An Efficient Forest-Based Tabu Search Algorithm for the Split-Delivery Vehicle Routing Problem

Zizhen Zhang,[1] **Huang He,**[1] **Zhixing Luo,**[2] **Hu Qin,**[3] **Songshan Guo**[1]
[1] Sun Yat-Sen University, China
[2] City University of Hong Kong, Hong Kong S.A.R
[3] Huazhong University of Science and Technology, China
{zhangzizhen@gmail.com}

Abstract

The split-delivery vehicle routing problem (SDVRP) is a natural extension of the classical vehicle routing problem (VRP) that allows the same customer to be served by more than one vehicle. This problem is a very challenging combinatorial optimization problem and has attracted much academic attention. To solve it, most of the literature articles adopted heuristic approaches in which the solution is represented by a set of delivery patterns, and the search operators were derived from the traditional VRP operators. Differently, our approach employs the combination of a set of routes and a forest to represent the solution. Several forest-based operators are accordingly introduced. We integrate the new operators into a simple tabu search framework and then demonstrate the efficiency of our approach by conducting experiments on existing benchmark instances.

1 Introduction

The split-delivery vehicle routing problem (SDVRP) has been studied by a number of researches; see (Archetti and Speranza 2008; 2012) for an overview. The defining characteristic of the SDVRP that distinguishes it from the classical vehicle routing problem (VRP) is that each customer can be served by more than one vehicle. Obviously, when the demand of a customer is lager than the vehicle capacity, it has to be split and the customer has to be visited more than once. As shown by (Dror and Trudeau 1989), when all customer demands are less than or equal to the vehicle capacity, split delivery can also lead to substantial cost savings. Some real-life applications validating the importance of the SDVRP can be found in (Mullaseril, Dror, and Leung 1997; Sierksma and Tijssen 1998; Song, Lee, and Kim 2002).

Many research efforts have been spent in designing heuristics for the SDVRP and its variants. The first heuristic for the SDVRP was the local search algorithm by (Dror and Trudeau 1989), which employs two types of problem-specific local search operators, namely *k-split interchange* and *route addition*. Eighteen years later, (Chen, Golden, and Wasil 2007) developed a heuristic that combines an MIP model and a record-to-record travel algorithm for this problem. (Archetti, Speranza, and Hertz 2006) proposed a tabu search algorithm to solve the k-SDVRP in which the vehicle capacity, all customer demands and the quantity delivered to each customer are integer numbers. After analyzing this tabu search algorithm, we believe that it can also deal with the SDVRP. Further, (Archetti, Speranza, and Savelsbergh 2008) devised an optimization-based heuristic for the k-SDVRP based on the algorithm by (Archetti, Speranza, and Hertz 2006). Most of heuristics were developed for the SDVRP with the minimum possible number of vehicles, such as a scatter search algorithm by (Mota, Campos, and Corberán 2007), two column generation based heuristics by (Jin, Liu, and Eksioglu 2008) and (Archetti, Bianchessi, and Speranza 2011), an adaptive memory algorithm by (Aleman, Zhang, and Hill 2010) and a tabu search algorithm with vocabulary building approach by (Aleman and Hill 2010). (Berbotto, García, and Nogales 2013) proposed a randomized granular tabu search algorithm for the k-SDVRP with the minimum possible number of vehicles. (Ho and Haugland 2004) developed a tabu search algorithm for the SDVRP with time windows.

In this paper, we propose a novel solution approach for the SDVRP, where vehicle capacity and customer demands are not required to be integer numbers, the number of vehicles is not limited to the minimum possible number, and the customer demands may exceed the vehicle capacity. The main contributions are threefold. First, we find a novel way to represent the solutions of the SDVRP, which is the combination of a set of vehicle routes and a forest. Second, based on this solution representation, we propose three classes of neighborhood search operators. Finally, we prove the effectiveness of our solution approach by extensive experiments on 67 benchmark instances.

2 Problem Description and Properties

The SDVRP is defined on a complete and directed graph $G = (V, E)$, where $V = \{0, 1, \dots, n\}$ is the vertex set and $E = \{(i, j) : i, j \in V, i \neq j\}$ is the edge set. Vertex 0 is known as the depot and the set of remaining vertices $V_C = \{1, \dots, n\}$ 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 travel cost $c_{i,j}$, where the cost matrix $[c_{i,j}]$ satisfies the triangle inequality. There is a sufficient number of homogeneous vehicles available each with a capacity Q. We define a *delivery pattern* p as a vertex sequence

r (i.e., a route) with the quantity $\delta_{p,i}$ delivered to each visited customer i. The delivery pattern whose total delivered quantity is less than or equal to the vehicle capacity is feasible. The SDVRP consists of assigning each vehicle a feasible delivery pattern, starting from and ending at vertex 0, such that all customers are served with the minimum total travel cost.

(Dror and Trudeau 1989) have shown the following two properties on the optimal solutions of the SDVRP.

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

Definition 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 contains a k-split cycle if p_1 includes i_1 and i_2, p_2 includes i_2 and i_3, $\ldots$, p_{k-1} includes i_{k-1} and i_k and p_k includes i_k and i_1.

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

The SDVRP solution is a set of delivery patterns while the solution value is only determined by the underlying set of vertex sequences. The solutions with the same underlying set of vertex sequences can be regarded identical since they must have equivalent solution values. As a result, from now on we use a set of vertex sequences (i.e., a set of routes in which the delivery quantity at each visited split customer is not determined) from which we can derive at least one feasible SDVRP solution (in the form of a set of feasible delivery patterns) to represent a feasible SDVRP solution.

Let $\digamma$ be the set of all feasible solutions and $\Omega \subseteq \digamma$ be the set of feasible solutions that contain neither two routes with more than one customer in common nor any k-split cycle. According to Properties 1 and 2, Ω must contain at least one optimal solution. For any solution $S' \in \digamma \backslash \Omega$, there must exist a solution $S \in \Omega$ such that S is at least as good as S'. Thus, we can find optimal SDVRP solutions by only exploring Ω.

Given a solution $S = \{r_1, r_2, \ldots, r_m\}$, where $S \in \Omega$, we can construct an undirected graph $G(S) = (N(S), A(S))$. In the remaining paper, we distinguish between the terms *node* and *vertex*, which are usually considered the same and are used interchangeably in standard graph terminology; we specify that *node* refers to the node in graph $G(S)$, while *vertex* refers to the vertex in graph G. The node set $N(S)$ comprises two types of nodes, namely the route node set $N_r(S)$ and the customer node set $N_c(S)$, where node j_k (respectively, i_h) in $N_r(S)$ (respectively, $N_c(S)$) corresponds to route $r_k \in S$ (respectively, customer $h \in V_C$). The edge set $A(S)$ contains an edge (j_k, i_h) only if customer h is served by route r_k. Because Properties 1 and 2 guarantee that no cycle can exist in $G(S)$, $G(S)$ may consist of one or more connected components, each of which is an unrooted tree, i.e., $G(S)$ should be a *forest*. For the sake of convenience, we represent $G(S) = \{T_1, T_2, \ldots, T_g\}$, where $T_k = \{r_p, r_{p+1}, \ldots, r_q\}$ $(1 \leq k \leq g)$ is one of the unrooted trees. Given a solution S as shown in Figure 1, its corresponding $G(S) = \{T_1, T_2, T_3\}$ is displayed in Figure 2, where the squares and circles represent the route nodes and customer nodes, respectively. It is worth pointing out that $G(S)$ does not contain the information on the visiting order of the customers in each route and the quantity delivered to each customer by each vehicle.

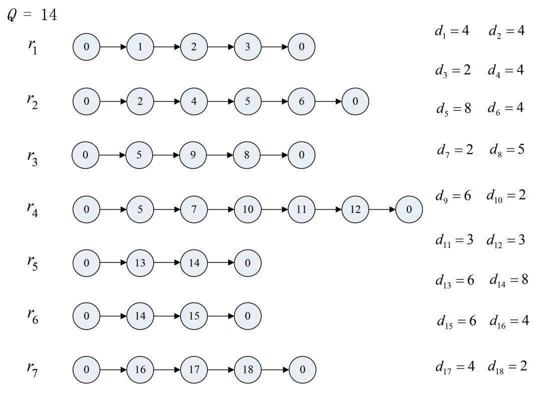

Figure 1: A solution $S = \{r_1, r_2, \ldots, r_7\}$.

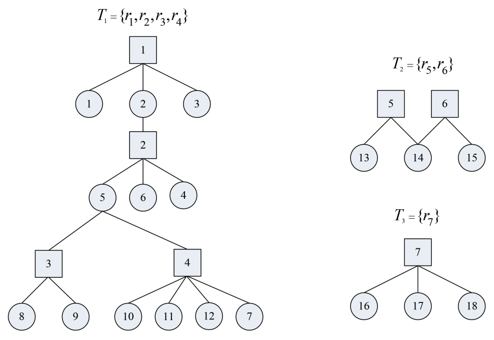

Figure 2: The graph $G(S)$ derived from the solution S shown in Figure 1, where the squares and circles represent the route nodes and customer nodes, respectively.

An unrooted tree T is called feasible if we can find a set of feasible delivery patterns for the routes in T. The solution S is feasible if all unrooted trees in $G(S)$ are feasible. Thus, the forest $G(S)$ can be viewed as an auxiliary data structure to check the feasibility of the solution S. Checking whether an unrooted tree is feasible with respect to the vehicle capacity restriction can be done by a greedy procedure, which is described as follows. First, we arbitrarily select a route node s as the root node, creating a rooted tree. We illustrate the general form of the rooted tree in Figure 3, where the even (respectively, odd) levels consist of only route (respectively, customer) nodes, node $i_{l,k}$ or $j_{l,k}$ represents the k-th node at level l. We denote by $L(x)$ the level of node x, by *residual*(j) the residual capacity of route j, by *residual*(i) the residual demand of customer i, and by $\xi^s_{i,j}$ the quantities of customer i's demand allocated to route j when node s is selected as the root node. We initially set *residual*$(j) = Q$ and *residual*$(i) = d_i$. Next, starting from the nodes at the deepest level of the rooted tree, we recursively compute $\xi^s_{i,j}$ by examining all tree nodes as follows.

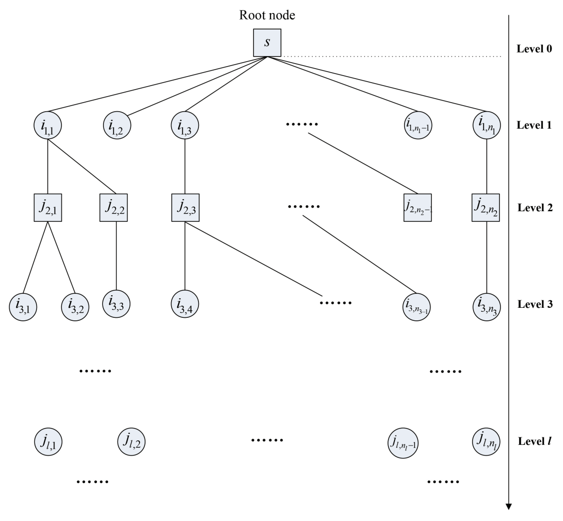

Figure 3: The general form of the rooted tree, in which the root is a route node.

For an edge (i, j), nodes i and j must appear in two consecutive levels. If $L(i) < L(j)$, we have:

$$\xi^s_{i,j} = \min\{residual(i), residual(j)\}, \quad (1)$$
$$residual(i) = residual(i) - \xi^s_{i,j}, \quad (2)$$
$$residual(j) = residual(j) - \xi^s_{i,j}; \quad (3)$$

if $L(i) > L(j)$, we have:

$$\xi^s_{i,j} = residual(i), \quad (4)$$
$$residual(j) = residual(j) - \xi^s_{i,j}, \quad (5)$$
$$residual(i) = residual(i) - \xi^s_{i,j}. \quad (6)$$

The tree nodes at the same level can be selected in any order and the customer nodes with *residual*$(i) = 0$ will not be considered any more. The unrooted tree is infeasible if a route node j with negative $residual(j)$ is encountered; otherwise, the tree under check is feasible. Now we present the detailed steps of checking the feasibility of the tree T_1 shown in Figure 2 using this greedy procedure as follows. (1) Vehicle 4 fully serves customers 10, 11, 12, 7 and its residual capacity is 4. Vehicle 3 fully serves customers 8, 9 and its residual capacity is 3. (2) 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 zero, and the residual demand of customer 5 is 1. (3) Vehicle 2 fully serves customers 2, 4, 6, and delivers 1 unit to customer 5. Then, its residual capacity is 1. (4) Vehicle 1 fully serves customers 1, 3 and its residual capacity is 8. Since the residual capacities of all vehicles are greater or equal to zero, we can claim that T_1 is feasible.

3 Neighborhood Operators and Tabu Search Algorithm

Neighborhood operators are crucial components that determine the performance of heuristics. Some neighborhood operators have been used for the SDVRP or its variants by several previous articles, including (Dror and Trudeau 1989;

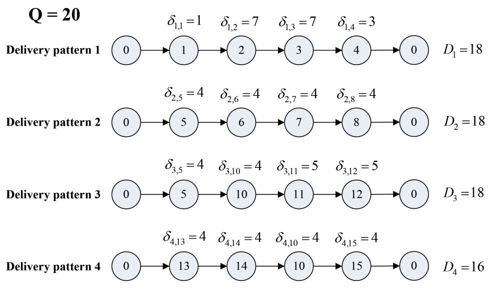
Q = 20
Delivery pattern 1
$\delta_{1,1}=1$ $\delta_{1,2}=7$ $\delta_{1,3}=7$ $\delta_{1,4}=3$
0 1 2 3 4 0 $D_1=18$
Delivery pattern 2
$\delta_{2,5}=4$ $\delta_{2,6}=4$ $\delta_{2,7}=4$ $\delta_{2,8}=4$
0 5 6 7 8 0 $D_2=18$
Delivery pattern 3
$\delta_{3,5}=4$ $\delta_{3,10}=4$ $\delta_{3,11}=5$ $\delta_{3,12}=5$
0 5 10 11 12 0 $D_3=18$
Delivery pattern 4
$\delta_{4,13}=4$ $\delta_{4,14}=4$ $\delta_{4,10}=4$ $\delta_{4,15}=4$
0 13 14 10 15 0 $D_4=16$

Figure 4: An example solution represented by a set of four delivery patterns.

Ho and Haugland 2004; Derigs, Li, and Vogel 2010; Aleman, Zhang, and Hill 2010; Berbotto, García, and Nogales 2013). These operators can be divided into two classes. The first class includes operators that can be applied to the classical VRP while the second one contains the operators that were specifically devised for the characteristic, namely split delivery, of the SDVRP. All these operators are based on representing the feasible solution by a set of delivery patterns. An example of such solution is shown in Figure 4, where D_i is the total quantity delivered by delivery pattern i.

To show the limitation of the existing operators, let us consider two representative operators *relocate* and *2-split interchange* used by (Berbotto, García, and Nogales 2013) and (Dror and Trudeau 1989). In Figure 4, if $c_{1,2} + c_{2,3} - c_{1,3} > c_{5,2} + c_{2,10} - c_{5,10}$, then relocating customer 2 from delivery pattern 1 to delivery pattern 3 at the position between customers 5 and 10 can reduce the total travel distance. However, this is an illegal operation since the capacity restriction will be violated. We can make this relocation possible by re-allocating the delivered quantities of the split customers. For example, we can move from delivery pattern 3 two units of customer 5's demand to delivery pattern 2 and three units of customer 10's demand to delivery pattern 4, and then relocate customer 2 to delivery pattern 3. Analogously, allocating customer 3's demand into delivery patterns 2 and 4 using 2-split interchange operation also results in an infeasible solution due to the violation of the vehicle capacity. We can make this 2-split interchange operation legal by moving from delivery pattern 4 one unit of customer 10's demand to delivery pattern 3.

The above observations tell us that if we have a mechanism to re-allocate the split customers' demands automatically and freely among all involved delivery patterns, the relocate and 2-split interchange operators would be capable of exploring larger solution regions. As shown in Figure 5, we can liken the customer demands to *water* and connect the split customers in different delivery patterns by *tubes* via which water can freely flow. Consequently, we do not need to explicitly fix the delivered quantities at the split customers any more. This finding motivates us to represent the solutions by a set of routes and create forest $G(S)$ to link together the routes with customers in common.

The neighborhood operators proposed in this paper are based on the route-based solution S and forest $G(S)$. The

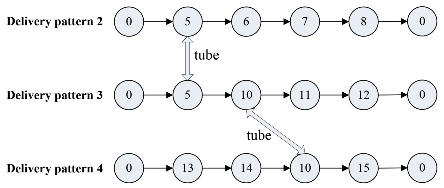

Figure 5: Connecting the split customers in different routes by tubes.

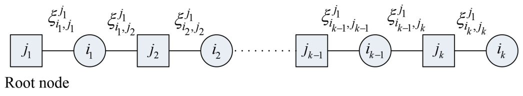

Figure 6: The path from root node j_1 to customer node i_k on graph $G(S)$.

routes preserve the visiting order of the customers, and $G(S)$ is used to quickly check the feasibility of the resulting solution after a certain operation. For the purpose of quickly checking the legality of operations, we introduce the following concepts:

(1) The *maximum residual capacity* mr_k of route k. Selecting route node k as the root node, mr_k is the residual of vehicle k, which can be obtained using expressions (1) – (6).

(2) The *maximum available capacity* $\gamma_{i,k}$ of route k before serving customer i. Suppose customer i is served by a set of routes $\{j_1, j_2, \ldots, j_k\}$. When customer i is selected as the root node, we can obtain γ_{i,j_h} $(1 \le h \le k)$ using expressions (1) – (6). Moreover, we denote the sum of all γ_{i,j_h} by γ_i, which is the total maximum available capacity from the set of routes.

(3) The *capacity saving* $\beta_{j_1,j_k}(x)$. Let $\mathcal{P}_{j_1,i_k} = \{j_1, i_1, j_2, i_2, \ldots, j_k, i_k\}$ be the path from root node j_1 to customer node i_k on $G(S)$ (see Figure 6). After decreasing $\xi^{j_1}_{i_k,j_k}$ by x, the value of $\xi^{j_1}_{i_h,j_h}$ $(1 \le h < k)$ would decrease by x if $\min\{\xi^{j_1}_{i_h,j_h}, \xi^{j_1}_{i_{h+1},j_{h+1}}, \ldots, \xi^{j_1}_{i_{k-1},j_{k-1}}\} \ge x$; otherwise it would decrease by $\min\{\xi^{j_1}_{i_h,j_h}, \xi^{j_1}_{i_{h+1},j_{h+1}}, \ldots, \xi^{j_1}_{i_{k-1},j_{k-1}}\}$. The value of mr_{j_1} would increase by $\beta_{j_1,j_k}(x)$, where

$$\beta_{j_1,j_k}(x) = \min\left\{\xi^{j_1}_{i_1,j_1}, \xi^{j_1}_{i_2,j_2}, \ldots, \xi^{j_1}_{i_{k-1},j_{k-1}}, x\right\}. \quad (7)$$

(4) The *cycle-detection route set* $R_{i,k}$. The set $R_{i,k}$ contains all routes that are in the same tree as route k after route node k and customer node i are disconnected. For example, for T_1 shown in Figure 2, we have $R_{1,1} = \{1, 2, 3, 4\}$, $R_{5,2} = \{2, 1\}$, $R_{5,3} = \{3\}$ and $R_{10,4} = \{1, 2, 3, 4\}$.

We will introduce three new neighborhood operators, namely *relocate, exchange* and *split*. These operators are based on a set of routes and the legality of all operations is checked with the aid of forest $G(S)$. With the values of mr_k, $\gamma_{i,k}$, $\xi^s_{i,j}$ and $R_{i,k}$, we can quickly check whether a certain operation is legal. That is, we do not need to check the legality by traversing the resultant tree of a certain operation. In addition, we denote by V_{NSC} the set of customers

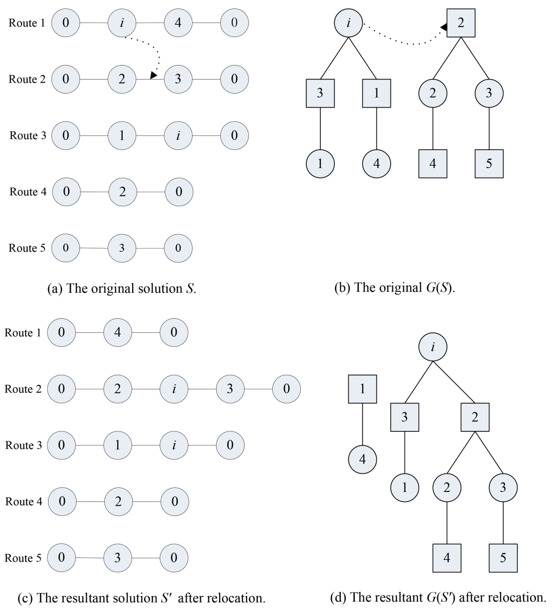
Route 1
Route 2
Route 3
Route 4
Route 5
(a) The original solution S.
(b) The original G(S).
(c) The resultant solution S′ after relocation.
(d) The resultant G(S′) after relocation.

Figure 7: A relocate operation with $i \in V_{SC}$ and $T_1 \neq T_2$.

that have not been split in the current solution. Accordingly, $V_{SC} = V_C \backslash V_{NSC}$ includes those customers that are served by more than one vehicle.

3.1 The Relocate Operator

The *relocate* operator selects a customer i from route 1 and relocates it to a position of route 2. We assume that route 1 (respectively, route 2) is contained in T_1 (respectively, T_2). Since two routes (i.e., routes 1 and 2) may be included in the same or different trees and there are two types of customers (i.e., the split customers in V_{SC} and the non-split customers in V_{NSC}), we can divide all relocate operations into the following four types.

Type 1 *relocate*: $i \in V_{SC}$ and $T_1 \neq T_2$ (see Figure 7). After relocating customer i from route 1 to route 2 at the position between customers 2 and 3, the resultant solution S' and $G(S')$ are shown in Figures 7 (c) and (d), respectively. We can use Figure 7(a) and (c) to calculate the distance saving, and use Figure 7(d) to check the legality of this operation. The relocate operation shown in Figure 7 is legal if $\gamma_i - \gamma_{i,1} + mr_2 \ge d_i$.

Type 2 *relocate*: $i \in V_{NSC}$ and $T_1 \neq T_2$. Relocating customer i from route 1 to route 2 is legal if $d_i \le mr_2$.

Type 3 *relocate*: $i \in V_{NSC}$ and $T_1 = T_2$. Removing customer i from route 1 to route 2 increases mr_2 by $\beta_{2,1}(d_i)$, where $\beta_{2,1}(d_i)$ is defined in expression (7). This relocate operation is legal if mr_2 + $\beta_{2,1}(d_i) \ge d_i$.

Type 4 *relocate*: $i \in V_{SC}$ and $T_1 = T_2$. If route 2 belongs to $R_{i,1}$, the resultant $G(S')$ does not contain a cycle. This relocate operation is legal if no cycle appears in $G(S')$ and $\gamma_i - \gamma_{i,1} + mr_2 + \beta_{2,1}(\xi^2_{i,1}) \ge d_i$.

3.2 The Exchange Operator

The *exchange* operator selects two customers and swaps their positions. Let i_1 and i_2 denote the selected two customers, routes 1 and 2 be the routes containing i_1 and i_2, and T_1 (respectively, T_2) be the tree containing route 1 (respectively, route 2). We can classify the exchange operations into the folowing six types.

Type 1 *exchange*: $i_1, i_2 \in V_{NSC}$ and $T_1 \neq T_2$. This exchange operation is legal if $mr_1+d_{i_1} \geq d_{i_2}$ and $mr_2+d_{i_2} \geq d_{i_1}$.

Type 2 *exchange*: $i_1, i_2 \in V_{NSC}$ and $T_1 = T_2$. Without loss of generality, we can assume $d_{i_2} \geq d_{i_1}$. Replacing customer i_2 with customer i_1 is equivalent to decreasing $\xi^1_{i_2,2}$ by $d_{i_2} - d_{i_1}$. As a result, this exchange operation is legal if $mr_1 + \beta_{1,2}(d_{i_2} - d_{i_1}) + d_{i_1} \geq d_{i_2}$.

Type 3 *exchange*: $i_1 \in V_{SC}, i_2 \in V_{NSC}$ and $T_1 \neq T_2$. Removing customer i_1 from route 1 is equivalent to increasing mr_1 by $\xi^1_{i_1,1}$. The resultant T_1 is feasible if $mr_1 + \xi^1_{i_1,1} \geq d_{i_2}$. Removing customer i_2 from route 2 is equivalent to increasing mr_2 by d_{i_2}. The resultant T_2 is feasible if $\gamma_i - \gamma_{i,1} + mr_2 + d_{i_2} \geq d_{i_1}$.

Type 4 *exchange*: $i_1 \in V_{SC}, i_2 \in V_{NSC}$ and $T_1 = T_2$. Obviously, if $R_{i_1,1}$ does not contain route 2, the resultant $G(S')$ must have a cycle. We further consider the following two situations: (1) $d_{i_2} \leq \xi^2_{i_1,1}$ and (2) $d_{i_2} > \xi^2_{i_1,1}$. If $d_{i_2} \leq \xi^2_{i_1,1}$, replacing customer i_1 with customer i_2 is equivalent to decreasing $\xi^2_{i_1,1}$ by $\xi^2_{i_1,1} - d_{i_2}$. So mr_2 would increase by $\beta_{2,1}(\xi^2_{i_1,1} - d_{i_2})$. If the new set of routes serving customer i_1 can provide sufficient capacity, namely $\gamma_{i_1} - \gamma_{i_1,1} + mr_2 + \beta_{2,1}(\xi^2_{i_1,1} - d_{i_2}) \geq d_{i_1}$, the resulting $G(S')$ is feasible. If $d_{i_2} > \xi^2_{i_1,1}$, then customer i_1 can be fully served by the new set of routes. Since replacing customer i_1 with customer i_2 is equivalent to increasing $\xi^2_{i_1,1}$ by $d_{i_2} - \xi^2_{i_1,1}$, which may cause the path from route 2 to route 1 infeasible. Let us still consider the path shown in Figure 6 and define $\Delta^{j_1}_{j_1,j_h}$ $(h = 1, 2, \ldots, k+1)$ as the maximum increment of $\xi^{j_1}_{i_h,j_h}$ that still keeps the path (j_1, i_1, j_2, i_2, …, j_h, i_h) feasible. The values of $\Delta^{j_1}_{j_1,j_h}$ can be computed by the following recursive functions (8) – (9).

$$\Delta^{j_1}_{j_1,j_1} = \theta^{j_1}_{j_1} = mr_{j_1} \tag{8}$$

$$\Delta^{j_1}_{j_1,j_{h+1}} = \min\left\{\theta^{j_1}_{j_{h+1}}, \Delta^{j_1}_{j_1,j_h} + \max\left\{0, \theta^{j_1}_{j_{h+1}} - \max\left\{0, d_{i_h} - (\gamma_{i_h} - \gamma_{i_h,j_h} - \gamma_{i_h,j_{h+1}})\right\}\right\}\right\}, \tag{9}$$

where $\theta^{j_1}_{j_{h+1}}$ is the residual capacity of route j_{h+1} before serving its father customer node when node j_1 is selected as the root node. Thus, in this situation, the resultant graph is feasible if $\Delta^2_{2,1} \geq d_{i_2} - \xi^2_{i_1,1}$.

Type 5 *exchange*: $i_1, i_2 \in V_{SC}$ and $T_1 \neq T_2$. This exchange operation is legal if $\gamma_{i_1} - \gamma_{i_1,1} + \xi^2_{i_2,2} \geq d_{i_1}$ and $\gamma_{i_2} - \gamma_{i_2,2} + \xi^1_{i_1,1} \geq d_{i_2}$.

Type 6 *exchange*: $i_1, i_2 \in V_{SC}$ and $T_1 = T_2$. This exchange operation replaces two edges in $G(S)$ with two new ones. Since this operation modifies $G(S)$ considerably, it is very time-consuming to check its legality. To save computation time, in our solution approach we do not consider the exchange operations of this type.

3.3 The Split Operator

The *split* operation is a tailored operation for the SDVRP, which replaces the set R_i of routes that serve customer i with a new set R'_i of routes. For each customer $i \in V_C$, it first computes the distance saving $SAV^{r_k}_i$ of removing i from route r_k in R_i by $SAV^{r_k}_i = c_{i^-,i} + c_{i,i^+} - c_{i^-,i^+}$, where i^- and i^+ are the immediate predecessor and successor of customer i. Next, it computes the cost $CI^{r_k}_i$ of inserting customer i into route r_k. If $r_k \notin R_i$, each position in r is examined and the position between customer p and p^+ with the minimal insertion cost is selected, namely $CI^{r_k}_i = c_{p,i} + c_{i,p^+} - c_{p,p^+}$. If $r_k \in R_i$, we set $CI^{r_k}_i = c_{i^-,i} + c_{i,i^+} - c_{i^-,i^+}$. Let U be the set of all subsets of the routes in a solution S and thus R'_i is an element of U. Since the size of U is usually very large, we only consider a subset U' of U.

We sort the routes in S based on the increasing value of $CI^{r_k}_i/mr_{r_k}$ if $r_k \in R_i$ or $CI^{r_k}_i/\gamma_{i,r_k}$ if $r_k \notin R_i$, and then select the first ten routes to construct a route set $\bar{S}$. If the number of routes in S is less than ten, then we set $\bar{S} = S$. For each subset R' of $\bar{S}$, we check the legality of inserting customer i into each of the routes in R'. First, each pair of routes in R' cannot appear in the same connected component after deleting from $G(S)$ edge (i, r_k) for all $r_k \in R_i$. Two routes in the same connected component implies that this insertion will generate a cycle in the resultant graph. Second, the routes in R' need to have enough capacity to serve customer i, i.e., $\sum_{r_k \in R'}(\gamma_{i,r_k}\mathbf{1}_{r_k \in R_i} + mr_{r_k}\mathbf{1}_{r_k \notin R_i}) \geq d_i$, where $\mathbf{1}_x$ is the indicator function. Inserting customer i into each of routes in R' generates a candidate solution, where the total distance saving can be obtained by $\sum_{r_k \in R_i} SAV^{r_k}_i - \sum_{r_k \in R'} CI^{r_k}_i$. The split operator selects the best candidate solution.

3.4 Tabu Search

Tabu search (TS) algorithm (Glover and Laguna 1998) is a well-known meta-heuristic that has been successfully applied to a wide variety of routing and scheduling problems, e.g., (Aleman and Hill 2010; Cai et al. 2013; Ho and Haugland 2004). To highlight the effectiveness of our proposed neighborhood operators, we integrate them into a simple and standard TS framework without a diversification phrase. In each iteration, TS always chooses the best allowable neighbor, i.e., the non-tabu neighbor or the tabu neighbor that satisfies the aspiration criterion. The aspiration criterion is that TS always accepts the tabu solution which is better than the best known solution so far. Note that the parameters related to the TS algorithm include only tabu tenure τ and the number of non-improvement steps η for algorithm termination.

4 Computational Experiments

We conducted experiments on two sets of benchmark SDVRP instances to evaluate the performance of our tabu search algorithm, which are:

- **Set 1**: This set of 42 instances was derived by (Archetti, Speranza, and Hertz 2006) from the capacitated VRP (CVRP) benchmark instances in (Gendreau, Hertz, and Laporte 1994) by varying the customer demands and keeping other characteristics unchanged.
- **Set 2**: This set contains 25 instances and was generated by (Belenguer, Martinez, and Mota 2000).

Our algorithm was coded in C++ and all experiments were conducted on a desktop with an Intel i5-2410 2.30 GHz CPU, 4 GB RAM and Windows 8 Pro 64-bit operating system. After conducting some preliminary experiments, the parameters in our solution approach were fixed as follows: $\tau = 0.4n$ and $\eta = 3000$.

We compare our solution approach with the following existing algorithms: (1) SPLITABU: the tabu search algorithm by (Archetti, Speranza, and Hertz 2006); (2) OH: the optimization-based heuristic by (Archetti, Speranza, and Savelsbergh 2008); (3) BPCH: the branch-and-price-and-cut based heuristic by (Archetti, Bianchessi, and Speranza 2011); (4) RGTS: the randomized granular tabu search by (Berbotto, García, and Nogales 2013); (5) ICA+VND: the variable neighborhood descent algorithm by (Aleman, Zhang, and Hill 2010), where the iterated constructive algorithm is employed to create the initial solution; (6) TSVBA: the tabu search with vocabulary building approach by (Aleman and Hill 2010). Because these algorithms were carried out on different machines, and their results did not contain the running time information in a unified format, we decide not to report their running times. In fact, all these algorithms executed within reasonable time limits.

The instances in Set 1 have been solved by SPLITABU, OH, BPCH and RGTS. We compare our results with those of these four algorithms in Table 1, where the columns "Cost" show the solution costs and the columns "Time" give the running time. The columns "Gap (%)" display the percentage gap between the solution costs of our solution and each benchmark algorithm; for example, the values of "Gap (%)" under the block "SPLITABU" is calculated by: (SPLITABU − Our approach)/Our approach. Moreover, (Archetti, Speranza, and Savelsbergh 2008) did not consider instances *px-110* and (Archetti, Bianchessi, and Speranza 2011) did not solve instances *px-00*. So we fill the corresponding cells with dashes ('–'). (Berbotto, García, and Nogales 2013) applied the RGTS several times with different parameter settings to solve the instances with the minimum possible number of vehicles, the corresponding best results are given in the block "RGTS". The rest of blocks report the results of the instances without any limitation on the vehicle number. The smallest solution cost in each row is marked in bold. The last two rows of this table give the numbers of best solutions found and the average values of "Gap (%)", respectively.

Table 1: Performance comparison on the instances in Set 1.

Instance	Our approach		SPLITABU		OH		BPCH		RGTS	
	Cost	Time	Cost	Gap (%)	Cost	Gap (%)	Cost	Gap (%)	Cost	Gap (%)
P1-00	532.0	18	530.8	−0.24	**527.7**	−0.82	–	–	529.2	−0.53
P1-110	461.0	31	462.9	0.40	–	–	**459.5**	−0.33	466.9	1.26
P1-1030	759.8	307	765.3	0.73	**758.2**	−0.21	770.2	1.37	784.6	3.27
P1-1050	1026.5	210	1,039.1	1.23	1,021.0	−0.53	**1,017.2**	−0.90	1,025.0	−0.14
P1-1090	1552.1	134	1,512.0	−2.58	1,497.2	−3.53	**1,489.4**	−4.04	1,503.3	−3.14
P1-3070	**1498.1**	151	1,503.9	0.39	1,502.0	0.26	1,499.3	0.08	1,503.2	0.34
P1-7090	2191.41	107.6	2,173.6	−0.81	2,166.8	−1.12	**2,166.3**	−1.15	2,195.7	0.19
P2-00	**827.4**	320	854.3	3.25	853.6	3.17	–	–	864.6	4.50
P2-110	637.0	44	**623.9**	−2.05	–	–	652.9	2.50	629.1	−1.25
P2-1030	**1118.1**	325	1,134.1	1.43	1,122.9	0.43	1,121.8	0.34	1,146.2	2.52
P2-1050	1525.7	318	1,556.7	2.03	1,548.5	1.50	**1,514.4**	−0.74	1,550.4	1.62
P2-1090	2358.8	322	2,338.7	−0.85	2,337.8	−0.89	**2,318.3**	−1.72	2,398.4	1.68
P2-3070	2280.3	406	2,293.5	0.58	2,263.1	−0.75	**2,237.2**	−1.89	2,240.0	−1.76
P2-7090	3259.0	200	3,285.4	0.81	**3,250.4**	−0.27	3,258.2	−0.03	3,259.4	0.01
P3-00	847.4	188	841.4	−0.71	**840.1**	-0.86	–	–	846.0	−0.17
P3-110	792.2	87	**771.5**	−2.62	–	–	788.2	−0.50	805.0	1.61
P3-1030	1476.9	326	1,515.2	2.59	1,505.5	1.93	**1,447.4**	−2.00	1,491.8	1.01
P3-1050	2023.2	318	2,054.1	1.53	2,024.6	0.07	2,040.9	0.88	2,062.5	1.94
P3-1090	3181.3	339	3,155.2	−0.82	3,136.3	−1.42	**3,127.1**	−1.71	3,171.6	−0.31
P3-3070	3044.1	325	3,070.9	0.88	3,055.5	0.37	**3,030.7**	−0.44	3,091.3	1.55
P3-7090	**4441.7**	300	4,470.7	0.65	4,452.6	0.25	4,467.6	0.58	4,465.0	0.53
P4-00	1081.6	426	1,070.9	−0.99	**1,055.1**	−2.45	–	–	1,059.7	−2.02
P4-110	953.0	369	**947.1**	−0.62	–	–	984.7	3.32	979.7	2.80
P4-1030	**2060.4**	375	2,101.8	2.01	2,093.3	1.60	2,066.5	0.30	2,093.2	1.59
P4-1050	**2910.8**	394	2,991.6	2.78	2,977.0	2.27	2,917.8	0.24	2,943.5	1.12
P4-1090	4681.7	389	4,674.1	−0.16	4,659.9	−0.47	4,678.5	−0.07	**4,652.1**	−0.63
P4-3070	4483.4	372	4,496.9	0.30	4,465.5	−0.40	**4,438.8**	−0.99	4,460.2	−0.52
P4-7090	**6459.8**	300	6,482.2	0.35	6,462.8	0.05	6,523.2	0.98	6,511.5	0.80
P5-00	1342.5	477	1,340.4	−0.16	**1,338.4**	−0.30	–	–	1,368.8	1.96
P5-110	**1126.6**	449	1,148.3	1.93	–	–	1,268.8	12.62	1,158.1	2.79
P5-1030	**2525.0**	418	2,585.8	2.41	2,582.6	2.28	2,596.9	2.85	2,571.0	1.82
P5-1050	**3542.5**	429	3,624.2	2.31	3,594.0	1.45	3,568.2	0.73	3,592.8	1.42
P5-1090	5700.7	500	5,715.8	0.27	5,710.2	0.17	**5,673.2**	−0.48	5,798.4	1.71
P5-3070	5585.1	438	5,571.1	−0.25	**5,549.8**	−0.63	5,560.3	−0.44	5,556.0	−0.52
P5-7090	8255.4	300	8,392.1	1.66	8,355.5	1.21	8,410.4	1.88	8,319.4	0.77
P11-00	1048.3	177	1,057.0	0.82	1,057.0	0.82	–	–	**1,043.9**	−0.42
P11-110	1119.2	344	**1,055.3**	−5.71	–	–	1,071.6	−4.25	1,099.3	−1.78
P11-1030	2953.1	344	3,060.5	3.64	3,017.9	2.20	2,983.8	1.04	**2,939.4**	−0.46
P11-1050	4298.4	345	4,502.6	4.75	4,476.4	4.14	**4,259.9**	−0.89	4,301.5	0.07
P11-1090	7206.2	358	7,350.1	2.00	7,117.2	−1.23	**6,995.9**	−2.92	6,967.5	−3.31
P11-3070	6858.1	354	7,168.3	4.52	7,126.8	3.92	6,822.3	−0.52	**6,770.1**	−1.28
P11-7090	10285.7	300	10,673.3	3.77	10,429.8	1.40	10,376.9	0.89	**10,132.5**	−1.49
# of best	10	–	4	–	7	–	14	–	5	–
Average	–	–	–	0.75	–	0.38	–	0.13	–	0.46

Compared with these four benchmark algorithms, the percentage gaps on the solution costs are 0.75%, 0.38%, 0.13% and 0.46%, respectively. Although these gaps are all below 1%, considering that the SDVRP was extensively studied in the last two decades, these improvements brought about by our approach are substantial. Thus, we can conclude that our approach outperforms SPLITABU, OH, BCPH and RGTS in terms of the solution quality when solving the instances in Set 1.

Tables 2 reports the results of the instances in Set 2, where the algorithms used to solve these instances can be found in the first row. Our approach and BPCH solved the instances without any restriction on the number of vehicles while the rest of algorithms dealt with the instances with the minimum possible number of vehicles. Since BPCH was unable to get feasible solutions for instances *eil22, eil23, eil30* and *eil33* in Set 3, we fill the corresponding cells in Table 2 with dashes ('–'). According to the number of the best solutions and the average gaps, we can find that our tabu search algorithm achieved the best performance on the instances in Set 2.

Our tabu search algorithm that employs the new neighborhood operators is very simple and standard. However, it achieved the best average results on 67 benchmark instances; this proved the effectiveness of the neighborhood operators.

5 Conclusions

In this study, we first represented the SDVRP solution using the combination of a set of vehicle routes and a forest. The vehicle routes are used to calculate the total travel distance while the forest is used to quickly check the feasibility of the solution. Next, we designed three classes of

Table 2: Performance comparison on the instances in Set 2.

Instance	Our approach		BPCH		ICA+VND		RGTS		TSVBA	
	Cost	Time	Cost	Gap (%)	Cost	Gap (%)	Cost	Gap (%)	Cost	Gap (%)
eil22	**375.3**	4	–	–	**375.3**	0.00	**375.3**	0.00	**375.3**	0.00
eil23	**568.6**	4	–	–	569.8	0.21	598.6	5.28	569.8	0.21
eil30	519.0	7	–	–	521.5	0.48	519.7	0.13	**505.0**	−2.70
eil33	**837.1**	10	–	–	870.4	3.98	843.6	0.79	843.6	0.79
eil51	528.0	23	528.0	0.00	540.8	2.43	**524.9**	−0.58	527.7	−0.06
eilA76	842.7	191	**832.7**	−1.18	880.3	4.46	860.9	2.16	853.2	1.25
eilB76	**1,017.1**	289	1,047.2	2.96	1,059.6	4.18	1,023.2	0.61	1,034.2	1.69
eilC76	754.3	73	765.3	1.46	758.5	0.56	**746.3**	−1.05	761.6	0.97
eilD76	701.1	57	705.3	0.60	719.4	2.62	702.3	0.17	**696.0**	−0.73
eilA101	**838.8**	194	854.0	1.82	854.1	1.83	850.0	1.34	844.2	0.65
eilB101	**1,096.1**	280	1,119.2	2.11	1,142.0	4.19	1,112.2	1.47	1,112.2	1.47
S51D1	464.8	13	**459.5**	−1.15	473.2	1.80	**459.5**	−1.15	468.8	0.85
S51D2	**711.9**	121	717.2	0.74	732.4	2.87	724.0	1.69	718.7	0.95
S51D3	**952.8**	215	960.4	0.79	1,001.2	5.08	970.7	1.87	969.8	1.78
S51D4	1,587.8	134	**1,569.9**	−1.13	1,708.0	7.57	1,614.1	1.66	1,628.2	2.55
S51D5	1,348.8	127	**1,339.4**	−0.70	1,404.5	4.13	1,381.7	2.44	1,362.2	0.99
S51D6	2,202.2	81	**2,182.1**	−0.91	2,230.1	1.26	2,213.9	0.53	2,236.2	1.54
S76D1	615.9	33	633.8	2.91	**610.2**	−0.93	629.6	2.22	613.7	−0.36
S76D2	**1,103.6**	329	1,104.6	0.09	1,169.8	6.00	1,113.4	0.89	1,128.2	2.22
S76D3	1,449.8	314	**1,435.1**	−1.02	1,490.1	2.78	1,460.0	0.70	1,472.9	1.59
S76D4	2,108.6	299	2,106.6	−0.09	2,220.9	5.32	**2,103.1**	−0.27	2,180.1	3.39
S101D1	**745.7**	223	791.1	6.09	765.5	2.66	791.2	6.11	749.9	0.57
S101D2	**1,394.6**	327	1,426.2	2.26	1,445.0	3.61	1,415.9	1.53	1,409.0	1.03
S101D3	1,913.3	325	1,911.1	−0.11	1,990.3	4.02	**1,907.9**	−0.28	1,947.6	1.79
S101D5	2,858.8	374	**2,824.2**	−1.21	2,999.3	4.92	2,896.0	1.30	2,910.7	1.82
# of best	11	–	7	–	2	–	6	–	3	–
Average	–	–	–	0.68	–	3.04	–	1.18	–	0.97

neighborhood search operators, namely *relocate, exchange* and *split*, and integrated them into a simple and standard tabu search framework. By comparing with six existing algorithms on 67 benchmark instances, we find that our tabu search algorithm achieved the best solutions on average. The experimental results show that our approach is an outstanding algorithm for the SDVRP. Future research can consider adapting our operators and developing more advanced neighborhood-based meta-heuristics for other vehicle routing variants that allow split delivery, such as the SDVRP with time windows or multi-depot SDVRP.

References

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

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

Archetti, C., and Speranza, M. G. 2012. Vehicle routing problems with split deliveries. *International Transactions in Operational Research* 19(1-2):3 – 22.

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

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

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

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

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

Cai, Y.; Zhang, Z.; Guo, S.; Qin, H.; and Lim, A. 2013. A tree-based tabu search algorithm for the manpower allocation problem with time windows and job-teaming constraints. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, 496–502. AAAI Press.

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

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

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

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

Glover, F., and Laguna, M., eds. 1998. *Tabu Search*. 101 Philip Drive, Assinippi Park, Norwell, Massachusetts 02061, USA: Kluwer Academic Publishers.

Ho, S. C., and 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.; and Eksioglu, B. 2008. A column generation approach for the split delivery vehicle routing problem. *Operations Research Letters* 36(2):265 – 270.

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

Mullaseril, P.; Dror, M.; and Leung, J. 1997. Split-delivery routeing heuristics in livestock feed distribution. *Journal of the Operational Research Society* 48(2):107 – 116.

Sierksma, G., and Tijssen, G. A. 1998. Routing helicopters for crew exchanges on off-shore locations. *Annals of Operations Research* 76:261 – 286.

Song, S. H.; Lee, K. S.; and Kim, G. S. 2002. A practical approach to solving a newspaper logistics problem using a digital map. *Computers and Industrial Engineering* 43(1-2):315 – 330.