



# An efficient local search algorithm for split delivery vehicle routing problem with three-dimensional loading

Han Zhang<sup>1,2</sup> · Qing Li<sup>2</sup> · Xin Yao<sup>3</sup>

Received: 10 April 2024 / Accepted: 2 March 2025 / Published online: 17 April 2025  
© The Author(s) 2025

## Abstract

The split delivery vehicle routing problem with three-dimensional loading constraints (3L-SDVRP) is a complex capacitated vehicle routing problem variant that considers split delivery and three-dimensional loading. It aims to determine the optimal routes for a fleet of vehicles by minimizing the number of vehicles required and the total travel distance. However, current methods are limited in efficiency and often yield suboptimal solutions. More efficient and effective methods are needed. Building on a state-of-the-art algorithm for solving the 3L-SDVRP, this paper proposes a more efficient algorithm with several novel features. Firstly, improvements to the packing method are introduced to enhance the loading performance and reduce required vehicles. Secondly, three new search operators are proposed to exploit problem characteristics in order to improve search efficiency significantly. Thirdly, a new adaptive splitting strategy dynamically decides when to split boxes according to the current status of the vehicle and node, thereby reducing computational costs. Lastly, the algorithm includes a new post-optimization method to further improve the solution quality. Extensive experiments validate that our proposed method efficiently reduces the number of required vehicles with fewer computational resources. The effectiveness of each novel component has also been confirmed through ablation experiments.

**Keywords** Routing · Packing · Split delivery · Hybrid evolutionary algorithms · Local search

## 1 Introduction

The split delivery vehicle routing problem with three-dimensional loading constraints (3L-SDVRP) is a challenging combinatorial optimization problem that combines two NP-hard problems, namely the three-dimensional packing (also called loading) problem and the split delivery vehicle routing problem. As shown in Fig. 1, each node contains a set of rectangular boxes to deliver. Multiple vehicles are required

to depart from a depot, travel to different customer nodes to load the boxes at each node, and subsequently return to the depot.

The 3L-SDVRP is different from other VRP variants ([17, 18, 22]) due to two key characteristics: (1) *Splitting of deliveries*. In real-world applications, vehicles have finite capacities, and the quantity and 3D sizes of boxes at each node may vary greatly. These differences can lead to two main issues: the total load from a node may exceed the maximum capacity of a vehicle, or the remaining capacity of a vehicle may be insufficient for all boxes when visiting a node. Consequently, the problem permits multiple vehicles to service the same node, enabling box delivery in separate batches. (2) *3D loading constraints*. In typical VRPs, only the volume and weight of boxes are considered to ensure that the load of a vehicle does not exceed its capacity. However, in the case of 3L-SDVRP, where boxes are treated as cuboids, it is necessary to consider not only the volume and weight, but also the 3D size, loading sequence, and placement position.

The features mentioned above add additional complexity to the problem because they require optimization of both routing and loading. Due to the high complexity of

✉ Xin Yao  
xinyao@ln.edu.hk

Han Zhang  
cshzhang@comp.polyu.edu.hk

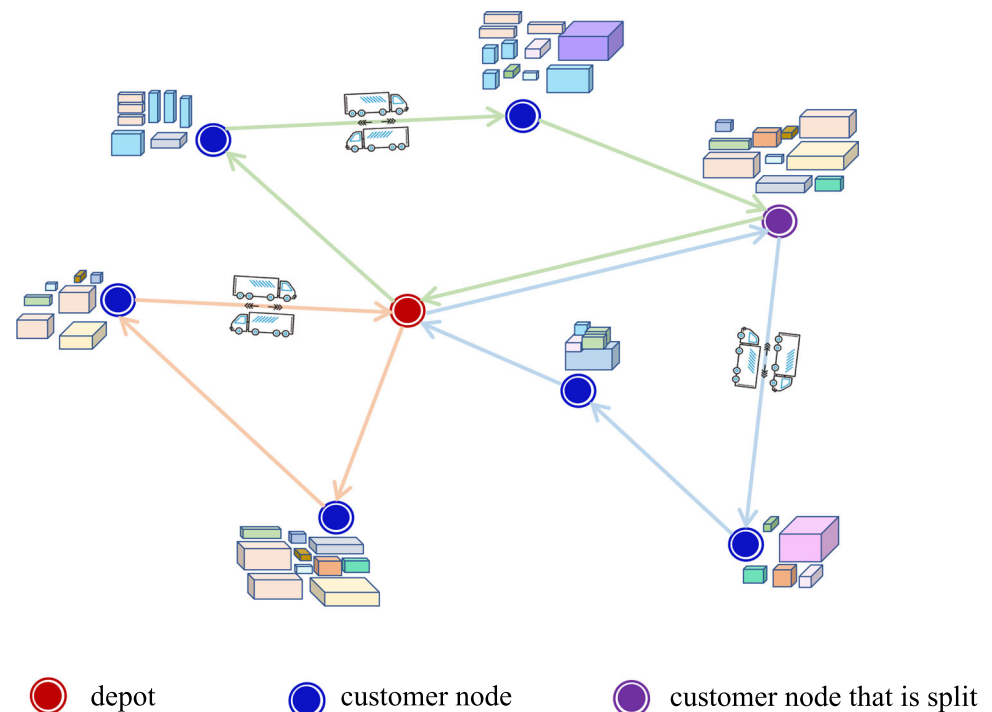
Qing Li  
qing-prof.li@polyu.edu.hk

<sup>1</sup> Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China

<sup>2</sup> Department of Computing, The Hong Kong Polytechnic University, Hong Kong, China

<sup>3</sup> School of Data Science, Lingnan University, Hong Kong, China

**Fig. 1** Illustration of the 3L-SDVRP



the 3L-SDVRP, exact algorithms are incapable of solving it effectively. Consequently, many heuristics [25], metaheuristics [16, 19, 20, 31], and hybrid algorithms [5, 7, 15] have been proposed in previous studies. However, despite these advanced algorithms, solving the problem efficiently and effectively remains challenging.

For example, Liu et al [16] introduced a machine learning-assisted multi-objective evolutionary algorithm named MOEA/D-OM for solving the 3L-SDVRP. This innovative approach integrates machine learning techniques to enhance the evolutionary search process, offering a novel solution to this complex optimization problem. However, its success largely depends on generating many packing-infeasible solutions during the search. The pre-trained packing evaluation model may limit algorithm adaptability and performance across diverse problems due to potential constraints in generalization ability of the training data. Additionally, MOEA/D-OM is not well-suited for solving large-scale problems: the instances used in their study contain a maximum of only 16 nodes, with an average of approximately 9 nodes and 400 boxes per instance, indicating that the problem size is relatively small. In contrast, Bortfeldt and Yi [5] proposed a local search-based algorithm called SDVRLH2, which maintains the best-known solutions for most problem instances in terms of solution quality and can solve large-scale problems (up to 200 nodes and 14230 boxes). Thus, SDVRLH2 can be considered the state-of-the-art method for solving the 3L-SDVRP. However, SDVRLH2 has the following weakness:

- The packing algorithm employs a strategy of packing two boxes at a time, which generates only two sub-spaces, potentially leading to wasted space and an increased number of utilized vehicles.
- The routing process of the algorithm utilizes traditional search operators, such as swap, shift, 2-opt, and 3-opt, but fails to exploit problem properties as heuristic information, thus reducing the search efficiency.
- The outer loop, which varies maximal admissible splitting costs, ignores the status of vehicle/node and causes high computational overhead.
- The post-optimization phase, which uses 3-opt, is computationally expensive and not very effective.

Inspired by the state-of-the-art SDVRH2 algorithm for solving the 3L-SDVRP, we propose a more efficient local search algorithm. Our contributions can be summarized as follows:

- We improve the box loading and sub-space generation process by two distinct approaches: (a) packing two boxes at once, generating five sub-spaces, and (b) packing one box at a time, generating three sub-spaces. Both methods generate packing solutions, and a heuristic rule is employed to determine the optimal packing solution. Additionally, the two customers packing pattern (2C-SP, which is described in Sect. 3.2) construction method is also adjusted.

- For routing, we design three novel search operators that exploit problem properties as heuristic information, resulting in improved efficiency and effectiveness.
- We propose an adaptive splitting strategy to replace the multiple local search loops in SDVRLH2. This strategy evaluates splitting feasibility at each node based on vehicle remaining capacity and box layer configuration, thus reducing computational overhead.
- We develop a new post-optimization approach which includes two stages: reassigning vehicle loads to reduce the number of utilized vehicles, and independently optimizing the travel distance of each vehicle. This further enhances the quality of the final solution.

The remainder of this paper is structured as follows. In Sect. 2, we provide a mathematical formulation of 3L-SDVRP and review related work. In Sect. 3, we introduce our algorithm, comprehensively describing its overall framework and key features. In Sect. 4, we present experimental studies, including comparisons with the state-of-the-art algorithm and in-depth analysis of our method. Finally, we conclude our paper in Sect. 5, summarizing our contributions and highlighting future research directions.

## 2 Background and related work

### 2.1 Problem formulation

This problem involves multiple customer nodes and a single depot. As shown in Fig. 1, the depot serves as both the starting and ending nodes. Distances between all nodes are predetermined. At each node, there are boxes characterized by weight and 3D size (length, width, height). The task is to dispatch vehicles to transport all boxes from these customer nodes to the depot. This requires decisions on the number of vehicles, the route for each vehicle, the assignment of boxes to vehicles, their spatial arrangement within vehicles, and the order of loading. The objectives are to minimize the number of vehicles and the total travel distance (*tt**d*) for all vehicles, addressing both routing and packing challenges.

#### 2.1.1 Notations

Assuming that there are  $N$  customer nodes and a single depot (with index 0 and  $N+1$ ) as the starting and ending nodes. The distances between all pairs of nodes are known. Additionally,  $N$  sets of boxes are provided, with all boxes at each node belonging to a single set. Each box is characterized by its weight and 3D sizes (i.e., length, width, and height). The notations used in the mathematical formulation are shown in Table 1. Decision variables include the number of vehicles used ( $p$ ), the route (i.e., nodes sequence) of each vehicle, and

the packing scheme (i.e., the allocation of boxes to vehicles, the 3D coordinates of each box, and the loading sequence for all boxes within each vehicle).

#### 2.1.2 A mathematical model

The following mathematical model is constructed based on Moura and Oliveira [20]. There are two objectives relevant to both routing and packing aspects of the problem. The first one:

$$\min f_1 = p. \quad (1)$$

In Eq. (1),  $p$  is the number of vehicles used. The second one:

$$\min f_2 = \sum_{t=1}^p \sum_{i=0}^{N+1} \sum_{j=0}^{N+1} d_{ij} X_{ij}^t. \quad (2)$$

The optimization process focuses on minimizing two objective functions. The first objective function,  $f_1$ , aims to minimize the number of vehicles required, while the second objective function,  $f_2$ , seeks to minimize the total travel distance (*tt**d*) for all vehicles.

Although 3L-SDVRP inherently involves two objectives, it is conventionally treated as a single-objective problem in most literature ([5, 7, 8, 15, 20, 25, 31]). In practice, minimizing the number of vehicles is considered more crucial than reducing total travel distance ([5]), due to the substantially greater expenses of obtaining, maintaining extra vehicles, and employing more drivers compared to the additional travel costs. Therefore, in alignment with previous studies, this paper prioritizes the optimization of the number of vehicles used as the primary objective, with the minimization of total travel distance (*tt**d*) as the secondary objective. When evaluating the quality of two solutions during the search process, we first examine the number of vehicles, with fewer vehicles indicating a superior solution. When the number of vehicles is the same, we then compare the total travel distances, with shorter distances representing better solutions.

The constraints are as follows.

$$\sum_{j=1}^N X_{0j}^t = 1, \quad t = 1, \dots, p. \quad (3)$$

$$\sum_{i=1}^N X_{i(N+1)}^t = 1, \quad t = 1, \dots, p. \quad (4)$$

$$\sum_{i=0}^N X_{ik}^t = \sum_{j=1}^{N+1} X_{kj}^t, \quad k = 1, \dots, N; \quad t = 1, \dots, p. \quad (5)$$

**Table 1** Notations used in the mathematical formulation

| Notations                                  | Descriptions                                                                                                                           |
|--------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------|
| $G = (V, E)$                               | Road network, a complete directed graph                                                                                                |
| $V = \{0, 1, \dots, N, N + 1\}$            | $V$ is the set of vertices<br>0 and $N + 1$ are the depot (i.e., the starting point and ending point)<br>$1 \sim N$ are customer nodes |
| $E = \{(i, j) \mid i, j \in V, i \neq j\}$ | the set of edges                                                                                                                       |
| $d_{ij}$                                   | Non-negative travel distance, $d_{ij} \neq d_{ji}$                                                                                     |
| $O = \{K_i \mid i = 1, \dots, M\}$         | one delivery order/dataset; $1 \leq M \leq N$                                                                                          |
| $K_i = \{1, \dots, m_i\}$                  | the set of boxes that need to be transported in the node $i$<br>$m_i$ : Number of boxes in the node $i$                                |
| $A_i^t \subset K_i$                        | The set of boxes that are loaded in vehicle $t$ in the node $i$                                                                        |
| $l_k, w_k, h_k$                            | The length, width, and height of the box $k$                                                                                           |
| $q_k, v_k$                                 | the weight and volume of the box $k$                                                                                                   |
| $x_k, y_k, z_k$                            | The coordinate of the center of the box $k$                                                                                            |
| $L_t, W_t, H_t, t = 1, \dots, p$           | the length, width, height of container vehicle $t$<br>$p$ : Total number of vehicles used to transport boxes                           |
| $Q_t, t = 1, \dots, p$                     | The weight capacity of container vehicle $t$                                                                                           |
| $X_{ij}^t$                                 | $= 1$ if vehicle $t$ travels from node $i$ to $j$<br>$= 0$ otherwise; ( $i, j = 0, 1, \dots, N + 1; t = 1, \dots, p$ )                 |
| $x_{k1}, y_{k1}, z_{k1}$                   | The smallest $x, y$ , and $z$ coordinates of the box $k$ in the vehicle.                                                               |
| $x_{k2}, y_{k2}, z_{k2}$                   | The largest $x, y$ , and $z$ coordinates of the box $k$ in the vehicle                                                                 |
| $x_{t1}, y_{t1}, z_{t1}$                   | The smallest $x, y$ , and $z$ coordinates of the container of the vehicle $t$ (the origin of the coordinate system)                    |
| $x_{t2}, y_{t2}, z_{t2}$                   | The largest $x, y$ , and $z$ coordinates of the container of the vehicle $t$                                                           |

$$\sum_{i=0; i \neq j}^N X_{ij}^t \leq 1, \quad k \in A_i^t, \quad t = 1, \dots, p; \quad i = 1, \dots, N, \quad (6)$$

$$B_k : \text{the set of boxes under and touching the box } k \text{ in vehicle } t. \quad (13)$$

$$\delta_1 (x_{k2} - x_{l1}) (x_{l2} - x_{k1}) \leq 0, \quad k, l \in A_i^t; \quad i = 1, \dots, N; \quad t = 1, \dots, p. \quad (7)$$

$$\sum_{k \in A_i^t} q_k \leq Q_t, \quad i = 1, \dots, N; \quad t = 1, \dots, p. \quad (14)$$

$$\delta_2 (y_{k2} - y_{l1}) (y_{l2} - y_{k1}) \leq 0, \quad k, l \in A_i^t; \quad i = 1, \dots, N; \quad t = 1, \dots, p. \quad (8)$$

$$x_{k1} \geq x_{t1}, \quad k \in A_i^t; \quad i = 1, \dots, N; \quad t = 1, \dots, p. \quad (15)$$

$$\delta_3 (z_{k2} - z_{l1}) (z_{l2} - z_{k1}) \leq 0, \quad k, l \in A_i^t; \quad i = 1, \dots, N; \quad t = 1, \dots, p. \quad (9)$$

$$y_{k1} \geq y_{t1}, \quad k \in A_i^t; \quad i = 1, \dots, N; \quad t = 1, \dots, p. \quad (16)$$

$$\delta_1, \delta_2, \delta_3 \in \{0, 1\}, \quad \delta_1 + \delta_2 + \delta_3 \geq 1. \quad (10)$$

$$\bigcup_{t=1, \dots, p} A_i^t = K_i, \quad i = 1, \dots, N. \quad (11)$$

$$z_{k1} \geq z_{t1}, \quad k \in A_i^t; \quad i = 1, \dots, N; \quad t = 1, \dots, p. \quad (17)$$

$$\bigcap_{t=1, \dots, p} A_i^t = \emptyset, \quad i = 1, \dots, N. \quad (12)$$

$$x_{k2} \leq x_{t2}, \quad k \in A_i^t; \quad i = 1, \dots, N; \quad t = 1, \dots, p. \quad (18)$$

$$y_{k2} \leq y_{t2}, \quad k \in A_i^t; \quad i = 1, \dots, N; \quad t = 1, \dots, p. \quad (19)$$

$$z_{k2} \leq z_{t2}, \quad k \in A_i^t; \quad i = 1, \dots, N; \quad t = 1, \dots, p. \quad (20)$$

$$\sum_{b \in B_k} [\min\{x_{b2} - x_{k1}, x_{k2} - x_{b1}\} \cdot \min\{y_{b2} - y_{k1}, y_{k2} - y_{b1}\}] \geq r \cdot w_k l_k,$$

$I_k > I_l$ , if equations (22, 23, 25) or (23, 24, 26) hold simultaneously.  $k \in A_i^t$ ;  $l \in A_j^t$ ;  $i, j = 1, \dots, N$ ;  $t = 1, \dots, p$ .  $I_k, I_l$ : the loading sequence of box  $k$  and  $l$  in the vehicle  $R_t$ .

(21)

$$(x_{k2} - x_{l1})(x_{l2} - x_{k1}) > 0. \quad (22)$$

$$(y_{k2} - y_{l1})(y_{l2} - y_{k1}) > 0. \quad (23)$$

$$(z_{k2} - z_{l1})(z_{l2} - z_{k1}) > 0. \quad (24)$$

$$z_{k1} \geq z_{l2}. \quad (25)$$

$$x_{k1} \geq x_{l2}. \quad (26)$$

The constraints in this study primarily focus on two aspects: vehicle routing and 3D loading. Equations (3) and (4) ensure valid route formation: each route must start from the starting node and end at the ending node. In this paper, the depot represents both the starting and ending nodes. Additionally, vehicles must adhere to flow conservation (Eq. (5)), which dictates that the number of vehicles arriving at a node must equal the number of vehicles departing from the node. Equation (6) enforces a single visit requirement, meaning that a customer can only be visited by the same vehicle once. The remaining constraints relate to 3D loading. Non-overlapping constraints (Eqs. (7–10)) ensure that boxes within the same vehicle do not overlap in any dimension. Non-splitting constraints (Eqs. (11–12)) specify that a box can only be loaded into one vehicle. Equation (13) is the full supporting stability constraint in which  $B_k$  represents the set of all boxes that are under box  $k$  and in contact with the bottom surface of  $k$ . Thus, this constraint requires that the supporting area must be greater than or equal to the percentage  $r$  of the bottom area of the box ( $r = 1.0$  in this paper). Weight capacity constraint (Eq. (14)) specifies that the total weight of boxes loaded onto a vehicle cannot exceed the weight capacity of a vehicle. Size constraints (Eqs. (15–20)) mean that all boxes must be loaded inside the vehicle container, with exceeding the container wall or door prohibited. Lastly, the last-in-first-out (LIFO) constraint, defined by Eqs. (21–26), ensures that boxes loaded later must be unloaded first. Once a vehicle leaves a node, boxes loaded there cannot be moved.

## 2.2 Related work

Table 2 summarizes the problem properties considered in various studies on 3L-SDVRP. It is evident that the problem exhibits varying characteristics in different scenarios, which must be taken into account when designing algorithms. Thus, a thorough analysis of the constraints and characteristics of the problem in each scenario is crucial for the effective design of algorithms.

Algorithms for the 3L-SDVRP can be categorized into local search-based and global search-based methods. Local search approaches encompass classic local search ([5, 20]), tabu search ([8, 31]), and simulated annealing ([7]). For global search, MOEA/D ([16]), estimation of distribution algorithms ([15]), and genetic algorithms ([19, 21]) are employed. In addressing packing constraints within the literature, various methods have been employed, including genetic algorithms ([5, 31]), local search techniques ([8, 19, 20]), constructive heuristics ([25]), and a bundle of heuristic approaches ([7, 15]), all aimed at determining efficient packing arrangements.

Global search-based approaches offer the advantage of exploring a broader solution space, potentially avoiding local optima. However, they usually encounter challenges such as higher computational costs and slower convergence rates, especially in the context of complex and large-scale problems. The most recent global search-based method, introduced by Liu et al [16], leverages machine learning models to evaluate the packing feasibility of routes, which improve the efficiency of solving the 3L-SDVRP. Despite this progress, the authors acknowledge the method is limited in its ability to handle larger-scale problems effectively. Moreover, in terms of solution quality, this approach falls short of the performance of SDVRLH2.

SDVRLH2, a local search-based algorithm by Bortfeldt and Yi [5], leads the field due to its strong performance across various problem instances. Nonetheless, it still has some limitations: (1) The packing algorithm employs a strategy of packing two boxes at a time, which generates only two sub-spaces, potentially leading to wasted space and an increased number of utilized vehicles. (2) The routing component of the algorithm utilizes traditional search operators, such as swap, 2-opt, and 3-opt, but fails to exploit problem properties as heuristic information, thus reducing the search efficiency. (3) The outer loop, designed to vary the maximal admissible splitting costs, incurs a substantial computational overhead. (4) Its post-optimization phase, which employs 3-opt, is computationally expensive and not very effective.

Therefore, our research aimed at proposing a more efficient algorithm on the basis of SDVRLH2. In the next section, we introduce our algorithm in details.

## 3 An efficient local search algorithm for 3L-SDVRP

Building on the SDVRLH2 algorithm ([5]), we propose a more efficient algorithm that incorporates several innovations to tackle the limitations of SDVRLH2. This section first gives an overview of the framework of our algorithm, and then provides an in-depth discussion of each innovation introduced.

**Table 2** 3L-SDVRP problem properties considered in different research

| Literature              | BD | HV | TW | WL | LIFO | Re | Or | VS | FS | VN |
|-------------------------|----|----|----|----|------|----|----|----|----|----|
| Liu et al [16]          | ✗  | ✓  | ✗  | ✓  | ✓    | ✗  | ✓  | ✓  | ✗  | ✗  |
| Rajaei et al [25]       | ✓  | ✓  | ✗  | -  | ✓    | ✓  | ✓  | ✗  | ✓  | ✗  |
| Pei et al [21]          | ✗  | ✓  | ✗  | ✓  | ✓    | ✗  | ✓  | ✓  | ✗  | ✗  |
| Bortfeldt and Yi [5]    | ✓  | ✓  | ✗  | ✓  | ✓    | ✗  | ✓  | ✓  | ✗  | ✗  |
| Chen et al [8]          | ✓  | ✗  | ✓  | ✓  | ✗    | ✗  | ✗  | ✗  | ✗  | ✓  |
| Li et al [15]           | ✓  | ✓  | ✗  | ✓  | ✓    | ✗  | ✓  | ✓  | ✗  | ✗  |
| Yi and Bortfeldt [31]   | ✓  | ✗  | ✗  | ✗  | ✗    | ✗  | ✓  | ✓  | ✗  | ✗  |
| Ceschia et al [7]       | ✓  | ✓  | ✗  | ✓  | ✓    | ✓  | ✓  | ✗  | ✗  | ✓  |
| Moura and Oliveira [20] | ✓  | ✗  | ✓  | ✗  | ✗    | ✗  | ✓  | ✗  | ✓  | ✓  |
| Moura [19]              | -  | ✗  | ✓  | ✗  | ✗    | ✗  | ✗  | ✗  | ✓  | -  |
| Ours                    | ✓  | ✗  | ✗  | ✓  | ✓    | ✗  | ✓  | ✗  | ✓  | ✗  |

BD = back to depot; HV = heterogeneous vehicles; TW = time window; WL = weight limit; LIFO = last-in-first-out; Re = reachability; Or = orientation; VS = vertical stability; RS = robust stability; FS = full support stability; SS = surrounded stability; LBS = load bearing strength; VN = vehicle number limit; The symbol “-” means the relative entry is not mentioned in the literature

### 3.1 Overview of our algorithm

Following SDVRLH2, our algorithm employs a “packing first routing second” strategy. As illustrated in Algorithm 1, this approach first utilizes packing heuristics to generate feasible packing arrangements (Steps 1–2), followed by a local search-based approach for routing optimization. In the routing process, a “giant tour” (also termed as an “individual” in Algorithm 1) acts as the representation of a solution. It is a sequence of all nodes but does not represent a feasible route, as a single vehicle cannot traverse all nodes in one route without violating constraints. Therefore, a decoding process is essential to transform a giant tour into a feasible routing and packing plan for evaluation. Detailed insights into the giant tour decoding are provided in Appendix A.

Referencing Algorithm 1, initially, we generate a random initial individual as the best individual discovered so far (Step 3). We conduct iterative exploration based on the current best individual,  $s_{best}$  (Steps 4–20). This process involves multiple searches (Steps 6–19), each centered on the current individual,  $s_{curr}$ . For  $s_{curr}$ , we first employ our proposed LKH mutation operator to identify an improved individual,  $s_{iter\_best}$  (Step 7). Next, we utilize our proposed route-pair swap operator to obtain the neighborhood,  $N_{swap}$ , of  $s_{iter\_best}$  (Step 8). After merging  $N_{swap}$  with  $s_{iter\_best}$  (Step 9), we decode individuals in  $N$  and select the best individual among them as the new  $s_{iter\_best}$  (Step 10). We then apply our multi-pair elitist recombination operator to get the neighborhood  $N_{MPER}$  (Step 11). This is done for individuals in  $N_{swap}$  that outperform the original  $s_{iter\_best}$  before the route-pair swap operation. We choose the best individual from  $N_{MPER}$  and  $s_{iter\_best}$  as the new  $s_{iter\_best}$  (Steps 12–13). Following the update of  $s_{best}$  and  $s_{curr}$  (Steps 14–15),

we terminate the current loop if  $s_{curr}$  has not been updated for  $n_{no\_imp}$  consecutive iterations (Steps 16–18). Once the search is completed, the algorithm implements the new post-optimization method on the best individual,  $s_{best}$ , for further optimization (Steps 21–25).

Overall, Algorithm 1 extends the hill-climbing approach by conducting exhaustive local searches with various moves from the current best solution,  $s_{best}$ . In the following subsections, we will describe our innovative contributions in detail.

### 3.2 Improved packing method

Our layer-based packing strategy, inspired by methods from Bortfeldt and Yi [5] and Bortfeldt and Gehring [4], organizes boxes at each node into cuboid basic layers as shown in Fig. 2. By checking the layer depth against the x-axis length of the vehicle, this approach simplifies the solving process. It eliminates the need for frequent 3D packing problem resolutions during routing, thus reducing computational requirements.

Building on the basic layer concept, Bortfeldt and Yi [5] introduced three packing patterns: one customer full load pattern (1C-FLP), one customer segment pattern (1C-SP), and two customers segment pattern (2C-SP). 1C-FLP groups basic layers from one node into a full vehicle load for direct delivery. When layers of a node exceed the vehicle capacity, one or more 1C-FLPs are formed as not all layers fit. The remaining basic layers of the node, which do not constitute any 1C-FLP, form a 1C-SP. The 2C-SP integrates 1C-SPs from two distinct nodes to formulate a mixed layer containing boxes from both nodes (as illustrated in Fig. 3). This approach strategically selects combinations that maximize space utilization, thereby minimizing the overall volume occupied by the boxes. By merging layers from different

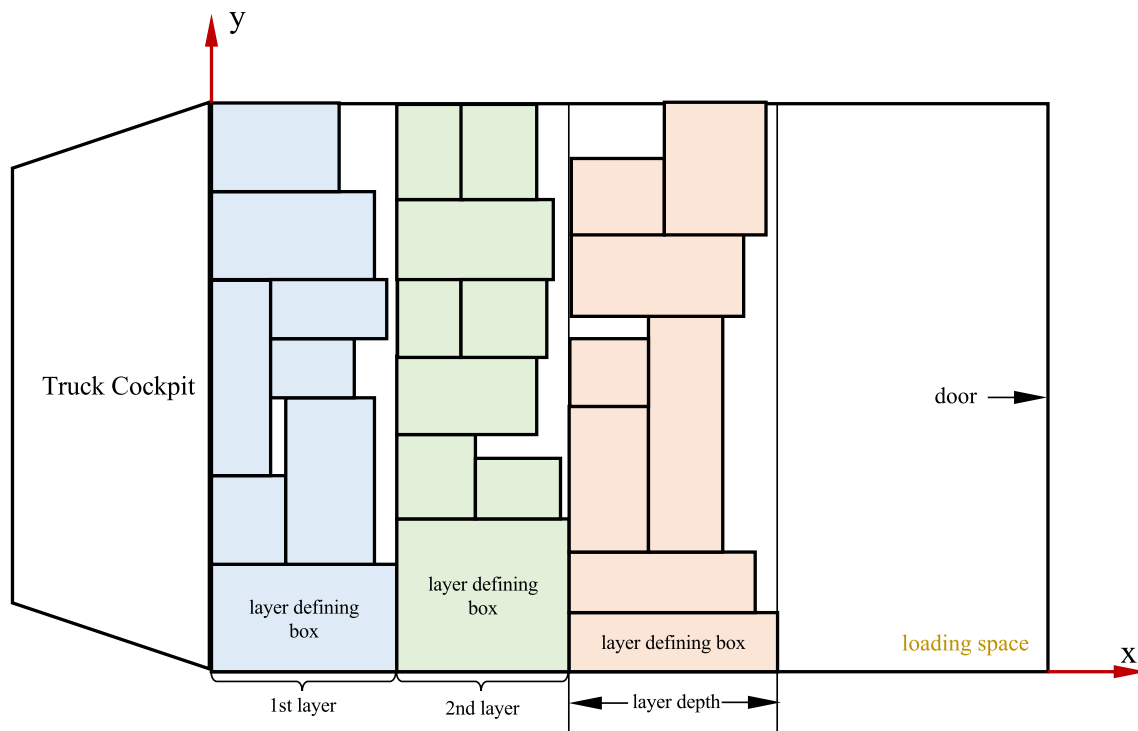


**Fig. 2** Packing plan with three basic layers (top-down view)

nodes, the 2C-SP not only optimizes packing efficiency but also potentially reduces the number of vehicles required for transportation, leading to improved operational efficiency and cost-effectiveness in logistics operations. For comprehensive definitions of these patterns, readers are referred to Appendix B and Bortfeldt and Yi [5].

Our enhancements to the packing method focus on two main aspects: refining the box-packing and space-generation strategy, and advancing the construction technique of the 2C-SP.

### 3.2.1 Improved box-placement and space-generation strategy

The basic layer is the smallest vehicle loading unit in our method. Comprehensive details on constructing a basic layer are provided in the Appendix B. In the packing phase, the term “space” denotes an empty cuboidal area available for box placement. Loading a box means aligning its rear-bottom-left vertex with the corresponding corner of the space, which creates new sub-spaces for further loading, changing the original space. This operation is crucial to the packing algorithm, significantly influencing its performance.

In Bortfeldt and Yi [5], the loading approach is loading two boxes (i.e., a box pair) at a time: first, load one box into current space  $s_{curr}$ , generate three subspaces, then load the second box into one of the three subspaces, and add the remaining two subspaces to the set of available spaces  $S_{avail}$

(note that the second box does not generate new subspaces). If no suitable box pair exists in  $B_{free}$ , one box that fits into  $s_{curr}$  is loaded. Through analysis, we believe that the method of loading box pairs has two disadvantages: it generates fewer subspaces, as the second box does not generate subspaces, potentially leading to space wastage; and the selected box pair may not maximize space utilization.

Consequently, we improved the box-packing and space-generation strategy in two directions, resulting in two modified strategies. One enhanced method (denoted as PM1) increases the number of subspaces generated. In the original method, two boxes are loaded at a time, with the second box not generating subspaces. Our improved strategy generates subspaces along the axes after loading both boxes, resulting in five subspaces, which enhances space utilization. Another improved method (denoted as PM2) loads one box at a time, creating three subspaces along the  $x$ ,  $y$ , and  $z$  axes, instead of loading two boxes at once.

Two improvement directions yield two modified packing methods, each producing a packing solution ( $p_1$  and  $p_2$ ). When entering the routing stage, a packing solution is selected based on the selection criteria. As shown in Algorithm 2, for each packing solution, first calculate the sum of the depths of all packing patterns (including 1C-FLP, 1C-SP, and 2C-SP) as  $l_1$  and  $l_2$  (Step 1), then divide  $l_1$  and  $l_2$  by the vehicle length  $l_v$  to obtain  $v_1$  and  $v_2$  (Step 2). Let  $n_{2C-SP}^{(1)}$  and  $n_{2C-SP}^{(2)}$  represent the number of 2C-SPs in  $p_1$  and  $p_2$ , respectively (Step 3). When  $v_2 - v_1 \leq \alpha_1$  or

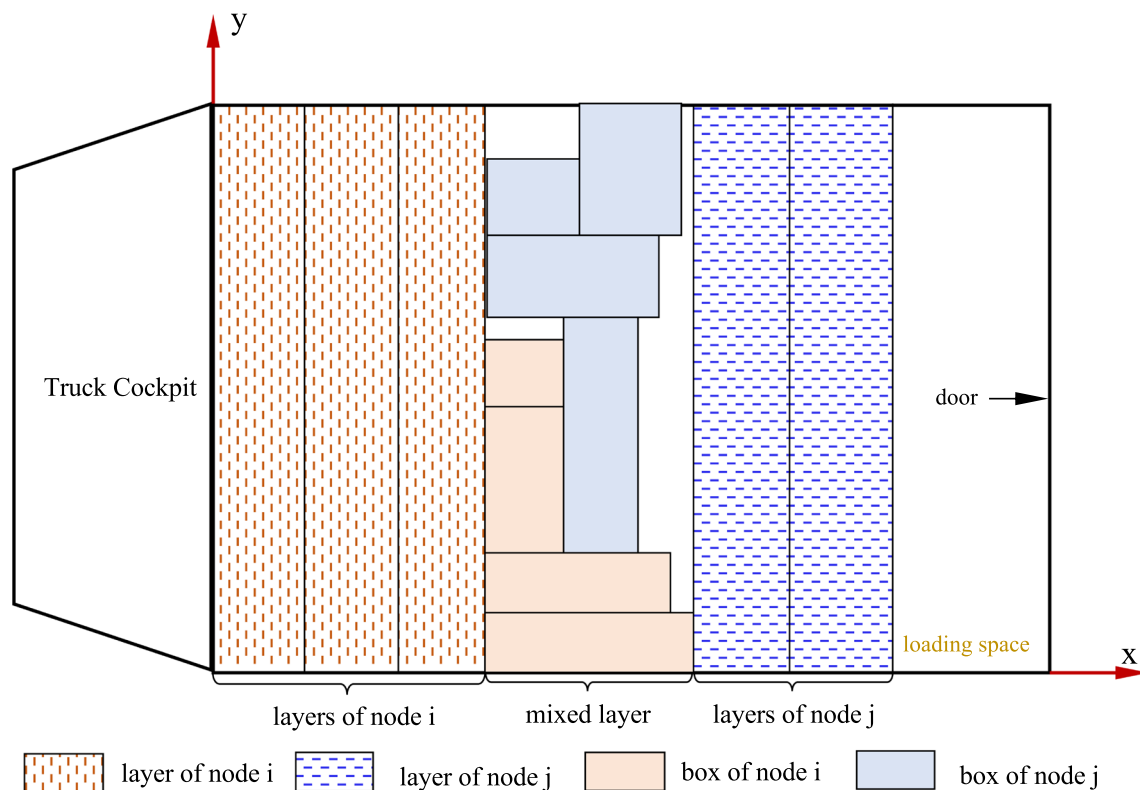


**Fig. 3** Illustration of two customers segment pattern (2C-SP). The mixed layer contains boxes of both nodes

$n_{2C-SP}^{(2)} - n_{2C-SP}^{(1)} \leq \alpha_2$ ,  $p_2$  is selected as the final packing solution; otherwise,  $p_1$  is chosen (Steps 4–8).

This approach tries to select the most suitable packing solution, taking into account both space utilization and the number of 2C-SPs. By implementing the two modified strategies, our method effectively improves the overall packing performance.

### 3.2.2 Multi-stage 2C-SP construction

As shown in Fig. 3, 2C-SPs include box layers from two different nodes and a mixed layer that contains boxes from both nodes. We extend the method for constructing 2C-SPs in SDVRLH2. The original approach allows a node to attempt 2C-SP construction with only some adjacent nodes. After construction, many nodes that could form 2C-SPs may remain. To address this issue, we propose a multi-stage method for 2C-SP construction, as described in Algorithm 3. This method considers  $N$  as a set of all nodes that have not yet constructed a 2C-SP. It applies the method from Bortfeldt and Yi [5] to construct 2C-SP for nodes in  $N$  (Step 4). Following a construction cycle,  $N$  is updated to only include nodes that have not yet constructed a 2C-SP (Step 7), and the procedure is reiterated until no further 2C-SPs can be established (Steps 3–7). Compared to a single construction

cycle, this method promises a more exhaustive formation of 2C-SPs, thereby further reducing the number of vehicle used.

### 3.3 New search operators

This study underscores the importance of employing advanced search operators that leverage domain knowledge in the search process. Commonly used operators in current research on the 3L-SDVRP include the swap operator ([5, 32]), the shift operator ([5, 29]), the 2-opt operator ([5, 26]), and the 3-opt operator ([5, 26]). Specifically, the swap operator exchanges the positions of two nodes; the shift operator moves a node to another position; the 2-opt operator breaks two edges and reconnects them to alter the node order; and the 3-opt operator breaks three edges and reconnects them to create a new sequence of nodes. While these general operators are applicable to a variety of combinatorial optimization problems, they lack the ability to effectively leverage problem-specific characteristics, leading to sub-optimal search performance when tackling problems with complex constraints or large-scale instances.

In the 3L-SDVRP studied in this paper, we use the giant tour as the solution representation, which is then decoded into multiple routes and feasible 3D packing plans for each route (i.e., each vehicle). The aforementioned general operators operate on the entire giant tour but typically have small

**Algorithm 1** Overview of our method (Key innovations in red boxes.)

**Input:** problem data, parameters  
**Output:** best solution  $s_{best}$

```

// Get packing arrangement
1:  $p_1, p_2 \leftarrow$  generate basic layer, formulate 1C-FLP and 1C-SP
   packing pattern, then construct 2C-SP (Algorithm 3) by using
   PM1 and PM2 respectively
2:  $p \leftarrow$  select packing solution from  $p_1$  and  $p_2$  for routing (Algo-
   rithm 2)
   // Perform routing process with  $p$ 
3: generate initial individual  $s_{init}$  and set  $s_{best} \leftarrow s_{init}$ 
4: for  $t \leftarrow 1$  to  $n_{out}$  do
5:    $s_{curr} \leftarrow s_{best}$ 
6:   for  $iter \leftarrow 1$  to  $n_{iter}$  do
7:      $s_{iter\_best} \leftarrow$  perform LKH mutation on  $s_{curr}$  and get
       updated solution
8:      $N_{swap} \leftarrow$  get neighborhood of  $s_{iter\_best}$  by route-pair swap
       (Algorithm 4)
9:      $N \leftarrow N_{swap} \cup \{s_{iter\_best}\}$ 
10:     $s_{iter\_best} \leftarrow$  decoding and evaluate individuals in  $N$ 
       (including determining the packing schedule of each vehicle,
       splitting arrangement (Algorithm 6), and route), then determine
       best individual
11:     $N_{MPER} \leftarrow$  get neighbourhood based on good individuals in
        $N_{swap}$  by multi-pair elitist recombination (Algorithm 5)
12:     $N \leftarrow N_{MPER} \cup \{s_{iter\_best}\}$ 
13:     $s_{iter\_best} \leftarrow$  decoding and evaluate individuals in  $N$ 
       (including determining the packing schedule of each vehicle,
       splitting arrangement (Algorithm 6), and route), then determine
       best individual
14:    update  $s_{best}$  by  $s_{iter\_best}$  where necessary
15:     $s_{curr} \leftarrow s_{iter\_best}$ 
16:    if  $s_{curr}$  has not been improved for  $n_{no\_imp}$  consecutive itera-
       tions then
17:      break
18:    end if
19:  end for
20: end for

// Post-optimization
21: if number of node  $\geq 100$  then
22:    $s_{best} \leftarrow$  perform reassignment (Algorithm 7) and single route
     optimization to  $s_{best}$ 
23: else
24:    $s_{best} \leftarrow$  perform single route optimization to  $s_{best}$ 
25: end if

26: return  $s_{best}$ 

```

search step sizes and overlook that a giant tour inherently consists of multiple distinct routes. To conduct more effective searches, we leveraged this characteristic and designed three novel search operators: the LKH mutation, the route-pair swap (RPS), and the multi-pair elitist recombination (MPER). The LKH mutation operator optimizes each route obtained after decoding the giant tour. The RPS operator selects two routes from the decoded giant tour and performs node swaps between them. The MPER operator aggregates

**Algorithm 2** Select a packing solution (Key innovation in red box.)

**Input:**  $p_1, p_2$ : two packing solutions obtained by the two improved methods PM1 and PM2;  $l_v$ : length of vehicle  
**Output:**  $p_{final}$ : selected packing solution

```

1:  $l_1, l_2 \leftarrow$  the sum of depth of all packing patterns (1C-FLP, 1C-SP,
   2C-SP) in  $p_1, p_2$ 
2:  $v_1 \leftarrow l_1 / l_v$ ;  $v_2 \leftarrow l_2 / l_v$ 
3:  $n_{2C-SP}^{(1)}, n_{2C-SP}^{(2)} \leftarrow$  the number of 2C-SP in  $p_1, p_2$ 
4: if  $v_2 - v_1 \leq \alpha_1$  or  $n_{2C-SP}^{(2)} - n_{2C-SP}^{(1)} \leq \alpha_2$  then
5:    $p_{final} \leftarrow p_2$ 
6: else
7:    $p_{final} \leftarrow p_1$ 
8: end if

9: return  $p_{final}$ 

```

**Algorithm 3** Multi-stage 2C-SP construction (Key innovation in red box.)

**Input:**  $N$ : the set of all nodes that have not constructed any 2C-SPs  
**Output:**  $P_{final}$ : all constructed 2C-SPs

```

1:  $P_{final} \leftarrow \emptyset$ 
2: while True do
3:    $n_1 \leftarrow |P_{final}|$ 
4:    $P' \leftarrow$  get 2C-SPs from  $N$  by the method in Bortfeldt and Yi [5]
5:    $P_{final} \leftarrow P_{final} \cup P'$ 
6:    $n_2 \leftarrow |P_{final}|$ 
7:   if  $n_2 == n_1$  then break else update  $N$  end if
8: end while

9: return  $P_{final}$ 

```

the effective swap operations identified during the route-pair swap process to further improve solution quality. The synergistic use of these three operators enables a more efficient search, enhancing both algorithm performance and solution quality. Detailed descriptions of these three operators are as follows.

**LKH mutation:** Following the decoding and evaluation of the parent giant tour (i.e., parent individual), a set of vehicle routes are obtained, with each route that can be regarded as an independent TSP problem. The TSP is a classical combinatorial optimization problem, and research progress has led to advanced TSP-solving algorithms. The LKH algorithm ([28]) is among the most effective and cutting-edge heuristic algorithms for solving TSP. The LKH mutation process independently optimizes each route derived from a giant tour using LKH algorithm and subsequently generates an optimized route set.

**Route-pair swap (RPS):** Algorithm 4 demonstrates the route-pair swap process, which swaps nodes between two different routes in  $R$  (Steps 2–10). Specifically, for each route  $r_1$  in  $R$ , a random route  $r_2$  is chosen from the remaining routes (Step 3). Then, for each node in  $r_1$ , a random node  $n_2$  is selected from  $r_2$  to swap, yielding two new routes,  $r'_1$  and  $r'_2$  (Steps 5–6). A giant tour  $g$  is created as a neighbor based

on  $r'_1$ ,  $r'_2$ , and the remaining routes (Steps 7–8). Finally, a neighbourhood  $N_{swap}$  is obtained.

---

**Algorithm 4** Route-pair swap
 

---

**Input:** a set of routes  $R$   
**Output:** neighbourhood  $N_{swap}$

```

1:  $N_{swap} \leftarrow \emptyset$ 
2: for  $r_1$  in  $R$  do
3:    $r_2 \leftarrow$  random select one route from the remaining routes
4:   for node  $n_1$  in  $r_1$  do
5:      $n_2 \leftarrow$  random select a node from  $r_2$ 
6:      $r'_1, r'_2 \leftarrow$  swap  $n_1$  and  $n_2$  and get two new routes
7:      $g \leftarrow$  get a giant tour based on  $r'_1, r'_2$  and remaining routes in
        $R$ 
8:      $N_{swap} \leftarrow N_{swap} \cup \{g\}$ 
9:   end for
10: end for
11: return  $N_{swap}$ 

```

---

**Multi-pair elitist recombination (MPER):** This operator is designed to be used in conjunction with RPS operator and is applied after it. As shown in Algorithm 5, after evaluating individuals in  $N_{swap}$ , those that outperform the parent individual (i.e., the individual before RPS operator) are selected to form a set  $S_{good}$  (Step 2). Subsequently, the algorithm selects as many mutually “compatible” individuals from  $S_{good}$  as possible to create  $S_{compat}$  (Step 4). The swaps in the  $S_{compat}$  individuals are then combined to form a new giant tour,  $g$ , as an individual in  $N_{MPER}$  (Steps 8–9), and the individuals in  $S_{compat}$  are removed from  $S_{good}$  (Step 10). The resulting neighbourhood for this operator is  $N_{MPER}$ .

---

**Algorithm 5** Multi-pair elitist recombination
 

---

**Input:** neighbourhood obtained by route-swap operator  $N_{swap}$   
**Output:** neighbourhood  $N_{MPER}$

```

1:  $N_{MPER} \leftarrow \emptyset$ 
2:  $S_{good} \leftarrow$  select individuals better than the parent individual from
    $N_{swap}$ 
3: while  $S_{good} \neq \emptyset$  do
4:    $S_{compat} \leftarrow$  select individuals that are compatible with each other
     from  $S_{good}$ 
5:   if  $S_{compat} = \emptyset$  then
6:     break
7:   end if
8:    $g \leftarrow$  obtain a giant tour by combining the swap of each individual
     in  $S_{compat}$ 
9:    $N_{MPER} \leftarrow N_{MPER} \cup \{g\}$ 
10:   $S_{good} \leftarrow S_{good} \setminus S_{compat}$ 
11: end while
12: return  $N_{MPER}$ 

```

---

It is important to define the concept of “compatible” individuals. Individuals are compatible if their swaps do not interfere. This means nodes involved in different swaps are not adjacent and avoids disrupting new edges created by a swap with subsequent swap operations. As illustrated in Fig.

4, the two swaps in (a) and (b) are compatible, whereas those in (c) are incompatible. This is because when the swaps in (c) are combined, nodes 3 and 1 become adjacent, indicating that one new edge has been disrupted.

Additionally, while RPS and MPER operators alter node positions across routes, there is no need to assess the impact on the packing feasibility of each route. This is because the adjusted routes are ultimately merged back into a giant tour, with a feasible solution obtained through giant tour decoding in the subsequent search process.

### 3.4 Adaptive splitting strategy

The 3L-SDVRP incorporates split delivery, a crucial feature that allows boxes from a single node to be loaded onto multiple vehicles. This flexibility significantly influences solution quality, as split decisions directly impact vehicle routing and box loading arrangements.

Bortfeldt and Yi [5] proposed a method where split delivery decisions are integrated into the giant tour decoding process. In this approach, vehicles sequentially visit nodes in the giant tour from front to back. Let  $n_{curr}$  represent the current visiting node,  $n_{next}$  the subsequent node, and 0 denote the depot. The distance between nodes  $i$  and  $j$  is given by  $d_{i,j}$ . When a vehicle cannot accommodate all box layers at  $n_{next}$ , it will still visit  $n_{next}$  and load as many layers as possible if the following condition is met:  $(d_{n_{curr}, n_{next}} + d_{n_{next}, 0}) / d_{n_{curr}, 0} \leq \max\_split\_costs$ . Here,  $\max\_split\_costs$  is a predefined hyperparameter. This method sets multiple  $\max\_split\_costs$  values as the outer loop and performs local searches for each value. However, this approach has two significant limitations:

- It fails to consider the remaining capacity of the vehicle and the box layer configuration at  $n_{next}$ . For instance, if the vehicle remaining capacity can only accommodate a minimal number of box layers, visiting  $n_{next}$  may be inefficient, as the incurred costs could outweigh the potential benefits.
- Using multiple  $\max\_split\_costs$  values requires separate local searches for each value, increasing computational time and complexity.

To address these shortcomings, we propose an adaptive splitting strategy. As illustrated in Algorithm 6, splitting delivery at node  $i$  is executed only when the ratio of  $l_i^v$  to  $l_i$  exceeds a specific threshold  $p$  (Steps 5–7). This strategy adaptively determines whether to perform splitting delivery based on the information of the vehicle and node. Additionally, it eliminates the need to traverse  $\max\_split\_costs$ , thereby conserving computational resources and enhancing efficiency. Therefore, our proposed approach:

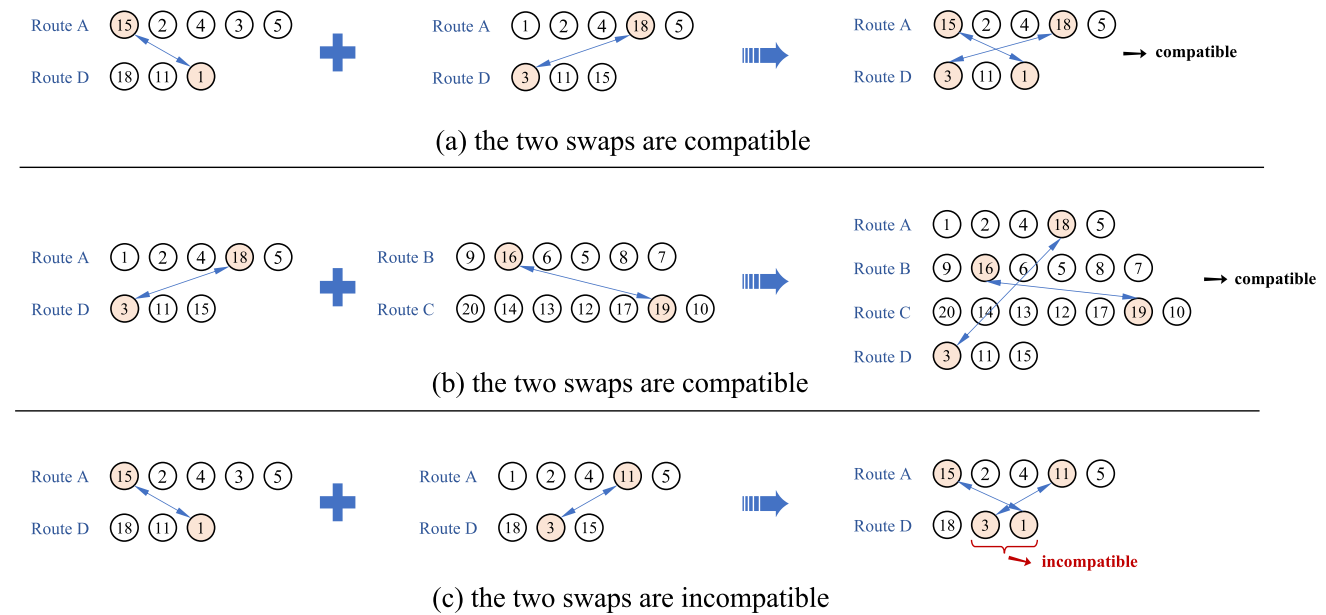


**Fig. 4** Compatible and incompatible swaps

- Dynamically determines whether to visit the next node based on the vehicle remaining capacity and the box layer configuration at that node.
- Eliminates the need for multiple local searches, thereby reducing computational overhead.

#### Algorithm 6 Adaptive splitting strategy

**Input:** vehicle  $v$ , node  $i$

**Output:**  $b_{splitting}$  (True: splitting; False: not splitting)

```

1:  $l_v \leftarrow$  the residual length of  $v$ 
2:  $l_i \leftarrow$  the sum of depths of all layers in  $i$ 
3:  $L_i^v \leftarrow$  select as many layers loaded into  $v$  as possible from  $i$  based on  $l_v$ 
4:  $l_i^v \leftarrow$  the sum of depths of layers in  $L_i^v$ 
5: if  $l_i^v / l_i < p$  then  $b_{splitting} \leftarrow False$ 
6: else  $b_{splitting} \leftarrow True$ 
7: end if
8: return  $b_{splitting}$ 

```

### 3.5 New post-optimization approach

In SDVRLH2, a 3-opt operator is employed for post-optimization. As a commonly used search operator in VRPs, the number of neighbourhood solutions generated by the 3-opt increases exponentially with the problem size ([3]), resulting in high computational complexity. Moreover, due to the lack of heuristic information utilization, the effectiveness of employing 3-opt for post-optimization in this problem is not ideal. Therefore, this paper designs a new

post-optimization method to further improve solution quality.

The proposed post-optimization method consists of two main components: reassignment and routing optimization. The former aims to reassign layers loaded on a specific vehicle to other vehicles to reduce the vehicle count, while the latter employs our LKH mutation operator to independently optimize each route to minimize the total travel distance.

**Reassignment.** The overall process of reassignment is demonstrated in Algorithm 7. All routes in the set  $R$  are sorted in ascending order based on the sum of their loaded layers' depths (Step 1). The reassignment is prioritized for routes with smaller loadings (Steps 2–12) since they have a higher likelihood of successful reassignment. Specifically, after determining the route  $r_1$  that requires reassignment, the remaining routes in  $R$  are sorted in descending order according to the residual length of the vehicles (Steps 3–4). Then, all layers in  $r_1$  are attempted to be reassigned to the routes in  $R$  (Step 5). If all layers in  $r_1$  are successfully reassigned,  $R$  is updated and its routes are re-sorted based on the sum of their loaded layers' depths (Steps 7–8), followed by the next iteration; otherwise, the reassignment process stops, and the algorithm ends (Step 10).

As shown in Algorithm 8, after identifying route  $r_a$  for reassignment, the algorithm reallocates its layers to other vehicles in  $R$ . This process involves calculating the maximum available space ( $rl_{max}$ ) for each route (Step 1) and sorting layers of  $r_a$  by depth in  $L_{r_a}$  (Step 2). The algorithm tries to reassign layers in  $L_{r_a}$  to suitable vehicles in  $R$ , marking success if all layers are reassigned (Steps 4–28). For each layer  $\ell$  in  $L_{r_a}$ , it seeks a new route  $r$ , assessing the needed

**Algorithm 7** Framework of Reassignment

---

**Input:** a set of routes  $R$  with loaded layers  
**Output:**  $R$  after reassign

```

1:  $R \leftarrow$  sort  $R$  in ascending order based on the sum of the depths of
   the layers each route carries
2: for route  $r_1 \in R$  do
3:    $R \leftarrow R \setminus \{r_1\}$ 
4:    $R \leftarrow$  sort  $R$  in descending order based on the residual length of
   each vehicle
5:    $b_{succ}, R_{reassign} \leftarrow$  reassign layers in  $r_1$  to remaining routes in  $R$ 
   by Algorithm 8
6:   if  $b_{succ}$  is True then
7:      $R \leftarrow R_{reassign}$ 
8:      $R \leftarrow$  sort  $R$  in ascending order based on the sum of the depths
   of the layers each route carries
9:   else
10:    break
11:   end if
12: end for
13: return  $R$ 

```

---

additional space ( $l_{need}$ , Step 8). If  $l_{need} \leq 0$ , the layer fits without adjustments. If  $l_{need} > rl_{max}$ ,  $\ell$  cannot be reassigned to  $r$ . For  $0 \leq l_{need} \leq rl_{max}$ , the algorithm might transfer layers from  $r$  to accommodate  $\ell$ , selecting the option that minimizes additional required space  $l_w$  (Steps 20–21). Failure to reassign any layer from  $L_{ra}$  indicates a reassignment failure, and the original  $R$  is retained; otherwise, the successful reassignment potentially reduces the number of needed vehicles.

**Single route optimization.** After the reassignment process, some layers on certain routes are redistributed to other routes, making it necessary to optimize the routes post-reassignment. In 3L-SDVRP, each customer node can only be visited once by the same vehicle. Consequently, each vehicle departs from the depot, visits some customer nodes, and returns to the depot. Given this, every vehicle route can be considered an independent TSP problem. Therefore, in this step, we employ our proposed LKH mutation operator to optimize the routes after reassignment, aiming to reduce the overall travel distance.

## 4 Computational studies

In this study, we have evaluated our algorithm against SDVRLH2 ([5]), which is the current state-of-the-art method for the 3L-SDVRP, across three widely-used datasets: B-Y ([5]), Shanghai ([5]), and SD instances ([7]). Instances are categorized based on the node count: those with fewer than 100 nodes are considered small-scale, and those with 100 to 200 nodes are considered large-scale. The comparative analysis is explained in Sect. 4.2. Additionally, Sect. 4.3 delves into further analyses. It examines how different objectives influence outcomes, assesses the novel features of our algo-

**Algorithm 8** Reassign layers in a route to other routes

---

**Input:**  $r_a$ : route to be reassigned with loaded layers;  $R$ : routes receiving layers from  $r_a$   
**Output:**  $b_{succ}$ : a Boolean variable used to indicate the success of the reassignment;  $R$ : other routes after reassignment

```

1:  $rl_{max} \leftarrow$  the maximal residual length of vehicles corresponding to
    $R$ 
2:  $L_{ra} \leftarrow$  sort layers of  $r_a$  in descending order based on layer depths
3:  $b_{succ} \leftarrow \text{True}$ 
4: for each layer  $\ell \in L_{ra}$  do
5:    $l_\ell \leftarrow$  the depth of  $\ell$ 
6:    $l_{w\_min} \leftarrow$  a big positive number;  $s \leftarrow \emptyset$ 
7:   for route  $r \in R$  do
8:      $l_v \leftarrow$  residual length of the vehicle for route  $r$ ;  $l_{need} \leftarrow l_\ell - l_v$ 
9:     if  $l_{need} \leq 0$  then
10:        $l_{w1} \leftarrow -l_{need}$ ;  $l_{w2} \leftarrow 0$ 
11:       else if  $l_{need} > rl_{max}$  then
12:         break
13:       else
14:          $\mathcal{L}_{remove} \leftarrow$  layers to be removed from  $r$ 
15:          $l_{re} \leftarrow$  the depth sum of  $\mathcal{L}_{remove}$ 
16:         if  $l_{re} \leq rl_{max}$  then
17:            $l_{w1} \leftarrow l_{re} - l_{need}$ ;  $l_{w2} \leftarrow$  move  $\mathcal{L}_{remove}$  to the most
           suitable route  $r'$  and update its vehicle residual length
18:           end if
19:           end if
20:            $l_w \leftarrow l_{w1} + l_{w2}$ 
21:           if  $l_w < l_{w\_min}$  then  $l_{w\_min} \leftarrow l_w$ ;  $s \leftarrow$  reassignment arrange-
           ment of  $\ell$  end if
22:         end for
23:         if  $s = \emptyset$  then
24:            $b_{succ} \leftarrow \text{False}$ ; return  $b_{succ}$ , original  $R$ 
25:         else
26:            $r_a, R \leftarrow$  perform reassignment according to  $s$  and update  $r_a$ 
           and  $R$ 
27:         end if
28:       end for
29: return  $b_{succ}, R$ 

```

---

rithm through ablation studies, and conducts a time analysis. For improved readability, this paper presents comparative evaluation data across different algorithms in the main text. Detailed experimental results are available in Appendix D.

### 4.1 Experimental setting

In line with the SDVRLH2, we prioritize the number of vehicles as the primary objective and total travel distance ( $ttd$ ) as the secondary objective (with lower priority). Therefore, when evaluating the quality of two solutions, we first examine the number of vehicles, with fewer vehicles indicating a superior solution. When the number of vehicles is equal, we then compare the  $ttd$ , with shorter distances representing better solutions. The parameter settings are as follows: in Algorithm 1,  $n_{out} = 2$ ,  $n_{iter} = 100$ , and  $n_{no\_imp} = 2$ ; in Algorithm 2,  $\alpha_1 = 0.5$  and  $\alpha_2 = 0$ . It is worth noting that we did not perform a rigorous parameter tuning process because we want to show that the strong performance of our algorithm originates from novel algorithm designs, not from

**Table 3** Comparison results between SD and Ov on small-scale instances (number of nodes <100)

| Instances (small) | Avg # v<br>Ov-SD | Avg ttd<br>(Ov-SD)/SD | Best # v<br>Ov-SD | Best ttd<br>(Ov-SD)/SD | CPU Time<br>Ov/SD | # FE<br>Ov/SD |
|-------------------|------------------|-----------------------|-------------------|------------------------|-------------------|---------------|
| B-Y1              | <u>-1.1</u>      | 0.2780                | -2                | 0.2141                 | 0.1264            | 0.0018        |
| B-Y2              | <u>-0.5</u>      | 0.4953                | 0                 | 0.4354                 | 0.0882            | 0.0019        |
| B-Y3              | <u>-1.0</u>      | 0.2007                | -1                | 0.1249                 | 0.1720            | 0.0027        |
| B-Y4              | <u>1.3</u>       | 0.5466                | 1                 | 0.4916                 | 0.1345            | 0.0019        |
| B-Y5              | <u>-2</u>        | 0.3155                | -2                | 0.2752                 | 0.0627            | 0.0017        |
| B-Y6              | <u>-0.3</u>      | 0.5965                | -1                | 0.5636                 | 0.0443            | 0.0018        |
| B-Y7              | <u>-2</u>        | 0.2363                | -2                | 0.1897                 | 0.0710            | 0.0016        |
| B-Y8              | <u>1.0</u>       | 0.7038                | 1                 | 0.6638                 | 0.0514            | 0.0016        |
| Sha1              | 0                | 0.0238                | 0                 | 0.0135                 | 2.0054            | 0.0210        |
| Sha2              | 0.2              | 0.0456                | 0                 | 0.0239                 | 0.0472            | 0.0126        |
| Sha3              | 0                | 0.0534                | 0                 | 0.0132                 | 0.4951            | 0.0050        |
| Sha4              | 0                | 0.1938                | 0                 | 0.0718                 | 0.2729            | 0.0051        |
| Sha5              | <u>-1</u>        | <u>-0.0636</u>        | -1                | <b>-0.0817</b>         | 0.1035            | 0.0083        |
| Sha6              | <u>-1</u>        | 0.3199                | -1                | 0.2425                 | 0.0867            | 0.0035        |
| Sha7              | <u>-0.7</u>      | 0.0600                | -1                | <b>-0.0071</b>         | 0.0967            | 0.0038        |
| Sha8              | 0                | 0.0494                | 0                 | 0.0238                 | 0.2648            | 0.0070        |
| Sha9              | 0                | 0.2128                | 0                 | 0.1463                 | 0.1394            | 0.0027        |
| Sha10             | <u>0</u>         | 0.1544                | <u>0</u>          | 0.0993                 | 0.1331            | 0.0034        |
| Sha11             | <u>-0.2</u>      | 0.2208                | 0                 | 0.1735                 | 0.0376            | 0.0013        |
| Sha12             | <u>-2</u>        | 0.2075                | -2                | 0.1535                 | 0.0641            | 0.0020        |
| Sha13             | <u>-0.7</u>      | 0.2785                | <u>0</u>          | 0.2510                 | 0.0182            | 0.0015        |
| SD1               | 0                | 0.1749                | 0                 | 0.1563                 | 1.1004            | 0.0079        |
| SD2               | <u>-1</u>        | 0.2033                | -1                | 0.1360                 | 0.3453            | 0.0036        |
| SD3               | 0                | 0.1501                | 0                 | 0.1409                 | 0.0888            | 0.0034        |
| SD4               | <u>-1</u>        | 0.4285                | -1                | 0.3341                 | 0.1273            | 0.0026        |
| SD5               | <u>2</u>         | 0.5586                | 2                 | 0.5486                 | 0.0618            | 0.0014        |
| SD6               | <u>-2</u>        | 0.4721                | -2                | 0.4322                 | 0.5147            | 0.0021        |
| SD7               | <u>-2.5</u>      | 0.4053                | -3                | 0.3213                 | 0.0558            | 0.0023        |
| SD8               | <u>-3</u>        | 0.2882                | -3                | 0.2431                 | 0.2041            | 0.0019        |
| SD9               | <u>-3</u>        | 0.5665                | -3                | 0.4841                 | 0.0568            | 0.0019        |
| SD10              | <u>-9</u>        | 0.3239                | -9                | 0.2461                 | 0.0397            | 0.0019        |
| SD11              | <u>-11.1</u>     | 0.6625                | -11               | 0.5747                 | 0.0238            | 0.0010        |
| Avg               | <u>-1.3</u>      | 0.3779                | <u>-1.3</u>       | 0.3224                 | <b>0.0599</b>     | 0.0018        |
| Total             | <u>-40.6</u>     | 0.3779                | <u>-42.0</u>      | 0.3224                 | <b>0.0599</b>     | 0.0018        |

SD refers to the SDVRLH2 algorithm; Ov refers to our algorithm with # v as the first objective. # v = the number of vehicles used; ttd=total travel distance; # FE= the number of fitness evaluations. The data in bold signifies that our strategy produces better results than the contrasted strategies numerically. The Mann-Whitney U test was employed, and the data with underlined values indicate a significant difference at the significance level  $\alpha = 0.05$

finely tuned parameters. The experiments were conducted 30 times using Python 3.7 on a Dell R370 server with 2x Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz, 128G RAM, and the CentOS 7.6 operating system.

## 4.2 Comparative analysis

The experimental results of our proposed algorithm and SDVRLH2 ([5]) are presented in Tables 3 and 4. The exper-

imental data for SDVRLH2 originates from Bortfeldt and Yi [5], and the calculation method for the number of FEs is provided in Appendix C.

*Small-scale instances.* Table 3 shows that our algorithm outperforms SDVRLH2 in the average number of vehicles used, achieving better outcomes on 21 instances and similar results on 9 instances. Our proposed algorithm outperforms SDVRLH2 in both average number of vehicles and average *ttd* on instance Sha5, establishing new records for the low-

**Table 4** Comparison results between SD and Ov on large-scale instances ( $100 \leq \text{number of node} \leq 200$ )

| Instances (large) | Avg # v<br>Ov-SD | Avg ttd<br>(Ov-SD)/SD | Best # v<br>Ov-SD | Best ttd<br>(Ov-SD)/SD | CPU Time<br>Ov/SD | # FE<br>Ov/SD |
|-------------------|------------------|-----------------------|-------------------|------------------------|-------------------|---------------|
| B-Y9              | <b>-4</b>        | <u>0.4413</u>         | <b>-4</b>         | 0.3581                 | 0.2627            | 0.0015        |
| B-Y10             | <b>-3</b>        | <u>0.6334</u>         | <b>-3</b>         | 0.5385                 | 0.1325            | 0.0016        |
| B-Y11             | <b>-4.1</b>      | <u>0.3295</u>         | <b>-5</b>         | 0.2523                 | 0.3184            | 0.0013        |
| B-Y12             | <u>1.0</u>       | <u>0.8921</u>         | 1                 | 0.8323                 | 0.1066            | 0.0014        |
| B-Y13             | <b>-7.1</b>      | <u>0.4152</u>         | <b>-8</b>         | 0.3443                 | 0.5825            | 0.0012        |
| B-Y14             | <b>-2</b>        | <u>0.6790</u>         | <b>-2</b>         | 0.6287                 | 0.1389            | 0.0013        |
| B-Y15             | <b>-10.5</b>     | <u>0.3515</u>         | <b>-10</b>        | 0.2958                 | 0.4684            | 0.0011        |
| B-Y16             | <u>1.1</u>       | 0.8698                | 2                 | 0.7213                 | 0.4735            | 0.0011        |
| B-Y17             | <b>-12.4</b>     | <u>0.3098</u>         | <b>-13</b>        | 0.2400                 | 0.7362            | 0.0008        |
| B-Y18             | <b>-6.6</b>      | <u>0.8257</u>         | <b>-6</b>         | 0.7099                 | 0.4009            | 0.0009        |
| B-Y19             | <b>-14.1</b>     | <u>0.2064</u>         | <b>-15</b>        | 0.1631                 | 1.3106            | 0.0008        |
| B-Y20             | <b>-0.9</b>      | <u>0.8021</u>         | 0                 | 0.6925                 | 0.5849            | 0.0009        |
| Sha14             | <b>-2.2</b>      | <u>0.6836</u>         | <b>-2</b>         | 0.6144                 | 0.0922            | 0.0007        |
| Sha15             | <b>-5.0</b>      | <u>0.9501</u>         | <b>-5</b>         | 0.8700                 | 0.3990            | 0.0005        |
| SD12              | 0                | <u>0.3429</u>         | 0                 | 0.2946                 | 0.1662            | 0.0015        |
| SD13              | <b>-3</b>        | <u>1.0800</u>         | <b>-3</b>         | 1.0135                 | 0.8826            | 0.0011        |
| Avg               | <b>-4.5</b>      | 0.5645                | <b>-4.6</b>       | 0.4989                 | 0.4391            | 0.0009        |
| Total             | <b>-72.6</b>     | 0.5645                | <b>-73.0</b>      | 0.4989                 | 0.4391            | 0.0009        |

SD refers to the SDVRLH2 algorithm; Ov refers to our algorithm with # v as the first objective. # v = the number of vehicles used; ttd = total travel distance; # FE = the number of fitness evaluations. The data in bold signifies that our strategy produces better results than the contrasted strategies numerically. The Mann-Whitney U test was employed, and the data with underlined values indicate a significant difference at the significance level  $\alpha = 0.05$

est number of vehicles on 17 instances and the smallest *ttd* on 2 instances. Notably, our proposed algorithm achieves a maximum reduction of 11 vehicles on instance SD11 and a 6.36% reduction in *ttd* on instance Sha5, contributing to a total reduction of 40.6 vehicles across all instances. Over multiple runs, our algorithm achieves new best-known solutions for the number of vehicles on 17 instances and for both objectives on 2 instances. Furthermore, our proposed algorithm requires significantly less CPU time and fewer fitness evaluations (FEs). Across 32 small-scale instances, our algorithm requires, on average, merely 0.18% of the FEs per instance compared to those required by SDVRLH2.

*Large-scale instances.* As shown in Table 4, when applied to 16 larger-scale problem instances, our algorithm significantly reduces the average number of vehicles required on 13 instances, achieving a maximum reduction of 15 vehicles compared to SDVRLH2 on instance B-Y19. There is a total reduction of 72.6 vehicles across all instances, with an average of 4.5 vehicles per instance. Based on the best results across multiple runs, the proposed algorithm identifies new best-known solutions on 12 problem instances in terms of the number of vehicles utilized. The total number of vehicles saved across all large-scale problem instances reaches 73, equivalent to an average reduction of 4.6 vehicles per instance. The algorithm also establishes new records

by achieving the lowest number of vehicles on 12 instances. Compared to SDVRLH2, our algorithm requires significantly fewer FEs and is therefore computationally more efficient. On all 16 instances, the number of FEs required by our algorithm does not exceed 0.2% of those used by SDVRLH2.

However, our proposed algorithm does increase the *ttd*, particularly in larger-scale problems where the substantial reduction in vehicles leads to longer travel distances. On small-scale instances, the average *ttd* increases by 37.79% compared to SDVRLH2. On large-scale instances, the *ttd* rises by 56.45%. This can be attributed to three factors: (1) The number of vehicles and *ttd* represent conflicting objectives: an improvement in one leads to a decline in the other. (2) Our method constructs more 2C-SPs, thereby saving vehicle space and reducing the number of vehicles. However, the 2C-SP may be maintained during the route search process if it helps reduce the number of vehicles used, which could potentially increase the *ttd*. (3) The algorithm uses few FEs in the route search process. The *ttd* could be further reduced by allocating additional computational resources to routing. These are areas for our future studies.

**Table 5** Comparison results between Ov and Ot on small-scale instances (number of nodes <100)

| Instances (small) | Avg # v<br>Ot-Ov | Avg ttd<br>(Ot-Ov)/Ov | Best # v<br>Ot-Ov | Best ttd<br>(Ot-Ov)/Ov | CPU Time<br>Ot/Ov | # FE<br>Ot/Ov |
|-------------------|------------------|-----------------------|-------------------|------------------------|-------------------|---------------|
| B-Y1              | 0.1              | <b>-0.0197</b>        | 1                 | 0                      | 1.353             | 1.363         |
| B-Y2              | 0.0              | <b>-0.0215</b>        | 0                 | 0                      | 1.524             | 1.417         |
| B-Y3              | <u>1.0</u>       | <b>-0.0151</b>        | 1                 | 0                      | 1.138             | 1.167         |
| B-Y4              | 0.7              | <b>-0.0400</b>        | 1                 | 0                      | 1.535             | 1.374         |
| B-Y5              | 0.8              | <b>-0.0068</b>        | 0                 | 0                      | 1.268             | 1.338         |
| B-Y6              | 0.3              | <b>-0.0266</b>        | 1                 | 0                      | 1.392             | 1.263         |
| B-Y7              | 0.2              | <b>-0.0095</b>        | 0                 | 0                      | 1.320             | 1.488         |
| B-Y8              | 0.9              | <b>-0.0198</b>        | 0                 | 0                      | 1.325             | 1.240         |
| Sha1              | 0.0              | <b>-0.0066</b>        | 0                 | 0                      | 1.594             | 1.730         |
| Sha2              | <b>-0.1</b>      | <b>-0.0084</b>        | 0                 | 0                      | 1.846             | 1.611         |
| Sha3              | 0.0              | <b>-0.0126</b>        | 0                 | 0                      | 1.304             | 1.460         |
| Sha4              | 0.0              | 0.0036                | 0                 | 0                      | 1.846             | 1.646         |
| Sha5              | 0.0              | <b>-0.0012</b>        | 0                 | 0                      | 1.778             | 1.583         |
| Sha6              | 0.0              | <b>-0.0134</b>        | 0                 | 0                      | 1.618             | 1.601         |
| Sha7              | 0.3              | <b>-0.0114</b>        | 0                 | 0                      | 1.341             | 1.556         |
| Sha8              | 0.0              | <b>-0.0105</b>        | 0                 | 0                      | 1.475             | 1.508         |
| Sha9              | 0.0              | <b>-0.0265</b>        | 0                 | 0                      | 1.492             | 1.582         |
| Sha10             | 0.0              | <b>-0.0150</b>        | 0                 | 0                      | 1.151             | 1.472         |
| Sha11             | 0.0              | <b>-0.0153</b>        | 0                 | 0                      | 1.222             | 1.517         |
| Sha12             | 0.1              | <b>-0.0063</b>        | 0                 | 0                      | 1.280             | 1.382         |
| Sha13             | 0.6              | <b>-0.0098</b>        | 0                 | 0                      | 0.940             | 1.306         |
| SD1               | 0.0              | <b>-0.0061</b>        | 0                 | 0                      | 1.756             | 1.403         |
| SD2               | 0.0              | <b>-0.0319</b>        | 0                 | 0                      | 1.837             | 1.589         |
| SD3               | 0.0              | <b>-0.0220</b>        | 0                 | 0                      | 1.507             | 1.327         |
| SD4               | 0.2              | <b>-0.0041</b>        | 0                 | 0                      | 1.616             | 1.433         |
| SD5               | 0.0              | <b>-0.0021</b>        | 0                 | 0                      | 1.857             | 1.628         |
| SD6               | 0.0              | <b>-0.0254</b>        | 0                 | 0                      | 1.370             | 1.452         |
| SD7               | <u>0.5</u>       | <b>-0.0520</b>        | 1                 | 0                      | 1.449             | 1.362         |
| SD8               | 0.0              | <b>-0.0108</b>        | 0                 | 0                      | 1.246             | 1.316         |
| SD9               | 0.3              | <b>-0.0263</b>        | 0                 | 0                      | 1.388             | 1.496         |
| SD10              | 0.6              | <b>-0.0473</b>        | 0                 | 0                      | 1.447             | 1.430         |
| SD11              | 0.0              | <b>-0.0249</b>        | 0                 | 0                      | 1.251             | 1.373         |
| Avg               | 0.2              | <b>-0.0220</b>        | 0.16              | 0                      | 1.322             | 1.379         |
| Total             | 6.5              | <b>-0.0220</b>        | 5                 | 0                      | 1.322             | 1.379         |

Ov (Ot) refers to our algorithm with # v (ttd) as the first objective. # v = the number of vehicles used; ttd=total travel distance; # FE= the number of fitness evaluations. The data in bold signifies that our strategy produces better results than the contrasted strategies numerically. The Mann-Whitney U test was employed, and the data with underlined values indicate a significant difference at the significance level  $\alpha = 0.05$

### 4.3 Further analysis

This section explores three aspects of our algorithm: (1) the effects of changing the primary objective on performance, (2) the contributions of each novel component through ablation studies, and (3) the efficiency of our algorithm through time analysis. Our experiments use a fixed iteration count, setting a consistent limit on fitness evaluations (FEs). However, Algorithm 1 can terminate early if no improvement is

observed after  $n_{no\_imp}$  consecutive iterations, suggesting a local optimum has been reached. Hence, the actual FEs used may vary across different algorithm configurations, reflecting the distinct search effectiveness of each setting.

#### 4.3.1 Comparative study of primary objectives

In real-world scenarios, reducing the number of vehicles holds greater importance than minimizing total travel dis-

**Table 6** Comparison results between Ov and Ot on large-scale instances ( $100 \leq \text{number of node} \leq 200$ )

| Instance (large) | Avg # v<br>Ot-Ov | Avg ttd<br>(Ot-Ov)/Ov | Best # v<br>Ot-Ov | Best ttd<br>(Ot-Ov)/Ov | CPU Time<br>Ot/Ov | # FE<br>Ot/Ov |
|------------------|------------------|-----------------------|-------------------|------------------------|-------------------|---------------|
| B-Y9             | <u>2.0</u>       | <b>-0.1048</b>        | 2                 | 0                      | 0.180             | 1.232         |
| B-Y10            | <u>1.3</u>       | <b>-0.0744</b>        | 2                 | 0                      | 0.188             | 1.077         |
| B-Y11            | <u>1.7</u>       | <b>-0.0809</b>        | 2                 | 0                      | 0.199             | 1.381         |
| B-Y12            | <u>1.9</u>       | <b>-0.0726</b>        | 1                 | 0                      | 0.322             | 1.176         |
| B-Y13            | <u>2.9</u>       | <b>-0.1026</b>        | 3                 | 0                      | 0.203             | 1.132         |
| B-Y14            | <u>2.0</u>       | <b>-0.0829</b>        | 2                 | 0                      | 0.555             | 1.212         |
| B-Y15            | <u>3.1</u>       | <b>-0.0996</b>        | 3                 | 0                      | 0.364             | 1.419         |
| B-Y16            | <u>2.5</u>       | <b>-0.1043</b>        | 2                 | 0                      | 0.186             | 1.334         |
| B-Y17            | <u>3.6</u>       | <b>-0.0978</b>        | 4                 | 0                      | 0.371             | 1.330         |
| B-Y18            | <u>2.4</u>       | <b>-0.1051</b>        | 2                 | 0                      | 0.390             | 1.247         |
| B-Y19            | <u>4.2</u>       | <b>-0.1067</b>        | 5                 | 0                      | 0.227             | 1.279         |
| B-Y20            | <u>3.2</u>       | <b>-0.1059</b>        | 3                 | 0                      | 0.347             | 1.225         |
| Sha14            | <u>1.0</u>       | <b>-0.0857</b>        | 1                 | 0                      | 0.128             | 1.481         |
| Sha15            | <u>1.1</u>       | <b>-0.0776</b>        | 2                 | 0                      | 0.078             | 1.269         |
| SD12             | 0.0              | <b>-0.0286</b>        | 0                 | 0                      | 0.186             | 1.301         |
| SD13             | 0.0              | <b>-0.0242</b>        | 0                 | 0                      | 0.051             | 1.325         |
| Avg              | 2.0              | <b>-0.0610</b>        | 2.13              | 0                      | 0.229             | 1.271         |
| Total            | 32.7             | <b>-0.0610</b>        | 34                | 0                      | 0.229             | 1.271         |

Ov (Ot) refers to our algorithm with # v (ttd) as the first objective. # v = the number of vehicles used; ttd = total travel distance; # FE = the number of fitness evaluations. The data in bold signifies that our strategy produces better results than the contrasted strategies numerically. The Mann-Whitney U test was employed, and the data with underlined values indicate a significant difference at the significance level  $\alpha = 0.05$

tance ([5]), due to the significantly higher costs associated with acquiring, maintaining, and employing drivers for additional vehicles compared to the costs incurred from increased travel distances. Our algorithm, aligned with SDVRLH2, aims to minimize the number of vehicles as the primary objective, with the total travel distance (*ttd*) as the secondary objective. To assess how the objective prioritizations affect algorithmic performance, we have performed experiments comparing the algorithms with the vehicle number (Ov) and *ttd* (Ot) as the primary objective respectively. Tables 5 and 6 present these results. Note that in the algorithm process, solution evaluations are initially based on the primary objective. When solutions have the same primary objective value, the secondary objective then distinguishes their quality.

*Small-scale instances.* As shown in Table 5, on 32 small-scale problem instances, in terms of average vehicle counts, Ot underperformed Ov on 10 instances but matched Ov on the remaining ones. Ot utilized 6.5 more vehicles in total compared to Ov. Concerning average *ttd*, Ot outperformed Ov on 11 instances, achieving reductions in *ttd* of up to 5.2%, with an average decrease of 2.2% per instance. Upon analyzing the best results from multiple runs, Ot used more vehicles than Ov on 5 instances but matched Ov in *ttd* across all instances.

*Large-scale instance.* According to Table 6, on the 16 large-scale instances, the gap between Ot and Ov widened

considerably. Ot needed more vehicles than Ov, with a notable increase on 14 instances, leading to 32.7 more vehicles in total. However, Ot significantly reduced the *ttd*, outperforming Ov by more than 7% on 14 instances and averaging a 6.1% *ttd* reduction. When looking at the best outcomes from multiple runs, Ot underperformed Ov in terms of vehicle count, but matched the *ttd* achievements of Ov.

There is a clear conflict between vehicle count and *ttd* when comparing Ov and Ot. Prioritizing vehicle count generally results in fewer vehicles used, while prioritizing *ttd* achieves shorter total distances, particularly in larger-scale problems. Whether focusing on minimizing vehicle count (Ov) or reducing *ttd* (Ot), it is challenging to perform well in both objectives simultaneously. From a Pareto dominance aspect, the results emphasize the inherent difficulty of achieving a balanced optimization between these objectives within a single-objective optimization framework. Thus, our work highlights the complexity of simultaneously optimizing vehicle number and *ttd*, presenting a challenging task in the search for improved overall solutions.

#### 4.3.2 Ablation study

To evaluate the effectiveness of each innovation in our approach, a comprehensive series of ablation experiments was conducted. Table 7 presents the algorithm notations

**Table 7** Notions for algorithms with different components in ablation experiments

| Notation | Algorithm with X          | Notation | Algorithm with X                 |
|----------|---------------------------|----------|----------------------------------|
| A1       | X=Improved packing method | A3       | X=A2+Adaptive splitting strategy |
| A2       | X=A1+New search operators | A4       | X=A3+New post-optimization       |

**Table 8** Ablation experimental results (A1 vs. SD) on small-scale instances (number of nodes <100)

| Instances (small) | Avg # v<br>A1-SD | Avg ttd<br>(A1-SD)/SD | Best # v<br>A1-SD | Best ttd<br>(A1-SD)/SD | CPU time<br>A1/SD | # FE<br>A1/SD |
|-------------------|------------------|-----------------------|-------------------|------------------------|-------------------|---------------|
| B-Y1              | <b>-1.8</b>      | <u>0.2621</u>         | <b>-2</b>         | 0.2019                 | 53.25             | 9.3636        |
| B-Y2              | <b>-0.5</b>      | <u>0.3572</u>         | 0                 | 0.3217                 | 62.04             | 8.9687        |
| B-Y3              | <b>-1</b>        | <u>0.1718</u>         | <b>-1</b>         | 0.0933                 | 47.16             | 8.5743        |
| B-Y4              | <u>1.1</u>       | <u>0.5064</u>         | 1                 | 0.4446                 | 66.08             | 8.2154        |
| B-Y5              | <b>-2</b>        | <u>0.2356</u>         | <b>-2</b>         | 0.2149                 | 70.22             | 15.9788       |
| B-Y6              | <b>-1</b>        | <u>0.5302</u>         | <b>-1</b>         | 0.4308                 | 93.59             | 15.7573       |
| B-Y7              | <b>-2</b>        | <u>0.1720</u>         | <b>-2</b>         | 0.1484                 | 63.39             | 15.7287       |
| B-Y8              | <u>1</u>         | <u>0.7079</u>         | 1                 | 0.6746                 | 77.21             | 13.2315       |
| Sha1              | 0                | <u>0.0135</u>         | 0                 | 0.0135                 | 151.76            | 1.3735        |
| Sha2              | 0                | <u>0.0278</u>         | 0                 | 0.0239                 | 5.22              | 2.0096        |
| Sha3              | 0                | <u>0.0438</u>         | 0                 | 0.0433                 | 95.87             | 1.8017        |
| Sha4              | 0                | <u>0.0973</u>         | 0                 | 0.0718                 | 34.22             | 2.2397        |
| Sha5              | <b>-1</b>        | <b>-0.0708</b>        | <b>-1</b>         | <b>-0.0887</b>         | 8.32              | 3.1993        |
| Sha6              | <b>-1</b>        | <u>0.2664</u>         | <b>-1</b>         | 0.2412                 | 14.89             | 2.5766        |
| Sha7              | <b>-1</b>        | <u>0.0488</u>         | <b>-1</b>         | 0.0156                 | 12.80             | 2.8038        |
| Sha8              | 0                | <u>0.0196</u>         | 0                 | 0.0104                 | 18.46             | 5.1853        |
| Sha9              | 0                | <u>0.1404</u>         | 0                 | 0.1493                 | 21.29             | 5.1815        |
| Sha10             | 0                | <u>0.0770</u>         | 0                 | 0.0791                 | 23.14             | 7.4412        |
| Sha11             | <b>-0.2</b>      | <u>0.1209</u>         | 0                 | 0.1034                 | 17.85             | 6.2510        |
| Sha12             | <b>-2</b>        | <u>0.1333</u>         | <b>-2</b>         | 0.1232                 | 18.86             | 5.5049        |
| Sha13             | <b>-0.7</b>      | <u>0.2149</u>         | 0                 | 0.2106                 | 18.07             | 10.4225       |
| SD1               | 0                | <u>0.0844</u>         | 0                 | 0.0775                 | 16.86             | 2.4892        |
| SD2               | <b>-1</b>        | <u>0.1118</u>         | <b>-1</b>         | 0.0827                 | 27.02             | 4.6064        |
| SD3               | 0                | <u>0.0250</u>         | 0                 | 0.0194                 | 17.78             | 7.4335        |
| SD4               | <b>-1</b>        | <u>0.2706</u>         | <b>-1</b>         | 0.2649                 | 38.69             | 7.4225        |
| SD5               | <u>2</u>         | <u>0.5496</u>         | 2                 | 0.5480                 | 35.17             | 6.3895        |
| SD6               | <b>-2</b>        | <u>0.4225</u>         | <b>-2</b>         | 0.4001                 | 24.97             | 8.3223        |
| SD7               | <b>-3</b>        | <u>0.2508</u>         | <b>-3</b>         | 0.1689                 | 24.03             | 9.3373        |
| SD8               | <b>-3</b>        | <u>0.2652</u>         | <b>-3</b>         | 0.2435                 | 20.96             | 8.0568        |
| SD9               | <b>-3</b>        | <u>0.4292</u>         | <b>-3</b>         | 0.4260                 | 20.38             | 11.2968       |
| SD10              | <b>-9</b>        | <u>0.1323</u>         | <b>-9</b>         | 0.1089                 | 18.04             | 13.7202       |
| SD11              | <b>-11.1</b>     | <u>0.3448</u>         | <b>-11</b>        | 0.3372                 | 20.50             | 17.0319       |
| Avg               | <b>-1.4</b>      | 0.2176                | <b>-1.3</b>       | 0.1939                 | 43.32             | 12.1300       |
| Total             | <b>-43.2</b>     | 0.2644                | <b>-42</b>        | 0.2451                 | 43.32             | 12.1300       |

# v = the number of vehicles used; ttd=total travel distance; # FE= the number of fitness evaluations. SD refers to the SDVRLH2 algorithm; The meaning of A1 is shown in Table 7. The data in bold signifies that our strategy produces better results than the contrasted strategies numerically. The Mann-Whitney U test was employed, and the data with underlined values indicate a significant difference at the significance level  $\alpha = 0.05$

employed in these experiments. To assess the impact of each innovation, we sequentially added each innovation to the previous algorithm and conducted 10 repeated runs to obtain experimental results. This approach determined whether the

innovation indeed enhanced the algorithm's performance. Tables 8 - 15 demonstrate the comparative results.

**Improved Packing Methods (A1 vs. SDVRLH2).** As shown in Tables 8 and 9, on average, on the 32 small-scale

**Table 9** Ablation experimental results (A1 vs. SD) on large-scale instances ( $100 \leq \text{number of node} \leq 200$ )

| Instances (large) | Avg # v<br>A1-SD | Avg ttd<br>(A1-SD)/SD | Best # v<br>A1-SD | Best ttd<br>(A1-SD)/SD | CPU time<br>A1/SD | # FE<br>A1/SD |
|-------------------|------------------|-----------------------|-------------------|------------------------|-------------------|---------------|
| B-Y9              | <b>-3</b>        | <u>0.3164</u>         | <b>-3</b>         | 0.2479                 | 2.68              | 1.1103        |
| B-Y10             | <b>-2.7</b>      | <u>0.4860</u>         | <b>-2</b>         | 0.4524                 | 1.89              | 1.1227        |
| B-Y11             | <b>-3</b>        | <u>0.2225</u>         | <b>-3</b>         | 0.1956                 | 2.81              | 1.0722        |
| B-Y12             | <u>2</u>         | <u>0.7164</u>         | 2                 | 0.6958                 | 2.03              | 1.0945        |
| B-Y13             | <b>-5.1</b>      | <u>0.2838</u>         | <b>-6</b>         | 0.2689                 | 24.04             | 1.1049        |
| B-Y14             | <b>-1</b>        | <u>0.5245</u>         | <b>-1</b>         | 0.5180                 | 16.70             | 1.1463        |
| B-Y15             | <b>-8.5</b>      | <u>0.2511</u>         | <b>-8</b>         | 0.2115                 | 13.35             | 1.1068        |
| B-Y16             | <b>2.6</b>       | <u>0.7161</u>         | <b>3</b>          | 0.5765                 | 10.09             | 1.1247        |
| B-Y17             | <b>-9.5</b>      | <u>0.2392</u>         | <b>-10</b>        | 0.1901                 | 41.11             | 1.1234        |
| B-Y18             | <b>-4.9</b>      | <u>0.6911</u>         | <b>-5</b>         | 0.5789                 | 28.47             | 1.1510        |
| B-Y19             | <b>-11.2</b>     | <u>0.1048</u>         | <b>-12</b>        | 0.0726                 | 74.49             | 1.1147        |
| B-Y20             | <b>1.2</b>       | <u>0.6532</u>         | 2                 | 0.5925                 | 58.18             | 1.1310        |
| Sha14             | <b>-1.2</b>      | <u>0.4816</u>         | <b>-1</b>         | 0.4923                 | 4.55              | 1.0575        |
| Sha15             | <b>-3.9</b>      | <u>0.8108</u>         | <b>-3</b>         | 0.7939                 | 38.58             | 1.0750        |
| SD12              | 0                | <u>0.0534</u>         | 0                 | 0.0435                 | 18.05             | 1.3288        |
| SD13              | <b>-3</b>        | <u>1.0577</u>         | <b>-3</b>         | 0.9710                 | 23.50             | 1.0858        |
| Avg               | <b>-3.2</b>      | 0.4755                | <b>-3.1</b>       | 0.4313                 | 23.52             | 1.1198        |
| Total             | <b>-51.2</b>     | 0.4098                | <b>-50.0</b>      | 0.3693                 | 23.52             | 1.1198        |

# v = the number of vehicles used; ttd=total travel distance; # FE= the number of fitness evaluations. SD refers to the SDVRLH2 algorithm; The meaning of A1 is shown in Table 7. The data in bold signifies that our strategy produces better results than the contrasted strategies numerically. The Mann-Whitney U test was employed, and the data with underlined values indicate a significant difference at the significance level  $\alpha = 0.05$

instances, A1 reduced the number of vehicles on 20 instances, with a maximum reduction of 11.1 vehicles (SD11) and a cumulative reduction of 43.2 vehicles across all instances. On the 16 large-scale instances, A1 reduced the number of vehicles on 12 instances, with a maximum reduction of 11.2 vehicles (B-Y19) and a cumulative reduction of 51.2 vehicles across all instances. The experimental results indicate that the improved packing methods effectively reduce the number of vehicles, especially on larger-scale problems.

**New Search Operators (A2 vs. A1).** Building upon A1, the old search operators were replaced with our proposed new search operators to obtain Algorithm A2. According to Tables 10 and 11, compared to A1, A2 significantly reduced the total travel distance (*ttd*) on 9 out of 32 small-scale instances and on all 16 large-scale instances, with maximum reductions of 6.99% on instance B-Y8 and 11.09% on instance SD13, respectively. Notably, the new operators exhibited more significant effects on larger-scale instances, reducing the average *ttd* per instance by 6.15%. On an additional 20 small instances, the *ttd* of A2 was found to be as good as that of A1. Importantly, the use of new operators significantly reduced the number of fitness evaluations (FEs) consumed by the algorithm. For the small-scale instances, A2 used less than 10% of the FEs of A1 on 28 instances. The maximum usage was 35.58% on instance Sha1, and the minimum was 1.44% on instance SD11. For the 16 large-scale

instances, A2 used only 20% to 40% of the FEs required by A1. This demonstrates that the new operators can greatly improve the search efficiency of the algorithm and help find better solutions.

**Adaptive Splitting Strategy (A3 vs. A2).** Algorithm A3 is obtained by incorporating our proposed adaptive splitting strategy into A2. As shown in Tables 12 and 13, compared to A2, A3 increased the *ttd* while maintaining almost the same number of vehicles, and further significantly reduced the number of FEs. The number of FEs used by A3 accounted for only 0.2% to 4.09% of those used by A2.

**Post-Optimization (A4 vs. A3).** Algorithm A4 has been developed by integrating an innovative post-optimization technique into Algorithm A3. As demonstrated in Tables 14 and 15, the results obtained by A4 on most small-scale problem instances show no significant difference from A3. For the 16 large-scale instances, A4 considerably reduces the number of vehicles on 14 instances and the *ttd* on a single instance, resulting in an aggregate reduction of 23.6 vehicles. Further, considering the best results from multiple runs, A4 outperforms A3 on 14 instances in reducing the number of vehicles and on 2 instances in reducing the *ttd*. The experimental results substantiate that the post-optimization method introduced in this study effectively lessens the number of vehicles on larger-scale problems. However, it is crucial to note that reducing the number of vehicles often results in an increase

**Table 10** Ablation experimental results (A2 vs. A1) on small-scale instances (number of nodes <100)

| Instances (small) | Avg # v<br>A2-A1 | Avg ttd<br>(A2-A1)/A1 | Best # v<br>A2-A1 | Best ttd<br>(A2-A1)/A1 | CPU time<br>A2/A1 | # FE<br>A2/A1 |
|-------------------|------------------|-----------------------|-------------------|------------------------|-------------------|---------------|
| B-Y1              | <u>0.7</u>       | <b>-0.0336</b>        | 0                 | <b>-0.0155</b>         | 0.07              | 0.0216        |
| B-Y2              | 0                | 0.0104                | 0                 | 0.0214                 | 0.06              | 0.0218        |
| B-Y3              | 0                | <b>-0.0658</b>        | 0                 | <b>-0.0114</b>         | 0.08              | 0.0237        |
| B-Y4              | <b>-0.1</b>      | <b>-0.0383</b>        | 0                 | <b>-0.0149</b>         | 0.07              | 0.0238        |
| B-Y5              | 0                | <b>-0.0114</b>        | 0                 | <b>-0.0119</b>         | 0.01              | 0.0173        |
| B-Y6              | 0.1              | 0.0172                | 0                 | 0.0225                 | 0.01              | 0.0175        |
| B-Y7              | 0                | <b>-0.0107</b>        | 0                 | <b>-0.0176</b>         | 0.02              | 0.0176        |
| B-Y8              | 0                | <b>-0.0699</b>        | 0                 | <b>-0.0465</b>         | 0.01              | 0.0207        |
| Sha1              | 0                | 0.0000                | 0                 | 0.0000                 | 0.25              | 0.3558        |
| Sha2              | 0                | <b>-0.0030</b>        | 0                 | 0.0000                 | 0.23              | 0.1580        |
| Sha3              | 0                | <b>-0.0342</b>        | 0                 | <b>-0.0288</b>         | 0.19              | 0.1267        |
| Sha4              | 0                | <b>-0.0251</b>        | 0                 | 0.0000                 | 0.23              | 0.0945        |
| Sha5              | 0                | <b>-0.0139</b>        | 0                 | 0.0033                 | 0.24              | 0.0740        |
| Sha6              | 0                | <b>-0.0012</b>        | 0                 | <b>-0.0049</b>         | 0.16              | 0.0687        |
| Sha7              | 0                | <b>-0.0293</b>        | 0                 | <b>-0.0236</b>         | 0.15              | 0.0619        |
| Sha8              | 0                | <b>-0.0143</b>        | 0                 | <b>-0.0017</b>         | 0.21              | 0.0436        |
| Sha9              | 0                | 0.0127                | 0                 | 0.0028                 | 0.13              | 0.0320        |
| Sha10             | 0                | <u>0.0233</u>         | 0                 | 0.0017                 | 0.07              | 0.0279        |
| Sha11             | 0                | <u>0.0302</u>         | 0                 | 0.0188                 | 0.05              | 0.0232        |
| Sha12             | 0                | <u>0.0169</u>         | 0                 | 0.0042                 | 0.07              | 0.0277        |
| Sha13             | 0                | 0.0067                | 0                 | 0.0140                 | 0.02              | 0.0199        |
| SD1               | 0                | <b>-0.0080</b>        | 0                 | 0.0471                 | 1.62              | 0.1008        |
| SD2               | 0                | <b>-0.0058</b>        | 0                 | 0.0159                 | 0.33              | 0.0377        |
| SD3               | 0                | 0.0193                | 0                 | 0.0194                 | 0.13              | 0.0269        |
| SD4               | 0                | <b>-0.0026</b>        | 0                 | <b>-0.0207</b>         | 0.10              | 0.0254        |
| SD5               | 0                | <b>-0.0008</b>        | 0                 | 0.0000                 | 0.08              | 0.0271        |
| SD6               | 0                | <b>-0.0250</b>        | 0                 | <b>-0.0128</b>         | 0.43              | 0.0260        |
| SD7               | 0                | <b>-0.0045</b>        | 0                 | 0.0499                 | 0.07              | 0.0217        |
| SD8               | 0                | <b>-0.0201</b>        | 0                 | <b>-0.0152</b>         | 0.23              | 0.0259        |
| SD9               | 0                | <b>-0.0062</b>        | 0                 | 0.0012                 | 0.08              | 0.0203        |
| SD10              | 0                | 0.0212                | 0                 | 0.0378                 | 0.07              | 0.0193        |
| SD11              | 0                | <u>0.0699</u>         | 0                 | 0.0495                 | 0.05              | 0.0144        |
| Avg               | 0.02             | <b>-0.0061</b>        | 0                 | 0.0026                 | 0.03              | 0.0191        |
| Total             | 0.7              | 0.0022                | 0                 | 0.0093                 | 0.03              | 0.0191        |

# v = the number of vehicles used; ttd=total travel distance; # FE= the number of fitness evaluations. The meaning of A1 and A2 is shown in Table 7. The data in bold signifies that our strategy produces better results than the contrasted strategies numerically. The Mann-Whitney U test was employed, and the data with underlined values indicate a significant difference at the significance level  $\alpha = 0.05$

in *ttd*. Moreover, post-optimization for large-scale problems demands considerable time, posing additional challenges.

In summary, the ablation experiments demonstrate the effectiveness and validity of each proposed innovation. The improved packing method (A1) effectively reduces the number of vehicles, particularly on larger-scale instances. The new search operators (A2) not only further reduce *ttd* on various instances but also significantly improve search efficiency by substantially reducing the number of FEs consumed. The adaptive splitting strategy (A3) maintains a similar number

of vehicles while achieving a further reduction in the number of FEs. Lastly, the new post-optimization method (A4) contributes to reducing the number of vehicles on large-scale instances.

#### 4.3.3 Time analysis

In real-world industrial scenarios, computational resource consumption is critical, placing higher demands on algorithm efficiency. Since theoretical analysis of computational

**Table 11** Ablation experimental results (A2 vs. A1) on large-scale instances ( $100 \leq \text{number of node} \leq 200$ )

| Instances (large) | Avg # v<br>A2-A1 | Avg ttd<br>(A2-A1)/A1 | Best # v<br>A2-A1 | Best ttd<br>(A2-A1)/A1 | CPU time<br>A2/A1 | # FE<br>A2/A1 |
|-------------------|------------------|-----------------------|-------------------|------------------------|-------------------|---------------|
| B-Y9              | 0                | <u>-0.0302</u>        | 0                 | <b>-0.0316</b>         | 0.38              | 0.2963        |
| B-Y10             | 0                | <u>-0.0304</u>        | 0                 | <b>-0.0322</b>         | 0.35              | 0.2933        |
| B-Y11             | 0                | <u>-0.0496</u>        | 0                 | <b>-0.0462</b>         | 0.41              | 0.3089        |
| B-Y12             | 0                | <u>-0.0453</u>        | 0                 | <b>-0.0393</b>         | 0.38              | 0.2996        |
| B-Y13             | 0.1              | <u>-0.0603</u>        | 1                 | <b>-0.0585</b>         | 0.45              | 0.3816        |
| B-Y14             | 0                | <u>-0.0745</u>        | 0                 | <b>-0.0692</b>         | 0.44              | 0.3679        |
| B-Y15             | -0.1             | <u>-0.0784</u>        | 0                 | <b>-0.0568</b>         | 0.85              | 0.3807        |
| B-Y16             | <u>0.4</u>       | <u>-0.0887</u>        | 0                 | <b>-0.0587</b>         | 0.84              | 0.3741        |
| B-Y17             | <u>-0.5</u>      | <u>-0.1028</u>        | 0                 | <b>-0.0751</b>         | 0.60              | 0.4082        |
| B-Y18             | <u>-0.2</u>      | <u>-0.0896</u>        | 0                 | <b>-0.0801</b>         | 0.48              | 0.3978        |
| B-Y19             | 0.2              | <u>-0.0810</u>        | 1                 | <b>-0.0603</b>         | 0.48              | 0.4103        |
| B-Y20             | 0.2              | <u>-0.0896</u>        | 1                 | <b>-0.0769</b>         | 0.48              | 0.4038        |
| Sha14             | 0                | <u>-0.0574</u>        | 0                 | <b>-0.0591</b>         | 0.24              | 0.2037        |
| Sha15             | 0                | <u>-0.0912</u>        | 0                 | <b>-0.0866</b>         | 0.30              | 0.2895        |
| SD12              | 0                | <u>0.0952</u>         | 0                 | <b>0.1003</b>          | 0.09              | 0.2898        |
| SD13              | 0                | <u>-0.1109</u>        | 0                 | <b>-0.0694</b>         | 0.08              | 0.3547        |
| Avg               | 0.01             | <b>-0.0615</b>        | 0.2               | <b>-0.0500</b>         | 0.43              | 0.3595        |
| Total             | 0.1              | <b>-0.0458</b>        | 3.0               | <b>-0.0273</b>         | 0.43              | 0.3595        |

# v = the number of vehicles used; ttd=total travel distance; # FE= the number of fitness evaluations. The meaning of A1 and A2 is shown in Table 7. The data in bold signifies that our strategy produces better results than the contrasted strategies numerically. The Mann-Whitney U test was employed, and the data with underlined values indicate a significant difference at the significance level  $\alpha = 0.05$

time complexity is only practical for simple metaheuristic algorithms on artificial problems, we conducted experiments to perform an empirical time analysis. To comprehensively evaluate the efficiency of our proposed algorithm, we compared its CPU runtime with that of the state-of-the-art SDVRLH2 algorithm for the 3L-SDVRP, using results reported in Bortfeldt and Yi [5].

As shown in Table 16, for all small-scale instances, SDVRLH2 required a total of 6627.5 seconds, averaging 207.1 seconds per instance. In contrast, our proposed algorithm, when minimizing the number of vehicles (Ov), took only 397.0 seconds in total, averaging 12.4 seconds per instance—just 6% of the time required by SDVRLH2. When total travel distance was set as the primary objective (Ot), our algorithm took 524.7 seconds in total, averaging 19.8 seconds per instance, which is only 8% of the running time required by SDVRLH2. For all large-scale instances, SDVRLH2 required a total of 30673.1 seconds, averaging 1917.1 seconds per instance. In comparison, our algorithm needed 13468.4 seconds in total, averaging 597.2 seconds per instance when minimizing the number of vehicles (Ov)—only 44% of the time required by SDVRLH2. When total travel distance was the primary objective (Ot), our algorithm required just 3078.0 seconds in total, averaging 192.4 seconds per instance, or only 10% of the CPU time of SDVRLH2. Overall, the runtime of our algorithm was just

37% of SDVRLH2 when optimizing for the number of vehicles (Ov), and only 9% when optimizing for total travel distance (Ot).

Moreover, our algorithm was implemented in Python, whereas SDVRLH2 was implemented in C++. Considering that programs implemented in C++ generally run faster than those in Python, the efficiency advantage of our proposed algorithm is likely even greater than these results suggest. This underscores the high efficiency of our method.

## 5 Conclusion

In this paper, we present an efficient algorithm for solving the 3L-SDVRP. Our proposed algorithm encompasses several innovations that contribute to its effectiveness and efficiency. Regarding packing, we introduce improvements to the box-packing, space-generation, and 2C-SP construction methods, enhancing space utilization and reducing the number of vehicles required. In terms of routing, we propose three novel search operators: LKH mutation, route-pair swap, and multi-pair elitist recombination. These operators leverage the characteristics of the 3L-SDVRP, improving search efficiency. Moreover, we introduce an adaptive splitting strategy that dynamically determines whether to split nodes based on the loading conditions of vehicles and nodes, thereby fur-

**Table 12** Ablation experiment results (A3 vs. A2) on small-scale instances (number of node <100)

| Instance (small) | Avg # v<br>A3-A2 | Avg ttd<br>(A3-A2)/A2 | Best # v<br>A3-A2 | Best ttd<br>(A3-A2)/A2 | # FE<br>A3/A2 |
|------------------|------------------|-----------------------|-------------------|------------------------|---------------|
| B-Y1             | <u>0</u>         | <u>0.0479</u>         | 0                 | 0.0292                 | 0.0085        |
| B-Y2             | 0                | <u>0.0983</u>         | 0                 | 0.0791                 | 0.0102        |
| B-Y3             | 0.1              | <u>0.1074</u>         | 0                 | 0.0763                 | 0.0130        |
| B-Y4             | 0.2              | <u>0.0797</u>         | 0                 | 0.0675                 | 0.0095        |
| B-Y5             | 0                | <u>0.0755</u>         | 0                 | 0.0698                 | 0.0065        |
| B-Y6             | <u>0.6</u>       | <u>0.0419</u>         | 0                 | 0.0894                 | 0.0062        |
| B-Y7             | 0                | <u>0.0680</u>         | 0                 | 0.0545                 | 0.0065        |
| B-Y8             | 0.1              | <u>0.0805</u>         | 0                 | 0.0668                 | 0.0059        |
| Sha1             | 0                | 0.0117                | 0                 | 0                      | 0.0409        |
| Sha2             | 0.2              | <u>0.0259</u>         | 0                 | 0.0083                 | 0.0411        |
| Sha3             | 0                | <u>0.0437</u>         | 0                 | 0.0103                 | 0.0194        |
| Sha4             | 0                | <u>0.1285</u>         | 0                 | 0                      | 0.0267        |
| Sha5             | 0                | <u>0.0285</u>         | 0                 | 0.0044                 | 0.0354        |
| Sha6             | 0                | <u>0.0416</u>         | 0                 | 0.0311                 | 0.0206        |
| Sha7             | 0.2              | <u>0.0808</u>         | 0                 | 0.0603                 | 0.0231        |
| Sha8             | 0                | <u>0.0394</u>         | 0                 | 0.0245                 | 0.0338        |
| Sha9             | 0                | <u>0.0626</u>         | 0                 | 0.0209                 | 0.0164        |
| Sha10            | 0                | <u>0.0487</u>         | 0                 | 0.0292                 | 0.0166        |
| Sha11            | 0                | <u>0.0623</u>         | 0                 | 0.0447                 | 0.0090        |
| Sha12            | 0                | <u>0.0714</u>         | 0                 | 0.0555                 | 0.0119        |
| Sha13            | 0                | <u>0.0567</u>         | 0                 | 0.0513                 | 0.0070        |
| SD1              | 0                | <u>0.1004</u>         | 0                 | 0.0484                 | 0.0281        |
| SD2              | 0                | <u>0.1045</u>         | 0                 | 0.0735                 | 0.0207        |
| SD3              | 0                | <u>0.0953</u>         | 0                 | 0.0982                 | 0.0162        |
| SD4              | 0                | <u>0.1290</u>         | 0                 | 0.0930                 | 0.0131        |
| SD5              | 0                | <u>0.0069</u>         | 0                 | 0.0004                 | 0.0084        |
| SD6              | 0                | <u>0.0713</u>         | 0                 | 0.0422                 | 0.0090        |
| SD7              | <u>0.6</u>       | <u>0.1230</u>         | 0                 | 0.0766                 | 0.0127        |
| SD8              | 0                | <u>0.0484</u>         | 0                 | 0.0553                 | 0.0082        |
| SD9              | 0                | <u>0.0978</u>         | 0                 | 0.0580                 | 0.0096        |
| SD10             | 0                | <u>0.1248</u>         | 0                 | 0.0828                 | 0.0077        |
| SD11             | 0                | <u>0.1559</u>         | 0                 | 0.1453                 | 0.0041        |
| Avg              | 0.06             | 0.0737                | 0.0               | 0.0515                 | 0.0158        |
| Total            | 2.0              | 0.0891                | 0                 | 0.0679                 | 0.0078        |

# v = the number of vehicle used; ttd=total travel distance; # FE= the number of fitness evaluations. The meaning of A2 and A3 is shown in Table 7. The Mann-Whitney U test was employed, and the data with underlined values indicate a significant difference at the significance level  $\alpha = 0.05$

ther reducing the consumption of computational resources. Finally, we design a new post-optimization method to further improve the solution quality. Comprehensive experimental results and further analysis confirm the efficiency and effectiveness of our proposed method. In particular, compared to the current state-of-the-art algorithm, SDVRLH2 ([5]), our proposed algorithm achieved a significant reduction in the number of vehicles on most examined problem instances.

Furthermore, it achieved a significant reduction in the quantity of fitness evaluations required, decreasing by two orders of magnitude compared to SDVRLH2.

**Table 13** Ablation experiment results (A3 vs. A2) on large-scale instances ( $100 \leq \text{number of node} \leq 200$ )

| Instances (large) | Avg # v<br>A3-A2 | Avg ttd<br>(A3-A2)/A2 | Best # v<br>A3-A2 | Best ttd<br>(A3-A2)/A2 | # FE<br>A3/A2 |
|-------------------|------------------|-----------------------|-------------------|------------------------|---------------|
| B-Y9              | <u>0.5</u>       | <u>0.0272</u>         | 0                 | 0.0592                 | 0.0048        |
| B-Y10             | 0.2              | <u>0.1133</u>         | 0                 | 0.1119                 | 0.0040        |
| B-Y11             | 0                | <u>0.0745</u>         | 0                 | 0.0649                 | 0.0040        |
| B-Y12             | 0                | <u>0.0912</u>         | 0                 | 0.0735                 | 0.0044        |
| B-Y13             | 0.1              | <u>0.0693</u>         | 0                 | 0.0558                 | 0.0026        |
| B-Y14             | 0                | <u>0.1281</u>         | 0                 | 0.1141                 | 0.0029        |
| B-Y15             | <u>0.6</u>       | <u>0.0751</u>         | 0                 | 0.0579                 | 0.0026        |
| B-Y16             | 0.1              | <u>0.0922</u>         | 1                 | 0.0752                 | 0.0025        |
| B-Y17             | <u>0.3</u>       | <u>0.0935</u>         | 0                 | 0.0785                 | 0.0018        |
| B-Y18             | <u>0.5</u>       | <u>0.0987</u>         | 1                 | 0.1525                 | 0.0018        |
| B-Y19             | 0.2              | <u>0.0880</u>         | -1                | 0.0625                 | 0.0016        |
| B-Y20             | -0.1             | <u>0.1056</u>         | -1                | 0.0803                 | 0.0019        |
| Sha14             | 0                | <u>0.1477</u>         | 0                 | 0.1235                 | 0.0031        |
| Sha15             | 0                | <u>0.1112</u>         | 0                 | 0.1101                 | 0.0017        |
| SD12              | 0                | <u>0.2068</u>         | 0                 | 0.1718                 | 0.0035        |
| SD13              | 0                | <u>0.1333</u>         | 0                 | 0.1124                 | 0.0029        |
| Avg               | 0.15             | 0.1035                | 0.0               | 0.0940                 | 0.0029        |
| Total             | 2.4              | 0.1312                | 0.0               | 0.1130                 | 0.0022        |

# v = the number of vehicles used; ttd=total travel distance; # FE= the number of fitness evaluations. The meaning of A2 and A3 is shown in Table 7. The Mann-Whitney U test was employed, and the data with underlined values indicate a significant difference at the significance level  $\alpha = 0.05$

**Table 14** Ablation experiment results (A4 vs. A3) on small-scale instances (number of node < 100)

| Instances (small) | Avg # v<br>A4-A3 | Avg ttd<br>(A4-A3)/A3 | Best # v<br>A4-A3 | Best ttd<br>(A4-A3)/A3 |
|-------------------|------------------|-----------------------|-------------------|------------------------|
| B-Y1              | <u>-0.9</u>      | <u>0.0942</u>         | 0                 | 0.0211                 |
| B-Y2              | 0                | -0.0098               | 0                 | -0.0020                |
| B-Y3              | -0.1             | -0.0045               | 0                 | -0.0040                |
| B-Y4              | <u>-0.2</u>      | 0.0065                | 0                 | 0                      |
| B-Y5              | <u>-1</u>        | <u>0.0832</u>         | -1                | 0.0836                 |
| B-Y6              | <u>-0.7</u>      | <u>0.0312</u>         | 0                 | -0.0076                |
| B-Y7              | <u>-1</u>        | <u>0.0533</u>         | -1                | 0.0303                 |
| B-Y8              | -0.1             | -0.0128               | 0                 | -0.0232                |
| Sha1              | 0                | 0.0000                | 0                 | 0                      |
| Sha2              | <u>-0.2</u>      | -0.0116               | 0                 | -0.0056                |
| Sha3              | 0                | 0.0000                | 0                 | 0                      |
| Sha4              | 0                | -0.0014               | 0                 | 0                      |
| Sha5              | 0                | 0.0000                | 0                 | 0                      |
| Sha6              | 0                | -0.0010               | 0                 | 0                      |
| Sha7              | <u>-0.2</u>      | <u>-0.0364</u>        | 0                 | -0.0557                |
| Sha8              | 0                | -0.0012               | 0                 | -0.0003                |
| Sha9              | 0                | -0.0122               | 0                 | -0.0258                |
| Sha10             | <u>-1</u>        | <u>0.0636</u>         | -1                | 0.0120                 |
| Sha11             | 0                | -0.0036               | 0                 | -0.0007                |

**Table 14** continued

| Instances (small) | Avg # v<br>A4-A3 | Avg ttd<br>(A4-A3)/A3 | Best # v<br>A4-A3 | Best ttd<br>(A4-A3)/A3 |
|-------------------|------------------|-----------------------|-------------------|------------------------|
| Sha12             | 0                | <b>-0.0121</b>        | 0                 | <b>-0.0152</b>         |
| Sha13             | <b>-0.8</b>      | 0.0284                | <b>-1</b>         | <b>-0.0057</b>         |
| SD1               | 0                | <b>-0.0077</b>        | 0                 | 0                      |
| SD2               | 0                | <b>-0.0091</b>        | 0                 | 0                      |
| SD3               | 0                | <b>-0.0053</b>        | 0                 | <b>-0.0003</b>         |
| SD4               | 0                | <b>-0.0098</b>        | 0                 | 0                      |
| SD5               | 0                | <b>-0.0002</b>        | 0                 | 0                      |
| SD6               | 0                | <b>-0.0168</b>        | 0                 | <b>-0.0057</b>         |
| SD7               | <b>-0.6</b>      | 0.0166                | 0                 | 0.0005                 |
| SD8               | 0                | <b>-0.0136</b>        | 0                 | <b>-0.0165</b>         |
| SD9               | 0                | <b>-0.0123</b>        | 0                 | <b>-0.0175</b>         |
| SD10              | 0                | <b>-0.0034</b>        | 0                 | 0                      |
| SD11              | 0                | <b>-0.0032</b>        | 0                 | 0                      |
| Avg               | <b>-0.2</b>      | 0.0059                | <b>-0.1</b>       | <b>-0.0012</b>         |
| Total             | <b>-6.8</b>      | <b>-0.0020</b>        | <b>-4</b>         | <b>-0.0030</b>         |

# v = the number of vehicle used; ttd=total travel distance; # FE= the number of fitness evaluations. The meaning of A3 and A4 is shown in Table 7. The Mann-Whitney U test was employed, and the data with underlined values indicate a significant difference at the significance level  $\alpha = 0.05$

**Table 15** Ablation experiment results (A4 vs. A3) on large-scale instances ( $100 \leq \text{number of node} \leq 200$ )

| Instance (large) | Avg # v<br>A4-A3 | Avg ttd<br>(A4-A3)/A3 | Best # v<br>A4-A3 | Best ttd<br>(A4-A3)/A3 |
|------------------|------------------|-----------------------|-------------------|------------------------|
| B-Y9             | <b>-1.5</b>      | 0.1017                | <b>-1</b>         | 0.0888                 |
| B-Y10            | <b>-0.3</b>      | 0.0064                | <b>-1</b>         | <b>-0.0156</b>         |
| B-Y11            | <b>-1.1</b>      | 0.0705                | <b>-2</b>         | 0.0311                 |
| B-Y12            | <b>-1</b>        | 0.0708                | <b>-1</b>         | 0.0634                 |
| B-Y13            | <b>-2.1</b>      | 0.0840                | <b>-2</b>         | 0.0658                 |
| B-Y14            | <b>-1</b>        | 0.0529                | <b>-1</b>         | 0.0356                 |
| B-Y15            | <b>-2.5</b>      | 0.0968                | <b>-2</b>         | 0.0883                 |
| B-Y16            | <b>-2</b>        | 0.0898                | <b>-2</b>         | 0.0787                 |
| B-Y17            | <b>-2.5</b>      | 0.0594                | <b>-3</b>         | 0.0446                 |
| B-Y18            | <b>-2</b>        | 0.0884                | <b>-2</b>         | 0.0661                 |
| B-Y19            | <b>-3.4</b>      | 0.1091                | <b>-3</b>         | 0.0860                 |
| B-Y20            | <b>-2.1</b>      | 0.0735                | <b>-2</b>         | 0.0723                 |
| Sha14            | <b>-1</b>        | 0.0628                | <b>-1</b>         | 0.0234                 |
| Sha15            | <b>-1.1</b>      | 0.0748                | <b>-2</b>         | 0.0450                 |
| SD12             | 0                | <b>-0.0274</b>        | 0                 | <b>-0.0280</b>         |
| SD13             | 0                | <b>-0.0071</b>        | 0                 | <b>-0.0096</b>         |
| Avg              | <b>-1.5</b>      | 0.0629                | <b>-1.6</b>       | 0.0460                 |
| Total            | <b>-23.6</b>     | 0.0281                | <b>-25.0</b>      | 0.0191                 |

# v = the number of vehicles used; ttd=total travel distance; # FE= the number of fitness evaluations. The meaning of A3 and A4 is shown in Table 7. The Mann-Whitney U test was employed, and the data with underlined values indicate a significant difference at the significance level  $\alpha = 0.05$

**Table 16** CPU time analysis (in seconds)

| Statistics | SDVRLH2 (SD) |         | Ov    |         | Ot    |        |
|------------|--------------|---------|-------|---------|-------|--------|
|            | Small        | Large   | Small | Large   | Small | Large  |
| Total      | 6627.5       | 30673.1 | 397.0 | 13468.4 | 524.7 | 3078.0 |
| Avg        | 207.1        | 1917.1  | 12.4  | 841.8   | 16.4  | 192.4  |
| Std        | 365.9        | 441.4   | 15.4  | 597.2   | 19.8  | 156.1  |
| Overall    | Ov/SD        |         | All   | Ot/SD   |       | All    |
|            | Small        | Large   |       | Small   | Large |        |
|            | 0.06         | 0.44    | 0.37  | 0.08    | 0.10  | 0.09   |

Ov (Ot) refers to our algorithm with number of  $v$  ( $ttd$ ) as the first objective. Total = the sum of CPU time across all problem instances; Avg = average CPU time; Std = standard deviation. Small = small problem instances whose number of nodes < 100. Large = large problem instances whose number of nodes  $\geq 100$

While our algorithm efficiently reduces the number of vehicles within a much shorter time, it does so at the expense of an increased total travel distance ( $ttd$ ). Our future work will focus on reducing the total distance further, although there is a conflict between the number of vehicles and the  $ttd$ . It is worth noting that the 3L-SDVRP considered in this paper is not a classical multi-objective optimization problem because the number of vehicles is our primary priority and the  $ttd$  is only our secondary objective.

## Appendix A: Giant tour decoding and evaluation

This appendix section describe the giant tour decoding process in detail to support Sect. 3.1 of the main text.

In this study, we utilize the concept of a giant tour to represent a solution for the 3L-SDVRP. It has been first proposed by Beasley [2] and has since been used in various variants of the VRP ([1, 5, 9, 10, 12–14, 19, 20, 23, 27, 30]). For a comprehensive survey on the giant-tour representation of VRP, interested readers can refer to Prins et al [24]. A giant tour comprises a permutation of all nodes, but it is not a feasible solution since it is not possible to visit all nodes with a single vehicle without violating any constraints. A key question arises: how can a giant tour be decoded into a feasible solution and subsequently evaluated? The giant tour decoding process follows the method outlined in Bortfeldt and Yi [5], summarized here for clarity. As demonstrated in Algorithm 9, we first examine and adjust  $s_{giant\_tour}$  to ensure that the two nodes corresponding to each 2C-SP are adjacent (Step 2). Then, we initialize an empty vehicle  $v$  (Step 3) that visits nodes and loads layers in the order specified by  $s_{giant\_tour}$  (Steps 4–16).

When visiting node  $i$ , if all layers in  $i$  can be loaded into  $v$ , the loading proceeds (Steps 5–6). Otherwise, a decision must

be made on whether to perform a splitting delivery at node  $i$  (Step 8). If required, as many layers as possible from  $i$  are loaded into  $v$  (Steps 9–11). Regardless of whether splitting delivery occurs at  $i$ ,  $v$  will not visit any other nodes (Step 12). Instead, a new empty vehicle is created to continue visiting subsequent nodes, starting from  $i$  (Step 13). After obtaining all vehicles (Step 15), we calculate and return the number of vehicles used ( $n_v$ ) and the total travel distance ( $d_{ttd}$ ) for all vehicles (Steps 17–18).

For solution evaluation, the 3L-SDVRP has two objectives: minimizing the number of vehicles and the  $ttd$ . In line with Bortfeldt and Yi [5], we prioritize the number of vehicles as the primary objective, while total travel distance serves as a secondary consideration. Therefore, when evaluating the quality of two solutions, we first examine the number of vehicles, with fewer vehicles indicating a superior solution. When the number of vehicles is the same, we then compare the total travel distances, with shorter distances representing better solutions.

### Algorithm 9 Decode and evaluate a giant tour (Bortfeldt and Yi [5])

---

**Input:** a giant tour  $s_{giant\_tour}$   
**Output:** number of vehicle  $n_v$ , total travel distance  $d_{ttd}$

---

```

1:  $V \leftarrow \emptyset$ 
2: check and adjust  $s_{giant\_tour}$  to ensure that both nodes of all 2C-SPs
   are adjacent
3:  $v \leftarrow$  initialize an empty vehicle
4: for node  $i$  in  $s_{giant\_tour}$  do
5:   if all layers in  $i$  can be loaded into  $v$  then
6:      $v \leftarrow$  load all layers in  $i$  into  $v$ 
7:   else
8:      $b_{splitting} \leftarrow$  determine whether node  $i$  requires splitting deliv-
       ery by our adaptive splitting strategy
9:     if  $b_{splitting}$  is true then
10:       $v \leftarrow$  load as many layers as possible into  $v$ 
11:    end if
12:     $V \leftarrow V \cup \{v\}$ 
13:     $v \leftarrow$  create a new empty vehicle and load remaining layers
       of  $i$ 
14:   end if
15:  $V \leftarrow V \cup \{v\}$ 
16: end for
17:  $n_v, d_{ttd} \leftarrow$  calculate the number of vehicle and the total travel
   distance based on  $V$ 
18: return  $n_v, d_{ttd}$ 

```

---

## Appendix B: Basic layer and packing patterns

This appendix section introduce the basic layer and three packing patterns in detail to support Sect. 3.2 of the paper main text.

The packing algorithm presented in this paper is inspired by the work of Bortfeldt and Yi [5], and its overall structure is delineated in Algorithm 10. We adopt a layer-based packing

strategy, which has been widely used in various studies ([4, 6, 7, 11, 20, 31]). As illustrated in Fig. 2, the layer-defining box is the first loaded box of the layer and thus determines the layer depth. For each node, all boxes form several cuboid layers with varying depths. During the routing search, the cuboid layers serve as the smallest packing units. Consequently, we only need to consider whether the depth of the layers can be accommodated in the vehicle, rather than considering the packing details (i.e., the 3D position and loading sequence) of each box within the cuboid layer. This approach focuses on the SDVRP by avoiding repeated packing problems during routing, thus reducing computational complexity.

#### Algorithm 10 Overview of our packing approach

**Input:** problem data, parameters

**Output:** packing solution

- 1: generate basic layers for each node by Algorithm 11
- 2: construct 1C-FLP and 1C-SP for each node based on its basic layers
- 3: construct 2C-SP pattern based on 1C-SP of each node
- 4: prepare final packing solution with 1C-FLP, 1C-SP, and 2C-SP
- 5: **return** packing solution

Within this context, there are several key concepts: basic layer, 1C-FLP, 1C-SP, and 2C-SP. A basic layer, as illustrated in Fig. 2, is a cuboid layer composed of boxes originating from the same node. Multiple distinct basic layers arise from the boxes of a single node.

Algorithm 11 outlines the packing heuristic method for obtaining all basic layers for a single node. In this method, each layer definition list comprises all unloaded boxes, sorted by their volumes. Initially, the layer defining list of the first layer,  $LDL_1$ , is generated. Then, for each layer defining box in  $LDL_1$ , a packing arrangement is produced after completing each outer loop (Algorithm 11 Steps 3–17). To elaborate, for each layer defining box in  $LDL_1$ , a cuboid layer is created as the first layer (Step 4). Subsequently, numerous candidate layers (stored in  $L_{next}$ ) are generated, and the most suitable one,  $l_{next}$ , is selected as the next layer (Steps 6–15). This process involves generating the layer defining list of the next layer,  $LDL_n$  (Step 7), followed by the construction of a candidate layer for each layer defining box in  $LDL_n$  (Steps 9–12). The best layer from  $L_{next}$  is then chosen as the next layer (Step 13). By repeating these steps, various packing arrangements stored in  $P$  are obtained. Ultimately, the best packing arrangement is selected as the final packing solution of the node. Throughout this process, the packing of different nodes remains independent.

As previously mentioned, the basic layer serves as the smallest unit of vehicle loading in our method. Thus, constructing basic layers from the boxes of each node significantly impacts algorithm performance. To avoid confusion, we introduce the concept of space: as illustrated in Fig. 5, a space in this paper refers to a vacant cuboidal region available

#### Algorithm 11 Construct basic layers

**Input:** problem data of one node, parameters

**Output:** basic layers of the node  $p$

- 1:  $P \leftarrow \emptyset$ ;  $s \leftarrow \emptyset$
- 2:  $LDL_1 \leftarrow$  generate the layer defining list of the first layer
- 3: **for**  $i \leftarrow 1$  to  $|LDL_1|$  **do**
- 4:  $l \leftarrow$  generate a layer with layer defining box  $LDL_1[i]$  via Algorithm 12
- 5:  $s \leftarrow s \cup \{l\}$
- 6: **while** existing unloaded boxes **do**
- 7:  $LDL_n \leftarrow$  generate the layer defining list of the next layer
- 8:  $L_{next} \leftarrow \emptyset$
- 9: **for**  $j \leftarrow 1$  to  $|LDL_n|$  **do**
- 10:  $l \leftarrow$  generate a layer with layer defining box  $LDL_n[j]$  via Algorithm 12
- 11:  $L_{next} \leftarrow L_{next} \cup \{l\}$
- 12: **end for**
- 13:  $l_{next} \leftarrow$  select the layer with the best loading rate from  $L_{next}$
- 14:  $s \leftarrow s \cup \{l_{next}\}$
- 15: **end while**
- 16:  $P \leftarrow P \cup \{s\}$ ;  $s \leftarrow \emptyset$
- 17: **end for**
- 18:  $p \leftarrow$  select best packing arrangement
- 19: **return**  $p$

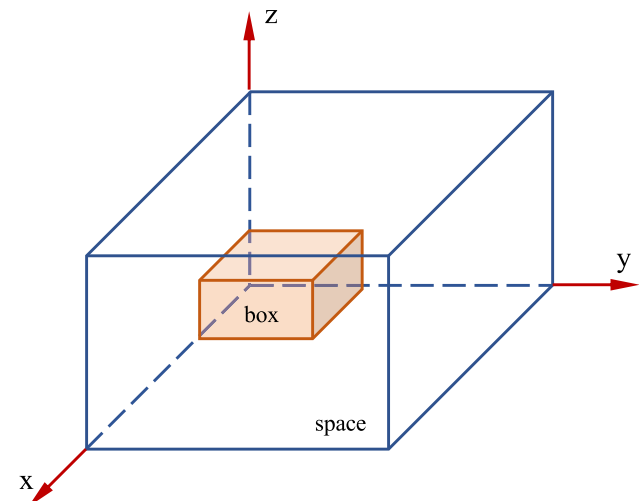


Fig. 5 Illustration of cuboid space and box

for loading boxes. Loading a box into a space entails placing the box such that its rear-bottom-left vertex coincides with that of the space.

Algorithm 12 delineates the process of constructing a basic layer. Initially, an empty layer is determined by the layer defining box  $b_0$  as the first space of the layer (Step 2). The layer depth equals the length of  $b_0$  in the x-axis direction, while its width and height correspond to the respective dimensions of the vehicle. After loading  $b_0$ , new subspaces are generated along the x, y, and z axes for subsequent loading (Steps 3–4). The following process is repeated until no available spaces remain or all boxes of the node have been loaded: extract spaces sequentially from the space stack  $S_{avail}$  (Steps 6–7), then select a box from the set of unloaded boxes  $B_{free}$

that can be loaded into the space, load it, and generate new subspaces along the x, y, and z axes (Steps 8–10).

---

**Algorithm 12** Generate a basic layer
 

---

**Input:**  $b_0$ : layer defining box;  $B_{free}$ : set of unloaded boxes

**Output:** a basic layer

```

1:  $S_{avail} \leftarrow \emptyset$ 
2:  $s_0 \leftarrow$  initialize an empty layer as the first space based on  $b_0$  (the
   length of  $b_0$  in the x-direction corresponds to the depth of the layer)
3:  $S_{new} \leftarrow$  load  $b_0$  into  $s_0$  and subsequently generate multiple sub-
   spaces
4:  $S_{avail} \leftarrow S_{avail} \cup S_{new}$ 
5: while  $S_{avail} \neq \emptyset$  and  $B_{free} \neq \emptyset$  do
6:    $s_{curr} \leftarrow$  select the uppermost space from  $S_{avail}$ 
7:    $S_{avail} \leftarrow S_{avail} \setminus \{s_{curr}\}$ 
8:    $b \leftarrow$  select a box that can be loaded in  $s_{curr}$  from  $B_{free}$ 
9:    $B_{free} \leftarrow B_{free} \setminus \{b\}$ 
10:   $S_{new} \leftarrow$  load  $b$  into  $s_{curr}$  and subsequently generate multiple
     subspaces
11:   $S_{avail} \leftarrow S_{avail} \cup S_{new}$ 
12: end while
13: return the constructed basic layer
  
```

---

Building upon the basic layer concept, Bortfeldt and Yi [5] proposed three distinct packing patterns: 1C-FLP, 1C-SP, and 2C-SP.

The 1C-FLP, or one customer full load pattern, is created by combining multiple basic layers from the same node. This pattern requires an empty vehicle to load and directly transport the cargo to the final destination without visiting other nodes. If the sum of the depths of all basic layers of a node exceeds the vehicle capacity, it indicates that a single vehicle cannot accommodate all layers of the node; thus, at least one 1C-FLP can be constructed. After constructing a 1C-FLP for a node, the remaining layers that cannot fill a vehicle (and therefore cannot form a 1C-FLP) constitute a 1C-SP, which stands for one customer segment pattern. As a result, each node will always have one 1C-SP and potentially zero, one, or multiple 1C-FLPs.

Following the construction of 1C-FLP and 1C-SP, the two customers segment pattern (2C-SP) is developed based on the 1C-SP. Initially, a customer pair  $(c_1, c_2)$  is selected, and a basic layer is chosen from each of the two customers' 1C-SPs to form a mixed layer. A customer pair may have various mixed layer construction schemes; for each scheme, a saving  $\delta = \text{depth}(1C - SP(c_1)) + \text{depth}(1C - SP(c_2)) - \text{depth}(2C - SP(c_1, c_2))$  is calculated. The construction

scheme with the maximum saving  $\delta$  is selected as the final 2C-SP solution for the customer pair. For additional information on 1C-FLP, 1C-SP, and 2C-SP, please refer to Bortfeldt and Yi [5].

## Appendix C: Number of fitness evaluations for SDVRLH2

This section describes how to calculate the lower bound of the fitness evaluations (FEs) for SDVRLH2, supporting Sect. 4 of the main text.

In Bortfeldt and Yi [5], detailed information on the number of FEs used by SDVRLH2 is absent. Hence, we estimated a lower bound for the FEs based on the description and parameters of the algorithm. To maintain clarity, we adopted the symbols and notation from Bortfeldt and Yi [5] in our explanation. Symbols  $nps$ ,  $nbh$ , and  $niter\_wimpr$  denote the number of outer iterations, the number of inner iterations, and the iteration threshold for improvement termination (i.e., the iteration terminates if the algorithm does not produce a better solution after  $niter\_wimpr$  consecutive iterations.), respectively.  $n_{swap}$ ,  $n_{2opt}$ , and  $n_{3opt}$  correspond to the neighborhood sizes of different operators. Thus, the minimal FEs for SDVRLH2 are calculated as:  $nps \times nbh \times niter\_wimpr \times (n_{swap} + n_{2opt} + n_{3opt})$ .

## Appendix D: More experimental results

This appendix section provides the raw data for each set of experiments, supporting Sect. 4 of the main text.

Tables 17 and 18 detail the experimental results of SDVRLH2 as reported in Bortfeldt and Yi [5]. Lacking specific details on the number of fitness evaluations (FEs) utilized by SDVRLH2, we deduced a lower bound for the FEs, with the calculation method outlined in Section C.

Tables 21 and 22 illustrate the average and best outcomes for our algorithm when the number of vehicles is used as the primary objective (Ov), across 30 runs.

Tables 19 and 20 illustrate the average and best outcomes for our algorithm when total travel distance ( $ttd$ ) is used as the primary objective, across 30 runs.

Additionally, Table 7 clarifies the notations A1-A4 utilized in our ablation studies. The outcomes of these algorithms are subsequently depicted in Tables 23–28.

**Table 17** Experimental results of SDVRLH2 on small-scale instances (number of nodes <100)

| Instances (small) | Avg   |          | Best  |          | CPU time<br>Avg | # FE<br>Lower bound |
|-------------------|-------|----------|-------|----------|-----------------|---------------------|
|                   | # v   | ttd      | # v   | ttd      |                 |                     |
| B-Y1              | 37    | 2081.6   | 37    | 2081.6   | 152.1           | 481696              |
| B-Y2              | 15.5  | 1055.29  | 15    | 1044.2   | 96              | 481696              |
| B-Y3              | 37    | 2071     | 37    | 2071     | 156.2           | 481696              |
| B-Y4              | 18    | 1184.62  | 18    | 1149.6   | 94.7            | 481696              |
| B-Y5              | 54    | 2961.2   | 54    | 2961.2   | 760.5           | 1057296             |
| B-Y6              | 25    | 1570.72  | 25    | 1541.4   | 448.6           | 1057296             |
| B-Y7              | 57    | 3215.8   | 57    | 3215.8   | 838.5           | 1057296             |
| B-Y8              | 26    | 1645.03  | 26    | 1606.8   | 503.5           | 1057296             |
| Sha1              | 3     | 562.6    | 3     | 562.6    | 0.3             | 1816                |
| Sha2              | 9     | 2786.6   | 9     | 2786.6   | 11.5            | 5900                |
| Sha3              | 4     | 396.48   | 4     | 393      | 1.1             | 17936               |
| Sha4              | 4     | 350.33   | 4     | 349.7    | 3               | 22500               |
| Sha5              | 6     | 1451.27  | 6     | 1447.6   | 7.8             | 13140               |
| Sha6              | 8     | 482.13   | 8     | 479.6    | 11.2            | 39856               |
| Sha7              | 12    | 1707.48  | 12    | 1694.8   | 16.8            | 45340               |
| Sha8              | 8     | 357.67   | 8     | 352.7    | 5.7             | 30080               |
| Sha9              | 13    | 944.28   | 13    | 928.7    | 16.9            | 116240              |
| Sha10             | 14    | 1605.36  | 14    | 1590.1   | 20.5            | 115440              |
| Sha11             | 26.2  | 523.21   | 26    | 521.5    | 150.7           | 575640              |
| Sha12             | 17    | 2968.08  | 17    | 2942.1   | 48.1            | 250736              |
| Sha13             | 27.7  | 3147.93  | 27    | 3097.3   | 594.1           | 907644              |
| SD1               | 5     | 5238.5   | 5     | 4941.8   | 0.9             | 13680               |
| SD2               | 14    | 11929.8  | 14    | 11620.2  | 8.4             | 99096               |
| SD3               | 23    | 16302.1  | 23    | 15800.9  | 36.4            | 154480              |
| SD4               | 14    | 12826.7  | 14    | 12655.5  | 27              | 225400              |
| SD5               | 17    | 14513.8  | 17    | 14513.8  | 33.5            | 315040              |
| SD6               | 17    | 15883.7  | 17    | 15332.5  | 70.2            | 272960              |
| SD7               | 13    | 12206.6  | 13    | 11804.4  | 73              | 342776              |
| SD8               | 23    | 19709.8  | 23    | 19225.8  | 123.3           | 394140              |
| SD9               | 20    | 16512.9  | 20    | 16015    | 220.5           | 551820              |
| SD10              | 18    | 16432.6  | 18    | 15874.5  | 273.8           | 569236              |
| SD11              | 21.1  | 18871.3  | 21    | 18593.2  | 1822.7          | 2197140             |
| Avg               | 19.0  | 6046.8   | 18.9  | 5912.4   | 207.1           | 419812.6            |
| Total             | 606.5 | 193496.5 | 605.0 | 189195.5 | 6627.5          | 13434004.0          |

# v = the number of vehicles used; ttd=total travel distance; # FE= the number of fitness evaluations

**Table 18** Experimental results of SDVRLH2 on large-scale instances ( $100 \leq \text{number of nodes} \leq 200$ )

| Instances (large) | Avg    |          | Best   |          | CPU time<br>Avg | # FE<br>Lower bound |
|-------------------|--------|----------|--------|----------|-----------------|---------------------|
|                   | # v    | ttd      | # v    | ttd      |                 |                     |
| B-Y9              | 71     | 4014.9   | 71     | 4014.9   | 1802.6          | 2041396             |
| B-Y10             | 34.7   | 2133.53  | 34     | 2077.3   | 1814.8          | 2041396             |
| B-Y11             | 76     | 4386.7   | 76     | 4386.6   | 1803.1          | 2041396             |
| B-Y12             | 34     | 2173.56  | 34     | 2151.3   | 1762            | 2041396             |
| B-Y13             | 109    | 6043.42  | 109    | 6022.4   | 1811.7          | 5332096             |
| B-Y14             | 48     | 2942.14  | 48     | 2888.3   | 1802.8          | 5332096             |
| B-Y15             | 117.6  | 6477.68  | 117    | 6451.1   | 1804.6          | 5332096             |
| B-Y16             | 51.9   | 3089.77  | 51     | 3177.8   | 1804.1          | 5332096             |
| B-Y17             | 148    | 8462.34  | 148    | 8456.4   | 1808.1          | 11464784            |
| B-Y18             | 65.6   | 3676.74  | 65     | 3647.1   | 1810.8          | 11464784            |
| B-Y19             | 154    | 9298.8   | 154    | 9296.4   | 1808.9          | 11464784            |
| B-Y20             | 74.6   | 4298.52  | 73     | 4351.4   | 1805.6          | 11464784            |
| Sha14             | 44.2   | 1574.4   | 44     | 1522.5   | 1802.3          | 3325320             |
| Sha15             | 71.9   | 4866.99  | 71     | 4795.4   | 3625.9          | 17234796            |
| SD12              | 44     | 37261.3  | 44     | 36311.5  | 1803.6          | 3713304             |
| SD13              | 25     | 22047.5  | 25     | 21694.3  | 1802.2          | 3713304             |
| Avg               | 73.1   | 7671.8   | 72.8   | 7577.8   | 1917.1          | 6458739.3           |
| Total             | 1169.5 | 122748.3 | 1164.0 | 121244.7 | 30673.1         | 103339828.0         |

# v = the number of vehicles used; ttd=total travel distance; # FE= the number of fitness evaluations

**Table 19** Experimental results of Ov on small-scale instances (number of nodes < 100)

| Instances (small) | Avg  |          | Best |          | CPU time<br>Avg | # FE<br>Avg |
|-------------------|------|----------|------|----------|-----------------|-------------|
|                   | # v  | ttd      | # v  | ttd      |                 |             |
| B-Y1              | 35.9 | 2660.23  | 35   | 2527.3   | 19.2            | 869.37      |
| B-Y2              | 15   | 1578.01  | 15   | 1498.8   | 8.5             | 925.80      |
| B-Y3              | 36.0 | 2486.69  | 36   | 2329.6   | 26.9            | 1307.13     |
| B-Y4              | 19.3 | 1832.08  | 19   | 1714.8   | 12.7            | 938.23      |
| B-Y5              | 52   | 3895.41  | 52   | 3776.2   | 47.7            | 1847.70     |
| B-Y6              | 24.7 | 2507.63  | 24   | 2410.2   | 19.9            | 1882.43     |
| B-Y7              | 55   | 3975.61  | 55   | 3825.8   | 59.5            | 1699.37     |
| B-Y8              | 27.0 | 2802.74  | 27   | 2673.4   | 25.9            | 1723.30     |
| Sha1              | 3    | 575.97   | 3    | 570.2    | 0.6             | 38.20       |
| Sha2              | 9.2  | 2913.71  | 9    | 2853.1   | 0.5             | 74.43       |
| Sha3              | 4    | 417.66   | 4    | 398.2    | 0.5             | 89.70       |
| Sha4              | 4    | 418.23   | 4    | 374.8    | 0.8             | 115.13      |
| Sha5              | 5    | 1359.03  | 5    | 1329.4   | 0.8             | 108.97      |
| Sha6              | 7    | 636.36   | 7    | 595.9    | 1.0             | 140.43      |
| Sha7              | 11.3 | 1809.92  | 11   | 1682.72  | 1.6             | 170.40      |
| Sha8              | 8    | 375.34   | 8    | 361.1    | 1.5             | 210.70      |
| Sha9              | 13   | 1145.18  | 13   | 1064.6   | 2.4             | 309.50      |
| Sha10             | 14   | 1853.21  | 14   | 1748     | 2.7             | 394.97      |
| Sha11             | 26   | 638.76   | 26   | 612      | 5.7             | 722.40      |
| Sha12             | 15   | 3583.87  | 15   | 3393.8   | 3.1             | 497.27      |
| Sha13             | 27   | 4024.55  | 27   | 3874.59  | 10.8            | 1374.43     |
| SD1               | 5    | 6154.90  | 5    | 5714.08  | 1.0             | 108.20      |
| SD2               | 13   | 14354.68 | 13   | 13200.62 | 2.9             | 360.60      |
| SD3               | 23   | 18748.91 | 23   | 18026.55 | 3.2             | 523.43      |

**Table 19** continued

| Instances (small) | Avg   |          | Best  |          | CPU time<br>Avg | # FE<br>Avg |
|-------------------|-------|----------|-------|----------|-----------------|-------------|
|                   | # v   | ttd      | # v   | ttd      |                 |             |
| SD4               | 13    | 18323.46 | 13    | 16883.47 | 3.4             | 579.20      |
| SD5               | 19    | 22620.52 | 19    | 22475.93 | 2.1             | 427.13      |
| SD6               | 15    | 23381.75 | 15    | 21959.31 | 36.1            | 583.10      |
| SD7               | 10.5  | 17154.40 | 10    | 15597.57 | 4.1             | 796.70      |
| SD8               | 20    | 25390.18 | 20    | 23898.87 | 25.2            | 740.10      |
| SD9               | 17    | 25866.86 | 17    | 23767.30 | 12.5            | 1074.07     |
| SD10              | 9     | 21755.58 | 9     | 19782.00 | 10.9            | 1061.70     |
| SD11              | 10    | 31373.40 | 10    | 29278.35 | 43.4            | 2231.23     |
| Avg               | 17.7  | 8331.7   | 17.6  | 7818.7   | 12.4            | 747.7       |
| Total             | 565.9 | 266614.8 | 563.0 | 250198.6 | 397.0           | 23925.3     |

Ov refers to our algorithm with # v as the primary objective. # v = the number of vehicles used; ttd=total travel distance; # FE= the number of fitness evaluations

**Table 20** Experimental results of Ov on large-scale instances ( $100 \leq \text{number of node} \leq 200$ )

| Instances (large) | Avg    |          | Best   |          | CPU time<br>Avg | # FE<br>Avg |
|-------------------|--------|----------|--------|----------|-----------------|-------------|
|                   | # v    | ttd      | # v    | ttd      |                 |             |
| B-Y9              | 67     | 5786.48  | 67     | 5452.80  | 473.5           | 3042.07     |
| B-Y10             | 31.7   | 3484.95  | 31     | 3196.00  | 240.5           | 3253.43     |
| B-Y11             | 71.9   | 5832.28  | 71     | 5493.30  | 574.0           | 2716.50     |
| B-Y12             | 35.0   | 4112.50  | 35     | 3941.80  | 187.8           | 2855.47     |
| B-Y13             | 101.9  | 8552.79  | 101    | 8095.70  | 1055.4          | 6353.80     |
| B-Y14             | 46     | 4939.79  | 46     | 4704.30  | 250.4           | 6715.97     |
| B-Y15             | 107.1  | 8754.55  | 107    | 8359.20  | 845.3           | 5845.30     |
| B-Y16             | 53.0   | 5777.26  | 53     | 5469.90  | 854.2           | 5653.77     |
| B-Y17             | 135.6  | 11083.66 | 135    | 10486.30 | 1331.1          | 9679.97     |
| B-Y18             | 59     | 6712.65  | 59     | 6236.20  | 726.0           | 10297.60    |
| B-Y19             | 139.9  | 11217.65 | 139    | 10812.20 | 2370.8          | 9002.93     |
| B-Y20             | 73.7   | 7746.30  | 73     | 7364.90  | 1056.0          | 10077.10    |
| Sha14             | 42     | 2650.70  | 42     | 2458.00  | 166.2           | 2290.43     |
| Sha15             | 66.9   | 9491.36  | 66     | 8967.30  | 1446.8          | 8772.87     |
| SD12              | 44     | 50039.03 | 44     | 47009.49 | 299.7           | 5473.57     |
| SD13              | 22     | 45857.84 | 22     | 43681.07 | 1590.7          | 4046.73     |
| Avg               | 68.6   | 12002.5  | 68.2   | 11358.0  | 841.8           | 6004.8      |
| Total             | 1096.9 | 192039.8 | 1091.0 | 181728.5 | 13468.4         | 96077.5     |

Ov refers to our algorithm with # v as the primary objective. # v = the number of vehicles used; ttd=total travel distance; # FE= the number of fitness evaluations

**Table 21** Experimental results of Ot on small-scale instances (number of nodes <100)

| Instances (small) | Avg   |          | Best  |          | CPU time<br>Avg | # FE<br>Avg |
|-------------------|-------|----------|-------|----------|-----------------|-------------|
|                   | # v   | ttd      | # v   | ttd      |                 |             |
| B-Y1              | 36    | 2607.72  | 36    | 2527.3   | 26.0            | 1185.3      |
| B-Y2              | 15    | 1544.11  | 15    | 1458.5   | 12.9            | 1311.9      |
| B-Y3              | 37    | 2449.05  | 37    | 2377.7   | 30.6            | 1524.8      |
| B-Y4              | 20    | 1758.83  | 20    | 1683     | 19.6            | 1289.6      |
| B-Y5              | 52.8  | 3868.88  | 52    | 3783.9   | 60.5            | 2472.9      |
| B-Y6              | 25    | 2440.96  | 25    | 2384.5   | 27.6            | 2376.7      |
| B-Y7              | 55.2  | 3937.7   | 55    | 3815     | 78.6            | 2529.4      |
| B-Y8              | 27.9  | 2747.16  | 27    | 2625.6   | 34.3            | 2136.9      |
| Sha1              | 3     | 572.16   | 3     | 570.2    | 1.0             | 66.1        |
| Sha2              | 9.1   | 2889.35  | 9     | 2853.1   | 1.0             | 119.9       |
| Sha3              | 4     | 412.38   | 4     | 398.2    | 0.7             | 131         |
| Sha4              | 4     | 419.73   | 4     | 374.8    | 1.5             | 189.5       |
| Sha5              | 5     | 1357.37  | 5     | 1329.4   | 1.4             | 172.5       |
| Sha6              | 7     | 627.8    | 7     | 596.7    | 1.6             | 224.9       |
| Sha7              | 11.6  | 1789.275 | 11    | 1682.72  | 2.2             | 265.2       |
| Sha8              | 8     | 371.377  | 8     | 364.37   | 2.2             | 317.8       |
| Sha9              | 13    | 1114.87  | 13    | 1064.6   | 3.5             | 489.6       |
| Sha10             | 14    | 1825.4   | 14    | 1748     | 3.1             | 581.3       |
| Sha11             | 26    | 629      | 26    | 608.1    | 6.9             | 1095.8      |
| Sha12             | 15.1  | 3561.36  | 15    | 3474.1   | 3.9             | 687.2       |
| Sha13             | 27.6  | 3985.27  | 27    | 3915.6   | 10.1            | 1795.1      |
| SD1               | 5     | 6117.48  | 5     | 5845.84  | 1.7             | 151.8       |
| SD2               | 13    | 13897.11 | 13    | 13251.72 | 5.3             | 573         |
| SD3               | 23    | 18336.13 | 23    | 17823.11 | 4.9             | 694.5       |
| SD4               | 13.2  | 18247.84 | 13    | 17614.07 | 5.6             | 830.2       |
| SD5               | 19    | 22573.92 | 19    | 22475.93 | 3.8             | 695.4       |
| SD6               | 15    | 22788.04 | 15    | 21959.31 | 49.5            | 846.9       |
| SD7               | 11    | 16262.30 | 11    | 15458.55 | 5.9             | 1085.4      |
| SD8               | 20    | 25115.09 | 20    | 24359.68 | 31.4            | 974         |
| SD9               | 17.3  | 25185.84 | 17    | 23839.37 | 17.4            | 1607.1      |
| SD10              | 9.6   | 20727.10 | 9     | 19324.46 | 15.7            | 1518.5      |
| SD11              | 10    | 30590.65 | 10    | 29169.05 | 54.3            | 3064.2      |
| Avg               | 17.9  | 8148.5   | 17.8  | 7836.1   | 16.4            | 1031.4      |
| Total             | 572.4 | 260751.3 | 568.0 | 250756.5 | 524.7           | 33004.4     |

Ot refers to our algorithm with ttd as the primary objective. # v = the number of vehicles used; ttd=total travel distance; # FE= the number of fitness evaluations

**Table 22** Experimental results of Ot on large-scale instances ( $100 \leq \text{number of node} \leq 200$ )

| Instances (large) | Avg    |          | Best   |          | CPU time<br>Avg | # FE<br>Avg |
|-------------------|--------|----------|--------|----------|-----------------|-------------|
|                   | # v    | ttd      | # v    | ttd      |                 |             |
| B-Y9              | 69     | 5180.27  | 69     | 5085.8   | 85.3            | 3746.4      |
| B-Y10             | 33     | 3225.55  | 33     | 3111.9   | 45.1            | 3503.2      |
| B-Y11             | 73.6   | 5360.52  | 73     | 5275.1   | 114.3           | 3751        |
| B-Y12             | 36.9   | 3813.84  | 36     | 3637.9   | 60.5            | 3359.3      |
| B-Y13             | 104.8  | 7675.13  | 104    | 7486.5   | 213.8           | 7193.8      |
| B-Y14             | 48     | 4530.47  | 48     | 4356.2   | 139.0           | 8142.3      |
| B-Y15             | 110.2  | 7882.79  | 110    | 7764.4   | 307.6           | 8293.4      |
| B-Y16             | 55.5   | 5174.41  | 55     | 5059.8   | 159.0           | 7544.9      |
| B-Y17             | 139.2  | 9999.87  | 139    | 9824.7   | 493.5           | 12873       |
| B-Y18             | 61.4   | 6006.99  | 61     | 5855.8   | 282.8           | 12843.8     |
| B-Y19             | 144.1  | 10020.58 | 144    | 9828.8   | 538.9           | 11515.9     |
| B-Y20             | 76.9   | 6926.28  | 76     | 6806.1   | 366.1           | 12342       |
| Sha14             | 43     | 2423.45  | 43     | 2328.4   | 21.3            | 3391.6      |
| Sha15             | 68     | 8755.14  | 68     | 8589.1   | 113.4           | 11135.7     |
| SD12              | 44     | 48608.00 | 44     | 46702.90 | 55.7            | 7122.5      |
| SD13              | 22     | 44749.23 | 22     | 43428.64 | 81.7            | 5362        |
| Avg               | 70.6   | 11270.8  | 70.3   | 10946.4  | 192.4           | 7632.6      |
| Total             | 1129.6 | 180332.5 | 1125.0 | 175142.0 | 3078.0          | 122120.8    |

Ot refers to our algorithm with ttd as the primary objective. # v = the number of vehicles used; ttd=total travel distance; # FE= the number of fitness evaluations

**Table 23** Experimental results of A1 on small-scale instances on (number of nodes <100)

| Instances (small) | Avg  |        | Best |        | CPU Time<br>Avg | # FE<br>Avg |
|-------------------|------|--------|------|--------|-----------------|-------------|
|                   | # v  | ttd    | # v  | ttd    |                 |             |
| B-Y1              | 35.2 | 2627.1 | 35   | 2501.8 | 8098.8          | 4510419.8   |
| B-Y2              | 15   | 1432.3 | 15   | 1380.1 | 5955.6          | 4320187.3   |
| B-Y3              | 36   | 2426.8 | 36   | 2264.3 | 7367.0          | 4130183.9   |
| B-Y4              | 19.1 | 1784.5 | 19   | 1660.7 | 6257.8          | 3957317.3   |
| B-Y5              | 52   | 3658.9 | 52   | 3597.5 | 53400.5         | 16894313.8  |
| B-Y6              | 24   | 2403.6 | 24   | 2205.4 | 41983.2         | 16660097.8  |
| B-Y7              | 55   | 3768.9 | 55   | 3693.1 | 53153.1         | 16629887.8  |
| B-Y8              | 27   | 2809.5 | 27   | 2690.8 | 38876.7         | 13989628.6  |
| Sha1              | 3    | 570.2  | 3    | 570.2  | 45.5            | 2494.2      |
| Sha2              | 9    | 2864.0 | 9    | 2853.1 | 60.1            | 11856.6     |
| Sha3              | 4    | 413.8  | 4    | 410    | 105.5           | 32315.4     |
| Sha4              | 4    | 384.4  | 4    | 374.8  | 102.7           | 50392.6     |
| Sha5              | 5    | 1348.6 | 5    | 1319.2 | 64.9            | 42038.8     |
| Sha6              | 7    | 610.6  | 7    | 595.3  | 166.7           | 102691.4    |
| Sha7              | 11   | 1790.9 | 11   | 1721.2 | 215.1           | 127123      |
| Sha8              | 8    | 364.7  | 8    | 356.4  | 105.2           | 155975.3    |
| Sha9              | 13   | 1076.8 | 13   | 1067.4 | 359.7           | 602298      |
| Sha10             | 14   | 1728.9 | 14   | 1715.9 | 474.4           | 859010.6    |
| Sha11             | 26   | 586.4  | 26   | 575.4  | 2690.3          | 3598298.3   |
| Sha12             | 15   | 3363.6 | 15   | 3304.6 | 907.3           | 1380265     |

**Table 23** continued

| Instances (small) | Avg   |            | Best |          | CPU Time<br>Avg | # FE<br>Avg |
|-------------------|-------|------------|------|----------|-----------------|-------------|
|                   | # v   | ttd        | # v  | ttd      |                 |             |
| Sha13             | 27    | 3824.6     | 27   | 3749.7   | 10733.9         | 9459955.4   |
| SD1               | 5     | 5680.5     | 5    | 5324.7   | 15.2            | 34052.2     |
| SD2               | 13    | 13263.3    | 13   | 12580.7  | 227.0           | 456475.9    |
| SD3               | 23    | 16710.0    | 23   | 16107.5  | 647.2           | 1148321.5   |
| SD4               | 13    | 16298.1    | 13   | 16008.0  | 1044.8          | 1673032.4   |
| SD5               | 19    | 22489.9    | 19   | 22467.1  | 1178.4          | 2012945.9   |
| SD6               | 15    | 22594.4    | 15   | 21466.3  | 1752.9          | 2271651.8   |
| SD7               | 10    | 15268.2    | 10   | 13798.6  | 1753.8          | 3200591     |
| SD8               | 20    | 24936.7    | 20   | 23906.6  | 2584.3          | 3175522.6   |
| SD9               | 17    | 23600.7    | 17   | 22837.7  | 4493.7          | 6233822.2   |
| SD10              | 9     | 18606.3    | 9    | 17602.8  | 4939.7          | 7810014.4   |
| SD11              | 10    | 25378.6    | 10   | 24862.8  | 37362.5         | 37421471.8  |
| Avg               | 17.6  | 7645.8     | 17.6 | 7361.6   | 8972.6          | 5092332.9   |
| Total             | 563.3 | 244665.682 | 563  | 235569.7 | 287123.4        | 162954653   |

# v = the number of vehicles used; ttd=total travel distance; # FE = the number of fitness evaluations

**Table 24** Experimental results of A1 on large-scale instances ( $100 \leq \text{number of node} \leq 200$ )

| Instances (large) | Avg    |          | Best   |          | CPU Time<br>Avg | # FE<br>Avg |
|-------------------|--------|----------|--------|----------|-----------------|-------------|
|                   | # v    | ttd      | # v    | ttd      |                 |             |
| B-Y9              | 68     | 5285.4   | 68     | 5010.1   | 4826.4          | 2266627     |
| B-Y10             | 32     | 3170.4   | 32     | 3017.1   | 3436.1          | 2291877     |
| B-Y11             | 73     | 5362.7   | 73     | 5244.8   | 5073.2          | 2188857     |
| B-Y12             | 36     | 3730.7   | 36     | 3648.1   | 3568.8          | 2234307     |
| B-Y13             | 103.9  | 7758.6   | 103    | 7641.6   | 43558.7         | 5891552     |
| B-Y14             | 47     | 4485.3   | 47     | 4384.3   | 30098.7         | 6112389.5   |
| B-Y15             | 109.1  | 8104.3   | 109    | 7815.8   | 24092.8         | 5901744.5   |
| B-Y16             | 54.5   | 5302.4   | 54     | 5009.7   | 18202.9         | 5996874.5   |
| B-Y17             | 138.5  | 10486.8  | 138    | 10063.8  | 74324.1         | 12879675    |
| B-Y18             | 60.7   | 6217.9   | 60     | 5758.5   | 51555.5         | 13196085    |
| B-Y19             | 142.8  | 10273.7  | 142    | 9971.2   | 134737.7        | 12780175    |
| B-Y20             | 75.8   | 7106.2   | 75     | 6929.6   | 105054.5        | 12967235    |
| Sha14             | 43     | 2332.6   | 43     | 2272.1   | 8202.8          | 3516423.1   |
| Sha15             | 68     | 8813.1   | 68     | 8602.3   | 139884.9        | 18527227    |
| SD12              | 44     | 39251.8  | 44     | 37889.9  | 32557.7         | 4934161     |
| SD13              | 22     | 45366.1  | 22     | 42760.4  | 42356.6         | 4031935     |
| Avg               | 69.9   | 10815.5  | 69.6   | 10376.2  | 45095.7         | 7232321.5   |
| Total             | 1118.3 | 173048.0 | 1114.0 | 166019.3 | 721531.5        | 115717144.6 |

# v = the number of vehicles used; ttd=total travel distance; # FE = the number of fitness evaluations

**Table 25** Experimental results of A2 on small-scale instances (number of nodes <100)

| Instances (small) | Avg  |           | Best |          | CPU Time<br>Avg | # FE<br>Avg |
|-------------------|------|-----------|------|----------|-----------------|-------------|
|                   | # v  | ttd       | # v  | ttd      |                 |             |
| B-Y1              | 35.9 | 2538.8    | 35   | 2463.1   | 589.5           | 97430.2     |
| B-Y2              | 15   | 1447.2    | 15   | 1409.6   | 347.5           | 94334.3     |
| B-Y3              | 36   | 2267.2    | 36   | 2238.4   | 622.1           | 97764.3     |
| B-Y4              | 19   | 1716.2    | 19   | 1635.9   | 451.9           | 94308       |
| B-Y5              | 52   | 3617.3    | 52   | 3554.7   | 773.2           | 292445.8    |
| B-Y6              | 24.1 | 2444.8    | 24   | 2255.1   | 419.4           | 290753.8    |
| B-Y7              | 55   | 3728.4    | 55   | 3628     | 868.0           | 292177.2    |
| B-Y8              | 27   | 2613.2    | 27   | 2565.6   | 495.8           | 289205.7    |
| Sha1              | 3    | 570.2     | 3    | 570.2    | 11.2            | 887.4       |
| Sha2              | 9    | 2855.5    | 9    | 2853.1   | 13.8            | 1873.1      |
| Sha3              | 4    | 399.7     | 4    | 398.2    | 19.6            | 4095.3      |
| Sha4              | 4    | 374.8     | 4    | 374.8    | 23.8            | 4762        |
| Sha5              | 5    | 1329.8    | 5    | 1323.6   | 15.9            | 3112.1      |
| Sha6              | 7    | 609.9     | 7    | 592.4    | 27.2            | 7052.5      |
| Sha7              | 11   | 1738.3    | 11   | 1680.55  | 32.7            | 7865.4      |
| Sha8              | 8    | 359.4     | 8    | 355.77   | 21.7            | 6807.6      |
| Sha9              | 13   | 1090.5    | 13   | 1070.4   | 46.6            | 19280       |
| Sha10             | 14   | 1769.1    | 14   | 1718.8   | 34.0            | 23981.2     |
| Sha11             | 26   | 604.1     | 26   | 586.2    | 133.3           | 83524.3     |
| Sha12             | 15   | 3420.3    | 15   | 3318.6   | 60.9            | 38253.1     |
| Sha13             | 27   | 3850.2    | 27   | 3802.3   | 238.2           | 188485.8    |
| SD1               | 5    | 5635.0    | 5    | 5575.7   | 24.6            | 3432.7      |
| SD2               | 13   | 13185.8   | 13   | 12780.6  | 75.7            | 17226.7     |
| SD3               | 23   | 17032.6   | 23   | 16419.4  | 82.9            | 30849.1     |
| SD4               | 13   | 16256.0   | 13   | 15676.8  | 101.6           | 42519.5     |
| SD5               | 19   | 22471.0   | 19   | 22467.1  | 99.4            | 54464.1     |
| SD6               | 15   | 22030.4   | 15   | 21191.1  | 757.7           | 59067.7     |
| SD7               | 10   | 15200.1   | 10   | 14487.4  | 128.1           | 69313.5     |
| SD8               | 20   | 24435.4   | 20   | 23542.4  | 589.0           | 82129.3     |
| SD9               | 17   | 23455.2   | 17   | 22865.8  | 377.2           | 126631.1    |
| SD10              | 9    | 19001.2   | 9    | 18268.9  | 367.3           | 150718.2    |
| SD11              | 10   | 27153.3   | 10   | 26094.2  | 1915.8          | 539316.9    |
| Avg               | 17.6 | 7662.53   | 17.6 | 7430.1   | 305.2           | 97314.6     |
| Total             | 564  | 245201.09 | 563  | 237764.7 | 9765.6          | 3114067.9   |

# v = the number of vehicles used; ttd=total travel distance; # FE = the number of fitness evaluations

**Table 26** Experimental results of A2 on large-scale instances ( $100 \leq \text{number of node} \leq 200$ )

| Instances (large) | Avg    |           | Best   |          | CPU Time<br>Avg | # FE<br>Avg |
|-------------------|--------|-----------|--------|----------|-----------------|-------------|
|                   | # v    | ttd       | # v    | ttd      |                 |             |
| B-Y9              | 68     | 5126.0    | 68     | 4851.6   | 1819.2          | 671605.2    |
| B-Y10             | 32     | 3073.9    | 32     | 2919.9   | 1186.1          | 672250.1    |
| B-Y11             | 73     | 5096.7    | 73     | 5002.6   | 2072.2          | 676170.8    |
| B-Y12             | 36     | 3561.8    | 36     | 3504.7   | 1342.5          | 669404.3    |
| B-Y13             | 104    | 7291.1    | 104    | 7194.3   | 19738.4         | 2248494.6   |
| B-Y14             | 47     | 4151.3    | 47     | 4081     | 13391.7         | 2248936.6   |
| B-Y15             | 109    | 7468.9    | 109    | 7371.5   | 20552.6         | 2246596.7   |
| B-Y16             | 54.9   | 4832.2    | 54     | 4715.8   | 15364.4         | 2243167.2   |
| B-Y17             | 138    | 9408.8    | 138    | 9307.9   | 44891.7         | 5257400.4   |
| B-Y18             | 60.5   | 5660.9    | 60     | 5297.1   | 24837.6         | 5249500.5   |
| B-Y19             | 143    | 9441.2    | 143    | 9370.4   | 64395.9         | 5243737.8   |
| B-Y20             | 76     | 6469.3    | 76     | 6397     | 50861.0         | 5235525.5   |
| Sha14             | 43     | 2198.7    | 43     | 2137.8   | 1998.5          | 716384.7    |
| Sha15             | 68     | 8009.1    | 68     | 7857.6   | 41719.9         | 5363288.9   |
| SD12              | 44     | 42988.9   | 44     | 41690.3  | 2887.5          | 1429910.7   |
| SD13              | 22     | 40337.0   | 22     | 39790.9  | 3492.9          | 1430011.6   |
| Avg               | 69.9   | 10319.73  | 69.8   | 10093.1  | 19409.5         | 2600149.1   |
| Total             | 1118.4 | 165115.64 | 1117.0 | 161490.3 | 310552.1        | 41602385.6  |

# v = the number of vehicles used; ttd=total travel distance; # FE = the number of fitness evaluations

**Table 27** Experiment results of A3 and A4 on small-scale instances (number of node < 100)

| Instances (small) | A3         |        | Best |        | # FE<br>avg | A4         |         | Best | # v | ttd     |
|-------------------|------------|--------|------|--------|-------------|------------|---------|------|-----|---------|
|                   | Avg<br># v | ttd    | # v  | ttd    |             | Avg<br># v | ttd     | # v  |     |         |
| B-Y1              | 35.9       | 2660.5 | 35   | 2535.1 | 831         | 35         | 2911.17 | 35   |     | 2588.50 |
| B-Y2              | 15         | 1589.5 | 15   | 1521.1 | 964.1       | 15         | 1573.92 | 15   |     | 1518.00 |
| B-Y3              | 36.1       | 2510.6 | 36   | 2409.2 | 1275.5      | 36         | 2499.25 | 36   |     | 2399.50 |
| B-Y4              | 19.2       | 1852.9 | 19   | 1746.4 | 895.3       | 19         | 1864.91 | 19   |     | 1746.40 |
| B-Y5              | 52         | 3890.2 | 52   | 3802.7 | 1890.4      | 51         | 4213.96 | 51   |     | 4120.50 |
| B-Y6              | 24.7       | 2547.2 | 24   | 2456.7 | 1809.1      | 24         | 2626.73 | 24   |     | 2438.10 |
| B-Y7              | 55         | 3982.1 | 55   | 3825.8 | 1908.7      | 54         | 4194.39 | 54   |     | 3941.70 |
| B-Y8              | 27.1       | 2823.7 | 27   | 2737   | 1698.8      | 27         | 2787.56 | 27   |     | 2673.40 |
| Sha1              | 3          | 576.9  | 3    | 570.2  | 36.3        | 3          | 576.87  | 3    |     | 570.20  |
| Sha2              | 9.2        | 2929.5 | 9    | 2876.8 | 76.9        | 9          | 2895.37 | 9    |     | 2860.70 |
| Sha3              | 4          | 417.2  | 4    | 402.3  | 79.3        | 4          | 417.16  | 4    |     | 402.30  |
| Sha4              | 4          | 423.0  | 4    | 374.8  | 127.1       | 4          | 422.40  | 4    |     | 374.80  |
| Sha5              | 5          | 1367.7 | 5    | 1329.4 | 110.1       | 5          | 1367.66 | 5    |     | 1329.40 |
| Sha6              | 7          | 635.2  | 7    | 610.8  | 145.1       | 7          | 634.62  | 7    |     | 610.80  |
| Sha7              | 11.2       | 1878.8 | 11   | 1781.9 | 181.8       | 11         | 1810.44 | 11   |     | 1682.72 |
| Sha8              | 8          | 373.6  | 8    | 364.5  | 230.3       | 8          | 373.16  | 8    |     | 364.37  |
| Sha9              | 13         | 1158.8 | 13   | 1092.8 | 315.5       | 13         | 1144.67 | 13   |     | 1064.60 |
| Sha10             | 14         | 1855.2 | 14   | 1769   | 398.3       | 13         | 1973.31 | 13   |     | 1790.30 |
| Sha11             | 26         | 641.8  | 26   | 612.4  | 752         | 26         | 639.46  | 26   |     | 612.00  |
| Sha12             | 15         | 3664.5 | 15   | 3502.8 | 454.3       | 15         | 3620.14 | 15   |     | 3449.70 |
| Sha13             | 27         | 4068.7 | 27   | 3997.2 | 1323.2      | 26.2       | 4184.43 | 26   |     | 3974.60 |
| SD1               | 5          | 6201.0 | 5    | 5845.8 | 96.6        | 5          | 6153.02 | 5    |     | 5845.84 |

**Table 27** continued

| Instances (small) | A3    |            |       |            |             | A4    |            |      |            |
|-------------------|-------|------------|-------|------------|-------------|-------|------------|------|------------|
|                   | Avg   |            | Best  |            | # FE<br>avg | Avg   |            | Best |            |
|                   | # v   | <i>ttd</i> | # v   | <i>ttd</i> |             | # v   | <i>ttd</i> | # v  | <i>ttd</i> |
| SD2               | 13    | 14564.2    | 13    | 13719.8    | 356.3       | 13    | 14431.29   | 13   | 13719.84   |
| SD3               | 23    | 18655.6    | 23    | 18031.2    | 499.8       | 23    | 18556.22   | 23   | 18026.55   |
| SD4               | 13    | 18353.2    | 13    | 17134.2    | 558.3       | 13    | 18173.17   | 13   | 17134.22   |
| SD5               | 19    | 22625.0    | 19    | 22475.9    | 455.6       | 19    | 22619.56   | 19   | 22475.93   |
| SD6               | 15    | 23601.9    | 15    | 22085.6    | 534.1       | 15    | 23206.14   | 15   | 21959.31   |
| SD7               | 10.6  | 17070.1    | 10    | 15597.6    | 880.5       | 10    | 17352.92   | 10   | 15605.32   |
| SD8               | 20    | 25618.2    | 20    | 24844.2    | 673.3       | 20    | 25270.13   | 20   | 24434.39   |
| SD9               | 17    | 25749.5    | 17    | 24191.8    | 1220.5      | 17    | 25432.79   | 17   | 23767.30   |
| SD10              | 9     | 21372.8    | 9     | 19782.0    | 1161.7      | 9     | 21300.45   | 9    | 19782.00   |
| SD11              | 10    | 31385.7    | 10    | 29885.1    | 2198.3      | 10    | 31286.05   | 10   | 29885.07   |
| Avg               | 17.7  | 8345.1     | 17.6  | 7934.8     | 754.3       | 17.5  | 8328.54    | 17.5 | 7910.89    |
| Total             | 566.0 | 267044.7   | 563.0 | 253912.1   | 24138.1     | 559.2 | 266513.30  | 559  | 253148.35  |

# v = the number of vehicle used; ttd=total travel distance; # FE= the number of fitness evaluations

**Table 28** Experimental results of A3 and A4 on large-scale instances ( $100 \leq \text{number of node} \leq 200$ )

| Instances (large) | A3     |          |        |          |             | A4     |           |        |           |
|-------------------|--------|----------|--------|----------|-------------|--------|-----------|--------|-----------|
|                   | Avg    |          | Best   |          | # FE<br>avg | Avg    |           | Best   |           |
|                   | # v    | ttd      | # v    | ttd      |             | # v    | ttd       | # v    | ttd       |
| B-Y9              | 68.5   | 5265.3   | 68     | 5138.8   | 3217.9      | 67     | 5800.54   | 67     | 5594.90   |
| B-Y10             | 32.2   | 3422.3   | 32     | 3246.6   | 2675.2      | 31.9   | 3444.00   | 31     | 3196.00   |
| B-Y11             | 73     | 5476.5   | 73     | 5327.4   | 2708.2      | 71.9   | 5862.50   | 71     | 5493.30   |
| B-Y12             | 36     | 3886.6   | 36     | 3762.4   | 2948.8      | 35     | 4161.97   | 35     | 4000.90   |
| B-Y13             | 104.1  | 7796.3   | 104    | 7595.8   | 5910.1      | 102    | 8450.96   | 102    | 8095.70   |
| B-Y14             | 47     | 4683.0   | 47     | 4546.6   | 6565        | 46     | 4930.62   | 46     | 4708.30   |
| B-Y15             | 109.6  | 8029.7   | 109    | 7798     | 5762.1      | 107.1  | 8807.14   | 107    | 8486.20   |
| B-Y16             | 55     | 5277.7   | 55     | 5070.6   | 5702.1      | 53     | 5751.77   | 53     | 5469.90   |
| B-Y17             | 138.3  | 10288.9  | 138    | 10038.4  | 9403.3      | 135.8  | 10900.13  | 135    | 10486.30  |
| B-Y18             | 61     | 6219.4   | 61     | 6104.8   | 9521.7      | 59     | 6769.18   | 59     | 6508.60   |
| B-Y19             | 143.2  | 10271.9  | 142    | 9956.2   | 8433.3      | 139.8  | 11392.56  | 139    | 10812.20  |
| B-Y20             | 75.9   | 7152.3   | 75     | 6910.9   | 9738.9      | 73.8   | 7678.03   | 73     | 7410.70   |
| Sha14             | 43     | 2523.4   | 43     | 2401.9   | 2189.3      | 42     | 2681.81   | 42     | 2458.00   |
| Sha15             | 68     | 8899.8   | 68     | 8722.5   | 9077.1      | 66.9   | 9565.31   | 66     | 9114.80   |
| SD12              | 44     | 51879.3  | 44     | 48854.5  | 5062.1      | 44     | 50457.49  | 44     | 47488.38  |
| SD13              | 22     | 45712.8  | 22     | 44263.0  | 4180.5      | 22     | 45388.23  | 22     | 43840.02  |
| Avg               | 70.1   | 11674.1  | 69.8   | 11233.7  | 5818.5      | 68.6   | 12002.64  | 68.3   | 11447.76  |
| Total             | 1120.8 | 186785.1 | 1117.0 | 179738.4 | 93095.6     | 1097.2 | 192042.24 | 1092.0 | 183164.19 |

# v = the number of vehicles used; ttd=total travel distance; # FE= the number of fitness evaluations

**Author Contributions** Han Zhang: Conducted the literature review, developed the algorithm, implemented the coding, designed and executed the experiments, analyzed the experimental data, and prepared the initial draft of the manuscript. Qing Li: Provided critical insights during the research development and contributed to refining the research methodology. Xin Yao: Supervised the project by offering strategic guidance on algorithm development, experimental setup, and manuscript structuring. Also contributed to the refinement of the manuscript through critical feedback and language polishing.

**Funding** Open Access Publishing Support Fund provided by Lingnan University This work was supported by the National Key R&D Program of China (Grant No. 2023YFE0106300), the National Natural Science Foundation of China (Grant No. 62250710682), the Guangdong Provincial Key Laboratory (Grant No. 2020B121201001), and the Program for Guangdong Introducing Innovative and Entrepreneurial Teams (Grant No. 2017ZT07X386).

## Declarations

**Competing interests** The authors have no conflicts of interest to declare that are relevant to the content of this article.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

- Aleman RE, Zhang X, Hill RR (2010) An adaptive memory algorithm for the split delivery vehicle routing problem. *J Heuristics* 16(3):441–473
- Beasley JE (1983) Route first-cluster second methods for vehicle routing. *Omega* 11(4):403–408
- Blazinski A, Misevicius A (2011) Combining 2-opt, 3-opt and 4-opt with k-swap-kick perturbations for the traveling salesman problem. Kaunas University of Technology, Department of Multimedia Engineering, Studentu St pp 50–401. [https://isd.ktu.lt/it2011/material/Proceedings/1\\_AI\\_5.pdf](https://isd.ktu.lt/it2011/material/Proceedings/1_AI_5.pdf)
- Bortfeldt A, Gehring H (2001) A hybrid genetic algorithm for the container loading problem. *Eur J Oper Res* 131(1):143–161
- Bortfeldt A, Yi J (2020) The split delivery vehicle routing problem with three-dimensional loading constraints. *Eur J Oper Res* 282(2):545–558
- Ceschia S, Schaerf A (2013) Local search for a multi-drop multi-container loading problem. *J Heuristics* 19(2):275–294
- Ceschia S, Schaerf A, Stützle T (2013) Local search techniques for a routing-packing problem. *Comput Indus Eng* 66(4):1138–1149
- Chen Z, Yang M, Guo Y et al (2020) The split delivery vehicle routing problem with three-dimensional loading and time windows constraints. *Sustainability* 12(17):6987
- Duhamel C, Lacomme P, Prodhon C (2011) Efficient frameworks for greedy split and new depth first search split procedures for routing problems. *Comput Operations Res* 38(4):723–739
- El-Hajj R, Guibadj RN, Moukrim A et al (2020) A PSO based algorithm with an efficient optimal split procedure for the multiperiod vehicle routing problem with profit. *Ann Oper Res* 291(1):281–316
- Gendreau M, Iori M, Laporte G et al (2006) A tabu search algorithm for a routing and container loading problem. *Transp Sci* 40(3):342–350
- Hu ZH, Zhao Y, Tao S et al (2015) Finished-vehicle transporter routing problem solved by loading pattern discovery. *Ann Oper Res* 234(1):37–56
- Khebbache-Hadji S, Prins C, Yalaoui A et al (2013) Heuristics and memetic algorithm for the two-dimensional loading capacitated vehicle routing problem with time windows. *CEJOR* 21(2):307–336
- Lacomme P, Toussaint H, Duhamel C (2013) A GRASP×ELS for the vehicle routing problem with basic three-dimensional loading constraints. *Eng Appl Artif Intell* 26(8):1795–1810
- Li X, Yuan M, Chen D et al (2018) A data-driven three-layer algorithm for split delivery vehicle routing problem with 3D container loading constraint. In: Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining, pp 528–536
- Liu F, Zhang Q, Zhu Q et al (2024) Machine learning assisted multiobjective evolutionary algorithm for routing and packing. *IEEE Trans Evolut Comput Early Access*. <https://doi.org/10.1109/TEVC.2024.3357819>
- Liu S, Tang K, Yao X (2021) Memetic search for vehicle routing with simultaneous pickup-delivery and time windows. *Swarm Evol Comput* 66:100927. <https://doi.org/10.1016/j.swevo.2021.100927>
- Lu X, Tang K, Menzel S et al (2020) A competitive co-evolutionary optimization method for the dynamic vehicle routing problem. In: 2020 IEEE symposium series on computational intelligence (SSCI), pp 305–312. <https://doi.org/10.1109/SSCI47803.2020.9308367>
- Moura A (2008) A multi-objective genetic algorithm for the vehicle routing with time windows and loading problem. In: Intelligent decision support. Springer, p 187–201
- Moura A, Oliveira JF (2009) An integrated approach to the vehicle routing and container loading problems. *OR Spectrum* 31(4):775–800
- Pei J, Hu C, Liu J et al (2021) Bi-objective splitting delivery VRP with loading constraints and restricted access. In: 2021 IEEE symposium series on computational intelligence (SSCI), IEEE, pp 01–09
- Pei J, Mei Y, Liu J et al (2022) An investigation of adaptive operator selection in solving complex vehicle routing problem. In: Khanna S, Cao J, Bai Q et al (eds) PRICAI 2022: Trends in artificial intelligence. Springer Nature Switzerland, Cham, pp 562–573
- Prins C (2004) A simple and effective evolutionary algorithm for the vehicle routing problem. *Comput Operat Res* 31(12):1985–2002
- Prins C, Lacomme P, Prodhon C (2014) Order-first split-second methods for vehicle routing problems: A review. *Transp Res Part C Emerging Technol* 40:179–200
- Rajaei M, Moslehi G, Reisi-Nafchi M (2022) The split heterogeneous vehicle routing problem with three-dimensional loading constraints on a large scale. *Eur J Oper Res* 299(2):706–721. <https://doi.org/10.1016/j.ejor.2021.08.025>
- Reil S, Bortfeldt A, Mönch L (2018) Heuristics for vehicle routing problems with backhauls, time windows, and 3D loading constraints. *Eur J Oper Res* 266(3):877–894
- Tan KC, Lee LH, Zhu Q et al (2001) Heuristic methods for vehicle routing problem with time windows. *Artif Intell Eng* 15(3):281–295

28. Tinós R, Helsgaun K, Whitley D (2018) Efficient recombination in the Lin–Kernighan–Helsgaun traveling salesman heuristic. In: Auger A, Fonseca CM, Lourenço N et al (eds) *Parallel problem solving from nature—PPSN XV*. Springer International Publishing, Cham, pp 95–107
29. Turkey A, Moser I, Aleti A (2017) An iterated local search with guided perturbation for the heterogeneous fleet vehicle routing problem with time windows and three-dimensional loading constraints. In: *Australasian conference on artificial life and computational intelligence*, Springer, pp 279–290
30. 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 Comput Intell Mag* 9(4):18–30
31. Yi J, Bortfeldt A (2018) The capacitated vehicle routing problem with three-dimensional loading constraints and split delivery—a case study. In: Fink A, Fügenshuh A, Geiger MJ (eds) *Operations research proceedings 2016*. Springer International Publishing, Cham, pp 351–356
32. Zhang Z, Wei L, Lim A (2015) An evolutionary local search for the capacitated vehicle routing problem minimizing fuel consumption under three-dimensional loading constraints. *Transp Res Part B Methodol* 82:20–35

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.