 42 SDVRP instances.

We compare the results generated by the FTS algorithm with those obtained by the following existing algorithms on the above sets of instances.

(1) SPLITABU: Tabu search heuristic of Archetti et al. (2006).

(2) VRTR + EMIP: Variable length record-to-record

travel algorithm with an endpoint MIP algorithm proposed by Chen et al. (2007).

(3) OH: Optimization-based heuristic of Archetti et al. (2008).

(4) ICA + VND: Variable neighborhood descent algorithm of Aleman et al. (2010).

(5) TSVBA: Tabu search with vocabulary building approach of Aleman and Hill (2010).

(6) ABHC: Attribute-based hill climbing heuristic of Derigs et al. (2010).

(7) BPCH: BPC-based heuristic of Archetti et al. (2011a).

(8) RGTS: Randomized granular tabu search by Berbotto et al. (2014).

(9) SplitILS: Iterated local search heuristic of Silva et al. (2015).

(10) SRC + VND: Multistart two-phase variable neighborhood descent heuristic of Han and Chu (2016).

Table S1 of the e-companion to this paper summarizes the programming languages and experimental environments of the above algorithms.

In Table 2, we report the results obtained by the different algorithms for the three sets of SDVRP instances. For each algorithm, the table reports the average of the best solution values computed over 10 runs and the corresponding average running time in seconds. If a value is marked with an "–", the set is not executed by the corresponding algorithm. In the table, the rows labeled "BKS" (best known solution) gives the averages of the best solution values found by the different algorithms proposed in the literature (see Silva et al. (2015)). Tables S2−S4 of the e-companion report the detailed results for the SDVRP.

Table 2 shows that SplitILS and FTS are, on average, the best and the second best algorithms for the SDVRP. The average percentage gaps of the solutions produced by FTS, SplitILS, BPCH, ICA + VND, RGTS and TSVBA with respect to the BKS are equal to 0.17%, 0.01%, 1.65%, 5.72%, 2.80% and 3.40%, respectively, thus showing the high 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 instance of Set 2 instances. In addition, Han and Chu (2016) only solved instances in Set 2 and 11 out of 14 instances in Set 1, and we report the corresponding results in the last row of Table 2. We can observe that SRC + VND obtained worse solutions than FTS for the instances in Set 1 while generating slightly better results for the instances in Set 2.

Table 2 Summary results on the SDVRP

Algorithm	Set 1		Set 2		Set 3	
	Cost	Time (s)	Cost	Time (s)	Cost	Time (s)
BKS	1367.5	–	9002.0	–	2799.5	–
SplitILS	1367.6	114	9006.2	592	2800.7	–
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	1455.7	14	9091.6	47	–	–
ABHC	–	–	9092.8	1557	–	–
VRTR + EMIP	–	–	9179.1	2116	–	–
SPLITABU	–	–	–	–	2903.9	–
SRC + VND	1595.4 (11/14)	21.3	9018.0	75.7	–	–

We implement SplitILS fully following the description in Silva et al. (2015) (this version is called SplitILS2), but unfortunately, we cannot obtain the same results as those reported by Silva et al. (2015). SplitILS uses 14 operators, 3 of which are relocated, exchanged, and split. Next, we replaced these 3 operators with our proposed operators and kept the remaining 11 operators unchanged, obtaining a new version of SplitILS called F-SplitILS. The experimental results show that F-SplitILS performs better than SplitILS2, which demonstrates the effectiveness of the proposed search operators. Although SplitILS is slightly better than FTS, it uses 14 operators, while the latter uses only 3 operators. Moreover, FTS consumes significantly less computation time than SplitILS.

Finally, we studied the impacts of our proposed operators by removing one or two operators and executing the resulting algorithms on benchmark instances. Table 3 presents the average percentage gaps of the solution values produced by the corresponding reduced algorithms with respect to the best known values. A negative gap indicates that the reduced version produces better solutions than the full version, whereas positive gaps are related to worse solutions. From the results shown in this table, we can conclude that these operators have positive effects on average solution quality.

Table 3 Summary results on the impacts of 3 operators

Operator	Gap		
	Set 1	Set 2	Set 3
Relocate + Split	0.39%	0.57%	0.43%
Exchange + Split	0.20%	0.00%	0.10%
Relocate + Exchange	0.11%	−0.01%	0.04%
Relocate	0.42%	0.39%	0.45%
Exchange	0.23%	0.02%	0.22%
Split	1.34%	3.17%	1.41%

5.3 Results on the MDSDVRP

We test the FTS algorithm on the following two sets of

benchmark instances.

(1) **Set 1**. This set is generated by Gulczynski (2010) and contains 12 instances and has a special geographical structure.

(2) **Set 2**. This set contains 30 instances, as proposed by Gulczynski et al. (2011). All instances are generated from 10 multidepot VRP instances by Christofides and Eilon (1969) and Gillett and Johnson (1976). Demand quantity for each customer is randomly generated from three intervals: $[0.1Q, 0.9Q]$, $[0.3Q, 0.7Q]$ and $[0.7Q, 0.9Q]$.

For both sets of instances, the number of depots ranges from 2 to 5.

We compare the results generated by the FTS algorithm with those obtained using the following algorithms proposed in the literature:

(1) VES: Visually estimated solutions algorithm proposed by Gulczynski et al. (2011);

(2) IDH: Interdepot heuristic algorithm proposed by Gulczynski et al. (2011);

(3) HP: Heuristic procedure algorithm proposed by Ray et al. (2014).

Table S5 of the e-companion reports the additional details regarding the algorithms listed above.

Table 4 summarizes the results obtained whereas detailed computational results are presented in Tables S6 and S7 of the e-companion. For each algorithm, Table 4 reports the average of the best solution costs computed over 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 4 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 instances.

Table 4 Summary results on the MDSDVRP

Algorithm	Set 1		Set 2	
	Cost	Time (s)	Cost	Time (s)
FTS	6731.9	375	7465.7	382
VES	6735.1	–	–	–
HP	6762.4	103	–	–
IDH	6767.4	760	7578.8	777

5.4 Results on the SDVRPTW

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

(1) **Set 1**. This set of instances is proposed by Solomon (1987) and contains 56 VRPTW instances.

(2) **Set 2**. This set of instances is proposed by Ho and Haugland (2004) and is derived from VRPTW instances in which the original customer demands are modified as follows: 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 the maximum and minimum customer demands, respectively, and 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), and (0.70, 1.00).

The FTS algorithm is compared with the delivery-pattern-based tabu search algorithm proposed by Ho and Haugland (2004), called the TSH algorithm, which is coded in C++ and executed on a Sun Ultra 10 (UltraSPARC-IIi 440 MHz) workstation. Because 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 5 summarizes the results obtained for the SDVRPTW. We present the detailed computational results in Tables S8 and S9 of the e-companion. For each algorithm, Table 5 reports the average of the best solution costs computed over 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 5 Summary results on the SDVRPTW

		FTS		VRPTW-EA	TSH
		Cost	Time (s)	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

The table shows that the FTS outperforms the average TSH algorithm. Furthermore, the detailed tables reported in the e-companion show that the FTS algorithm is capable of computing improved solutions with respect to existing VRPTW solutions (column VRPTW-EA), which is not the case for TSH.

6 Conclusions

In this study, we design new heuristic algorithms for SDVRPs, including the classic SDVRP, the MDSDVRP and the SDVRPTW. These problems have found several practical applications and are among the most difficult VRPs.

The main difficulty of the SDVRPs lies in determining the delivery quantity for each customer. To overcome this difficulty, we propose a novel method for representing feasible SDVRPs solutions. This method uses a forest-based structure that can quickly check the feasibility of a solution. Additionally, in this study, we devise three new forest-based neighborhood operators and implement a standard tabu search heuristic that uses them.

We test our new algorithm on several classes of SDVRP, MDSDVRP, and SDVRPTW instances and compare the results with those generated by existing algorithms. The algorithm proposed in this study is highly competitive with SDVRP instances and is capable.

Competing Interests The authors declare that they have no competing interests.

Electronic Supplementary Material Supplementary material is available in the online version of this article at https://doi.org/10.1007/s42524-023-0259-z and is accessible for authorized users.

References

Ahuja R K, Magnanti T L, Orlin J B (1993). Network Flows: Theory, Algorithms, and Applications. Englewood Cliffs, NJ: 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: An International Journal, 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 G (2008). The split delivery vehicle routing problem: A survey. In: Golden B, Raghavan S, Wasil E, eds. The Vehicle Routing Problem: Latest Advances and New Challenges. New York, NY: Springer, 103–122

Archetti C, Speranza M G, Hertz A (2006). A tabu search algorithm for the split delivery vehicle routing problem. Transportation Science, 40(1): 64–73

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

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 (2014). A randomized granular tabu search heuristic for the split delivery vehicle routing problem. Annals of Operations Research, 222(1): 153–173

Bianchessi N, Irnich S (2019). Branch-and-cut for the split delivery vehicle routing problem with time windows. Transportation Science, 53(2): 442–462

Bortfeldt A, Yi J (2020). The split delivery vehicle routing problem with three-dimensional loading constraints. European Journal of Operational Research, 282(2): 545–558

Chen P, Golden B, Wang X, Wasil E (2017). A novel approach to solve the split delivery vehicle routing problem. International Transactions in Operational Research, 24(1–2): 27–41

Chen S, Golden B, Wasil E (2007). The split delivery vehicle routing problem: Applications, algorithms, test problems, and computational results. Networks: An International Journal, 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: An International Journal, 30(2): 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(9): 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, 40(10): 1276–1290

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

Glover F (1989). Tabu search—Part I. ORSA Journal on Computing, 1(3): 190–206

Glover F (1990). Tabu search—Part II. ORSA Journal on Computing, 2(1): 4–32

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. Dissertation for the Doctoral Degree. College Park, MD: University of Maryland

Han A F W, Chu Y C (2016). A multi-start heuristic approach for the split-delivery vehicle routing problem with minimum delivery amounts. Transportation Research Part E: Logistics and Transportation Review, 88: 11–31

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

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: Methodological, 40(4): 265–284

Li J, Qin H, Baldacci R, Zhu W (2020). Branch-and-price-and-cut for the synchronized vehicle routing problem with split delivery, proportional service time and multiple time windows. Transportation Research Part E: Logistics and Transportation Review, 140: 101955

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 Á (2007). A new metaheuristic for the vehicle routing problem with split demands. In: Proceedings of 7th European Conference on Evolutionary Computation in Combinatorial Optimization. Valencia: Springer, 121–129

Munari P, Savelsbergh M (2022). Compact formulations for split delivery routing problems. Transportation Science, 56(4): 1022–1043

Ozbaygin G, Karasan O, Yaman H (2018). New exact solution approaches for the split delivery vehicle routing problem. EURO Journal on Computational Optimization, 6(1): 85–115

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, 19(2): 201–232

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

Qin H, Su X, Ren T, Luo Z (2021). A review on the electric vehicle routing problems: Variants and algorithms. Frontiers of Engineering Management, 8(3): 370–389

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

Shi J, Zhang J, Wang K, Fang X (2018). Particle swarm optimization for split delivery vehicle routing problem. Asia-Pacific Journal of Operational Research, 35(2): 1840006

Silva M M, Subramanian A, Ochi L S (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

Toth P, Vigo D (2014). Vehicle Routing: Problems, Methods, and Applications. 2nd ed. Philadelphia, PA: Society for Industrial and Applied Mathematics, 241–271

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): 2864–2870

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 29th AAAI Conference on Artificial Intelligence. Austin, TX: AAAI Press, 3432–3438