European Journal of Operational Research 299 (2022) 706–721

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Innovative Applications of O.R.

The split heterogeneous vehicle routing problem with three-dimensional loading constraints on a large scale

Maryam Rajaei, Ghasem Moslehi*, Mohammad Reisi-Nafchi

Department of Industrial and Systems Engineering, Isfahan University of Technology, Isfahan 84156-83111, Iran

A R T I C L E I N F O

Article history:
Received 20 August 2020
Accepted 14 August 2021
Available online 26 August 2021

Keywords:
Routing
Three-dimensional loading
Split delivery
Column generation

A B S T R A C T

This paper introduces a complex real-world problem on a large scale that combines the split delivery heterogeneous vehicle routing problem with three-dimensional loading and considers new routing and loading constraints. The problem aims to find a set of routes with minimum transportation cost by satisfying the demand of all customers and loading constraints. We propose a column generation-based heuristic algorithm that employs three heuristic algorithms to solve the subproblem and a hybrid algorithm for loading. Computational experiments on the literature instances indicate that the proposed algorithm can produce very near solutions to the literature algorithm's solutions in significantly shorter running times. This paper introduces real-world instances with 230 to 850 customers, several times larger than the existing benchmarks. The results of the proposed algorithm for real-world instances are effective, and compared to the current situation, it leads to a reduction of 20% in the average total cost of transportation. Also, the average number of vehicles is reduced by 36%.

1. Introduction

A major challenge for distributing companies is to load shipments efficiently into vehicles and deliver them to customers. Gendreau, Iori, Laporte and Martello (2006) introduced the capacitated vehicle routing problem with three-dimensional loading constraints. This problem seeks to find a set of routes that meet demand of all customers while minimizing transportation costs. Vehicles are homogeneous, and each customer is visited only once. Bortfeldt and Yi (2020) categorized the split delivery vehicle routing problem with three-dimensional loading constraints (3L-SDVRP), in which the customer can be served with more than one vehicle. In general, this problem has two variants, forced splitting (3L-SDVRP-f) and optional splitting (3L-SDVRP-o). In forced splitting, only the demand of customers whose shipments cannot be loaded in one vehicle is split. In optional splitting to reduce costs and to use most of the loading space, the demand of other customers may also be split.

This paper's problem is a complex large problem in the real world that can be considered as an integrated vehicle routing and three-dimensional loading problem. In this problem, the shipments of some customers cannot be sent with just one vehicle, and reducing the dispersion of the customer boxes in different vehicles is considered. So we are facing a forced-splitting variant, and optional splitting is not allowed. Vehicles are heterogeneous, and customers can choose the type of vehicle. Customer boxes must be packed according to loading constraints, including geometric feasibility, orientation, stacking, vertical stability, and last-in-first-out (LIFO), which have been considered in most research in the literature. In addition, allocation-connectivity, allocation-separation, reachability, and complexity constraints for loading and unloading the boxes are considered. According to our knowledge, only Ceschia, Schaerf and Stützle (2013) discussed reachability constraints. The other three constraints are addressed for the first time in the integrated vehicle routing and loading problem. We call our problem 3L -SDHVRP-f.

This paper aims to find high-quality solutions and simple loading patterns for 3L-SDHVRP-f in a short time. For this purpose, a column generation-based heuristic algorithm is proposed. First, the problem is formulated as a set-partitioning model. Then, in the form of a column generation algorithm, this model's linear relaxation is considered as the master problem of the column generation. At first, the number of columns in the master problem is restricted, and new columns are generated by solving subproblems. These columns, which are feasible in terms of loading, are added to the restricted master problem. Here a three-stage procedure is presented. At each stage, a heuristic algorithm is used to solve the subproblem. A hybrid heuristic algorithm is designed that employs wall building, stack building, and extreme points approaches to load the boxes in each route. Loading patterns are simple, understandable, and executable by the operator. The proposed

* Corresponding author.
E-mail address: moslehi@iut.ac.ir (G. Moslehi).

https://doi.org/10.1016/j.ejor.2021.08.025
0377-2217/

algorithm is tested with four sets of instances: the Shanghai instances (Bortfeldt & Yi, 2020), the Ceschia instances (Ceschia et al., 2013), the B-Y instances (Bortfeldt & Yi, 2020), and the 3L-CVRP instances (Gendreau et al., 2006). Then large-scale real-world instances are introduced, and the ability of the algorithm to solve them is demonstrated.

The innovations of this article can be summarized as follows:

- The vehicle type constraint is considered in the integrated routing and loading problem.
- Allocation-connectivity and allocation-separation constraints are considered in the integrated routing and loading problem.
- Understandable and straightforward loading patterns are produced, considering the complexity constraint.
- In the proposed column generation-based heuristic algorithm, a three-stage procedure is used to generate the columns.
- Boxes are loaded using a heuristic algorithm that combines the constructive methods of wall building, stack building, and extreme points.
- Four sets of the literature instances are tested. The proposed algorithm can obtain quality solutions in significantly shorter running times.
- Real-world instances, several times larger than the existing benchmarks, are studied.
- The proposed algorithm can solve the introduced real-world instances in less than an hour. Compared to the current situation, it can decrease the average total cost of transportation by 20% and the average number of vehicles by 36%.

The remainder of the paper is organized in this way. In Section 2, the related literature is reviewed. In Section 3, a detailed definition and formulation of the problem is provided. Section 4 introduces a solution algorithm for solving the 3L-SDHVRP-f. Computational results are presented and analyzed in Section 5. Finally, Section 6 provides conclusions and suggestions for future research.

2. Literature review

Bortfeldt and Wäscher (2013) identified and categorized constraints of container loading problems. Pollaris, Braekers, Caris, Janssens and Limbourg (2015) reviewed the literature on vehicle routing problems with loading constraints. (See also Iori & Martello, 2010).

Gendreau et al. (2006) were the first to introduce 3L-CVRP. They considered a fleet of identical vehicles. Besides routing aspects and a weight constraint, they considered geometric feasibility, orientation, fragility, vertical stability, and LIFO constraints for packing. Then some researchers investigated the same problem and proposed metaheuristic algorithms for routing and heuristic procedures for packing to reduce running times and improve the best solutions. Different algorithms have been used to find the routes, e.g., tabu search (Bortfeldt, 2012; Gendreau et al., 2006; Tao & Wang, 2015; Zhu, Qin, Lim & Wang, 2012), a combination of tabu search and guided local search (Tarantilis, Zachariadis & Kiranoudis, 2009), a genetic algorithm (Bortfeldt & Homberger, 2013; Moura, 2008), an ant colony (Fuellerer, Doerner, Hartl & Iori, 2010), a combination of the greedy-randomized adaptive search procedure and an evolutionary local search (Lacomme, Toussaint & Duhamel, 2013), a honeybee mating optimization (Ruan, Zhang, Miao & Shen, 2013), and an evolutionary local search (Zhang, Wei & Lim, 2015).

Junqueira, Oliveira, Carravilla and Morabito (2013) presented an integer linear programming model that solved small instances with up to 15 nodes and 32 boxes. Hokama, Miyazawa and Xavier (2016) developed a branch and bound algorithm. They did not consider fragility and stability constraints. Only Mahvash, Awasthi and Chauhan (2017) used a column generation-based heuristic for the integrated problem of routing and three-dimensional loading. They employed a heuristic method to find a new column and an extreme point-based constructive method for loading. Moura (2019) presented a mathematical model that could solve only small instances. A mathematical model-based heuristic was proposed to solve large instances up to 200 nodes, which could find high-quality solutions but required long running times. It should be noted that this author did not consider the LIFO constraint.

Some papers have dealt with other routing constraints. Hu, Zhao, Tao and Sheng (2015); Mak-Hau, Moser and Aleti (2018); Moura (2008); Moura (2019); Moura and Oliveira (2009); Song, Jones, Asgari and Pigden (2019) and Vega-Mejía, Montoya-Torres and Islam (2019)) considered the time window constraint. Ceschia et al. (2013); Wei, Zhang and Lim (2014); and Pace, Turky, Moser and Aleti (2015) assumed the vehicles to be heterogeneous. Männel and Bortfeldt (2016) and Männel and Bortfeldt (2018) considered the vehicle routing problem with pickups and deliveries. In pickup and delivery problems, items have to be transported from pickup nodes to the corresponding delivery nodes. Along the route, in addition to the delivery nodes, there are also pickup nodes. They introduced new constraints to prevent reloading effort. Reil, Bortfeldt and Mönch (2018) and Koch, Bortfeldt and Wäscher (2018) investigated the vehicle routing problem with backhauls in which items have to be received from customers and returned to the depot. Most of the mentioned articles have considered orientation, LIFO, fragility or stacking and stability constraints for loading. However, some articles have also considered other features such as reachability constraints (Ceschia et al., 2013), pallet loading (Song et al., 2019; Zachariadis, Tarantilis & Kiranoudis, 2012), and axle weight constraints and pallet loading (Pollaris, Braekers, Caris, Janssens & Limbourg, 2017). In pallet loading, boxes are first packed in pallets, and then the pallets are loaded in vehicles.

The vehicle routing problem with split deliveries and loading constraints has been investigated in four studies. Ceschia et al. (2013) allowed the customer to visit more than once, which means that they considered optional splitting. They developed a one-stage local search approach based on simulated annealing and large neighborhood search. They solved 13 real-world instances but were not able to show the benefit of splitting customer demand. Li, Yuan, Chen, Yao and Zeng (2018) also dealt with optional splitting. They proposed a data-driven three-layer algorithm but only tested it in very small instances. Yi and Bortfeldt (2018) allowed a customer's demand to be split if it cannot be packed in a single vehicle. They employed a hybrid algorithm. Bortfeldt and Yi (2020) developed this algorithm for both optional splitting and forced splitting. They showed that optional splitting could reduce transportation costs. In all the referenced studies, the maximum number of customers in the solved instances is 200.

Table 1 provides an overview of routing and packing features in papers on 3L-CVRP.

3. Problem description

The 3L-SDHVRP-f can be represented by a graph $G = (V, E)$. In this graph, $V = \{0, 1, \ldots, N\}$ is the set of nodes, including the depot, with index 0 and N customers. Also, $E = \{(k, l) : k, l \in V, k\backslash neql\}$is the set of edges. The traveling cost from k to l or the cost assigned to each edge (k, l) is given by d_{kl}. There is a fleet of vehicles of T different types in the depot, and the number of vehicles of any one type is unlimited. Each type of vehicle t $(t = 1, \ldots, T)$ has a maximum weight capacity Q_t, a fixed cost F_t, and a three-dimensional rectangular loading space with length L_t, width W_t, and height H_t. Each vehicle is loaded from the rear door. Different types of vehicles are sorted by increasing order of weight capacity

Table 1
Papers on 3L-CVRP.

Reference	HV	SD	TW	GF	WL	WD	Or	St	Sb	LIFO	Rc	AC	AS	Cm	PL
Gendreau et al. (2006)				×	×		×	×	×	×					
Tarantilis et al. (2009)				×	×		×	×	×	×					
Moura (2008)			×	×			×		×	×					
Moura and Oliveira (2009)			×	×			×		×	×					
Fuellerer et al. (2010)				×	×		×	×	×	×					
Bortfeldt (2012)				×	×		×	×	×	×					
Zachariadis et al. (2012)				×	×		×	×	×	×					×
Zhu et al. (2012)				×	×		×	×	×	×					
Bortfeldt and Homberger (2013)			×	×	×		×	×	×	×					
Ceschia et al. (2013))	×	×		×	×		×	×	×	×	×				
Junqueira et al. (2013)				×	×		×	×	×						
Lacomme et al. (2013)				×	×		×								
Ruan et al. (2013)				×	×		×	×	×	×					
Wei et al. (2014))	×			×	×		×	×	×	×					
Hu et al. (2015)			×	×											
Pace et al. (2015)	×		×	×	×	×		×							
Tao and Wang (2015)				×	×		×	×	×	×					
Zhang et al. (2015))	×			×	×		×	×	×	×					
Hokama et al. (2016)				×	×		×			×					
Männel and Bortfeldt (2016)				×	×		×	×	×	×					
Mahvash et al. (2017)				×	×		×	×	×	×					
Pollaris et al. (2017)				×	×	×		×		×					×
Koch et al. (2018)			×	×	×		×	×	×	×					
Li et al. (2018)		×		×	×		×	×	×	×					
Mak-Hau et al. (2018))	×		×	×	×	×				×					
Männel and Bortfeldt (2018)				×	×		×	×	×	×					
Reil et al. (2018)			×	×	×		×	×	×	×					
Yi and Bortfeldt (2018)		×		×	×		×	×	×	×					
Moura (2019)			×	×	×		×								
Song et al. (2019)			×	×	×					×					×
Vega-Mejía et al. (2019))			×	×	×	×	×	×	×	×					
Bortfeldt and Yi (2020)		×		×	×		×	×	×	×					

HV = heterogeneous vehicles, SD = split delivery, TW = time window, GF = geometric feasibility, WL = weight limit, WD = weight distribution, Or = orientation, St = stacking, Sb = Stability, Rc = reachability, AC = allocation-connectivity, AS = allocation-separation, Cm = complexity, PL = pallet loading.

($Q_1 < Q_2 < ... < Q_T$). Because a larger vehicle usually costs more, it is assumed $F_1 < F_2 < ... < F_T$.

Each customer k needs a set K_k that includes M_k rectangular boxes. Each box $K_{ki}(i = 1, ..., M_k)$ has weight q_{ki}, length l_{ki}, width w_{ki}, and height h_{ki}.

Vehicle loading space is displayed using a Cartesian coordinate system similar to Fig. 1. In this system, the length, width, and height of the loading space are parallel to the x, y, and z axes, respectively. The box placement K_{ki} in a loading space is characterized by (x_{ki}, y_{ki}, z_{ki}), which is the coordinates of its closest corner to the coordinate system's origin (Point A in Fig. 1).

The goal of solving 3L-SDHVRP-f is to find a set of vehicle routes with the minimum total transportation cost, considering the constraints listed below. The total transportation cost includes the fixed cost of using the vehicles and the traveling cost of the routes.

- Each vehicle route starts from the depot and ends there.
- Each customer k is visited by the minimum number of vehicles required to pack setK_k.
- The total weight and volume of boxes packed in each vehicle must not exceed the maximum weight capacity and volume.
- A customer may prohibit transporting their boxes with one or more types of vehicles.
- The loading pattern of all boxes within a vehicle must satisfy the following constraints:
  - ■ **Geometric feasibility constraints:** Each box lie within the loading space entirely and orthogonally, and no two boxes within a vehicle overlap.
  - ■ **Orientation constraint:** Each box has a fixed vertical orientation, but horizontal 90°rotation is allowed.
  - ■ **LIFO constraint:** Any box of a customer visited later than customer k on the route cannot be placed on top of K_{ki} or between K_{ki} and rear of the vehicle. In other words, all boxes of customer k must be unloaded by only using movements parallel to the length of the vehicle.
  - ■ **Stacking constraint:** A flag $s_{kilj}(k, l \in V\backslash\{0\}, i = 1, ..., M_k, j = 1, ..., M_l)$ is assigned to each pair of boxes. If $s_{kilj} = 1$, box

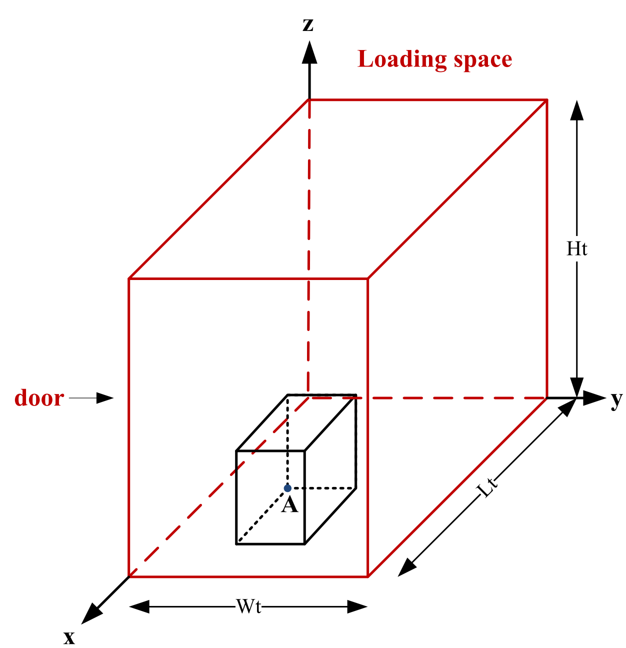

Fig. 1. A loading space with a placed box.

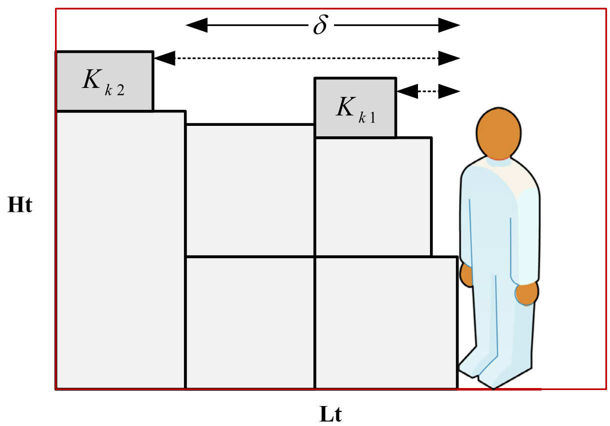

Fig. 2. The reachability constraint.

K_{ki} can be placed on top of box K_{lj}, otherwise $s_{kilj} = 0$. The parameter $s_{kilj}$is determined by factors such as the weight and fragility of the products inside the boxes.

- **Vertical stability constraint:** A box must either be on the vehicle floor or placed precisely on another box such that its base is 100% supported by the bottom box.
- **Allocation-connectivity constraint:** Some products have more than one component, and all of them need to be placed in one vehicle. This constraint applies to the customer, where it is not possible to pack all their boxes in one vehicle.
- **Allocation-separation constraints:** A flag $u_{kl}(k, l \in V\backslash\{0\})$ is assigned to each pair of customers. If $u_{kl} = 1$, the boxes of two customers k and l must not be packed in the same vehicle. If two customers are in the same area and compete, their corresponding flag is equal to one.
- **Reachability:** In loading and unloading operations, the distance between the operator and a box must be smaller than or equal to the specified value δ. It is assumed that the operator is as close as possible to the boxes inside the vehicle. In Fig. 2, only two boxes K_{k1} and K_{k2} belong to customer k. The position of both boxes is determined by satisfying the LIFO constraint. Box K_{k1} is reachable, but box K_{k2} is not reachable.
- **Complexity constraint:** Loading patterns must not be complex, i.e., they must be easily understood by operators and not difficult to implement. Here, the patterns should consist of several walls with several stacks in each wall and several layers in each stack.

4. Solution methodology

In this section, a column generation-based heuristic (CGH) procedure for solving 3L-SDHVRP-f is presented. Fig. 3 shows the flowchart for this procedure.

In this procedure, if a customer's demand cannot be packed in the largest permissible vehicle, forced splitting will occur. Forced splitting is described in Section 4.1. The next step in the solution procedure is the column generation approach (CG) discussed in Section 4.2. The column generation finds an LP solution that is a lower bound for the problem. If this solution is an integer, then the optimal or near-optimal solution for the proposed problem is obtained. Otherwise, deriving an integer solution step is needed to transform the fractional solution into a feasible integer solution. Deriving an integer solution is described in Section 4.5. This step sets the value of one or more variables to 1. After fixing some variables, the column generation approach is repeated, and the solution is used in deriving an integer solution step. This process is continued until an integer solution, which is an upper bound for the given problem, is found.

4.1. Forced splitting

Forced splitting occurs if the demand of a customer cannot be loaded in the largest permissible vehicle. A vehicle of type t will be permitted for a customer when they have not prohibited its use to transport their boxes. Forced splitting for vehicle routing problems with three-dimensional loading constraints depends on customer demand and the loading algorithm used. Let $\alpha_k$be the ratio of the demand volume of customer k to the volume of loading space of the largest permissible vehicle for customer k. Also, let β_k be the ratio of the demand weight of customer k to the weight capacity of the largest permissible vehicle for customer k. If α_k or $\beta_k$is greater than 100%, forced splitting is needed. However, α_k and β_k may be less than 100%, but the loading algorithm used may not allow for packing the customer's boxes in the vehicle by satisfying all the loading constraints. In this case, forced splitting will also occur. In forced splitting, the minimum split of customer demand or, in other words, the minimum dispersion of the customer's boxes, is permitted. We load the customer's boxes in the largest permissible vehicle using a heuristic loading algorithm, subject to the loading constraints, especially the allocation-connectivity constraint. This process continues until the customer's remaining boxes can be packed into a single vehicle. Then the customer's demand is updated. After forced splitting, the problem is solved as a heterogeneous vehicle routing problem with three-dimensional loading constraints (3L-HVRP). In other words, we consider another instance of the problem where there are only the remaining boxes for each customer. Boxes that are packed in forced splitting are not considered during routing. The vehicles used in forced splitting visit only the customer in question.

4.2. Column generation

As mentioned, in the 3L-SDHVRP solution procedure, after forced splitting, each customer's remaining boxes can be loaded into a single vehicle, and then the problem is solved like a 3L-HVRP. The column generation technique is useful for solving linear programming models with a large number of variables. Here, first, a set-partitioning model for the 3L-HVRP problem is presented. In the set-partitioning model, the variables are feasible route-vehicle pairs. A route-vehicle indicated by the pair (r, t) is feasible if the vehicle of type t starts and ends route r at the depot. At the same time, a subset of the customers is visited subject to the capacity and vehicle type constraints and all loading constraints. The symbols, parameters, and variables used in the set-partitioning formulation are as follows:

Symbols		
R	set of feasible route-vehicle pairs	
R_t	set of route-vehicle pairs for vehicle type t	$t \in \{1, ..., T\}$
Parameters		
a_k^{rt}	1, if route-vehicle (r, t) visits customer k; otherwise, 0	$k \in V\backslash\{0\}$ $t \in \{1, ..., T\}$ $r \in R_t$
c^{rt}	cost of route-vehicle (r, t) includes the fixed cost of using vehicle type t and the traveling cost of route r	$t \in \{1, ..., T\}$ $r \in R_t$
Variables		
x^{rt}	1, if route-vehicle (r, t) is selected; otherwise, 0	$t \in \{1, ..., T\}$ $r \in R_t$

The set-partitioning formulation of 3L-SDHVRP-f that we call problem P is as follows:

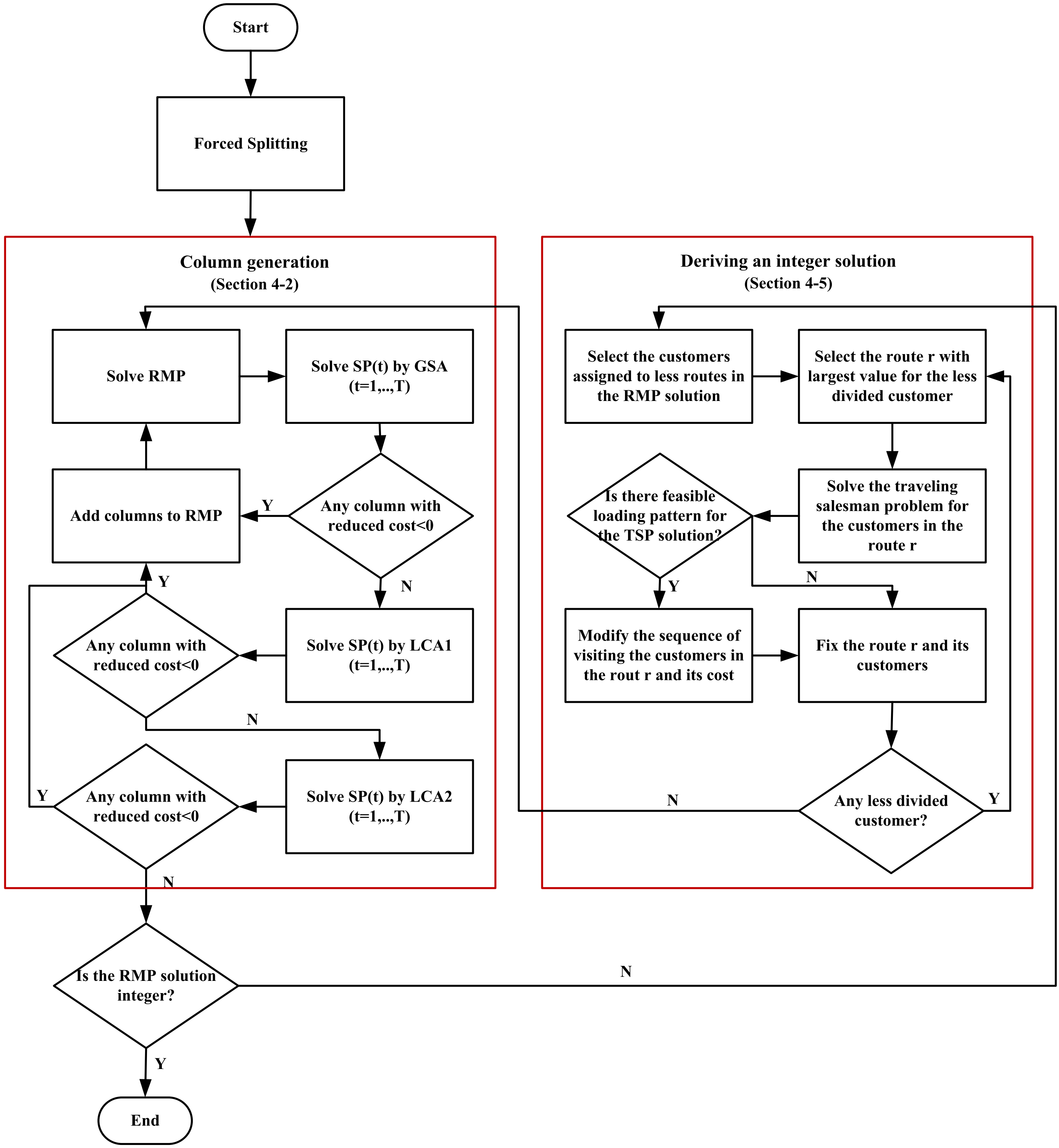

Fig. 3. Flowchart of the proposed column generation-based heuristic procedure (CGH).

(P):

$$min \sum_{t=1}^{T} \sum_{r \in R_t} c^{rt} x^{rt} \tag{1}$$

s.t.

$$\sum_{t=1}^{T} \sum_{r \in R_t} a_k^{rt} x^{rt} = 1 \tag{2}$$

$$x^{rt} \in \{0, 1\} \tag{3}$$

The objective function (1) minimizes the total cost of the selected route-vehicle pairs. Constraint (2) ensures that each customer is served, and by only one route-vehicle.

Let the master problem (MP) be the LP relaxation of problem P (defined by Eqs. (1 to 3), in which constraint (3) is relaxed. The optimal solution of MP is a lower bound for P. In MP, any decision variable or column corresponds to a feasible route-vehicle. The number of decision variables is very large, even in small instances, and increases exponentially with increases in customers and vehicle types. In such cases, the use of the column generation technique is recommended. The column generation deals with the restricted master problem (RMP), which is the MP with a subset of feasible columns$R'(\subset R)$. The CG process begins with several columns that make RMP feasible, and after solving the RMP, it uses the dual values to find a new column or variable to add to the RMP. This process is repeated until any suitable new variable is found to add to the RMP. A column will be added to the RMP if its reduced cost is negative, and if such a column is not found, the CG will stop.

Let $\pi_k, k \in V\backslash\{0\}$be the dual variables related to constraint (2), then the reduced cost of the vehicle-route (r,t) that is shown by $\bar{c}^{rt}$is calculated as follows:

$$\bar{c}^{rt} = c^{rt} - \sum_{k \in V\backslash\{0\}} \pi_k a_k^{rt} \tag{4}$$

$$\bar{c}^{rt} = F_t + \sum_{(k,l) \in \bar{E}^{rt}} d_{kl} - \sum_{k \in V\backslash\{0\}} \pi_k a_k^{rt} \tag{5}$$

Here the set $\bar{E}^{rt}$is the edges of the route-vehicle (r,t). The reduced cost of the route-vehicle (r,t) can be reformulated as follows:

$$\bar{c}^{rt} = F_t + \sum_{(k,l) \in \bar{E}^{rt}} \bar{c}_{kl}, \bar{c}_{kl} = \begin{Bmatrix} d_{kl} - \pi_l \forall (k,l) \in \bar{E}^{rt}, l \neq 0 \\ d_{kl} \forall (k,l) \in \bar{E}^{rt}, l = 0 \end{Bmatrix} \tag{6}$$

A new variable can be found by solving subproblem (SP), which is as follows:

(SP):

$$\min F_t + \sum_{(k,l) \in \bar{E}^{rt}} \bar{c}_{kl} \tag{7}$$

$$s.t : (r,t) \in R \tag{8}$$

As mentioned, the column generation process begins with some feasible route-vehicle pairs. For this purpose, each customer is assigned to the smallest permissible vehicle in which the customer's boxes can be packed, subject to the loading constraints. New variables are found by solving the subproblem based on the dual values obtained from solving the RMP. A column will be added to the RMP if its reduced cost is negative. The subproblem solving procedure is discussed in Section 4.3. This procedure has three stages. In the first stage, the subproblem is solved using a greedy heuristic algorithm. The second stage is to try to find new columns using a heuristic label-correcting algorithm. Finally, in the third stage, the subproblem is solved using another heuristic label-correcting algorithm. If new columns are found at each stage, they are added to the RMP, and the RMP is solved again. If no new column is found in any of the stages, the CG will stop. The pseudocode of the column generation procedure is presented in the form of Algorithm 1 in Fig. 4.

4.3. Subproblem

The subproblem is looking for a feasible route-vehicle with minimum reduced cost. The subproblem for vehicle type t is represented by SP(t) and is defined as follows:

(SP(t)) :

$$min \sum_{(k,l) \in \bar{E}^{rt}} \bar{c}_{kl} \tag{9}$$

$$s.t : (r,t) \in R_t \tag{10}$$

SP(t) can be represented by graph $G^t = (V^t, E^t)$, in which V^t includes the depot and all customers who have not prohibited the transport of their boxes by vehicle type t, and E^t is the set of edges. SP(t) is an elementary shortest path problem with resource constraints (ESPPRC), in which the loading constraints expressed in the problem description must also be satisfied. It should be noted that ESPPRC with loading constraints is strongly NP-hard (Mahvash et al., 2017). Here a three-stage procedure is proposed. At each stage, a heuristic algorithm is employed to solve the subproblem. The reduced cost of a column obtained using heuristic methods is not necessarily the minimum. Each proposed heuristic algorithm can generate multiple columns at the same time and add them to the RMP.

4.3.1. Greedy search algorithm

A greedy search algorithm (GSA) is used to solve SP(t). The pseudocode of the GSA is given in the form of Algorithm 2 in Fig. 5. Here, for each node $k \in V^t$,B_k is the set of the nodes $l \in V^t\backslash\{0\}, (l,k) \in E^t$. In the GSA, boxes are packed by a heuristic loading algorithm, which is described in section 4.4. It should be noted that several columns may be generated simultaneously in the GSA for each type of vehicle.

4.3.2. Label-correcting algorithm 1

Label-correction algorithm 1 (LCA1) is designed based on the algorithm of Desrochers (1988). The Desrochers algorithm is label-correcting, and it is a developed version of the Ford-Bellman algorithm, which is presented to solve the shortest path problem with resource constraints (SPPRC). In this algorithm, each node gets some labels. Each label is associated with a partial path and indicates the cost and resource consumption of this path. A partial path starts at the depot and goes through one or more nodes. The algorithm examines the nodes individually and develops all their labels in all possible directions to generate new paths and labels. The efficiency of this approach depends on identifying and eliminating inappropriate partial paths.

Let r_{0l} indicate the path from the depot to node l. Here, for each SP(t) (defined by Eqs. (9) and 10), there are two resources, the weight and the volume of vehicle type t. So, we show $r_{0l}$with a four-part label (k, P_l^1, P_l^2, C_l) that is assigned to node l. This label indicates that node k is visited just before node l. Also, r_{0l}uses P_l^1 units of the volume resource and P_l^2 units of the weight resource. The reduced cost of the path is C_l.

In LCA1, $\sum_{l \in V^t\backslash\{0\}} \bar{c}_{kl}$is calculated for each customer k. Let B be the set of $k \in V^t\backslash\{0\}$sorted by increasing order of $\sum_{l \in V^t\backslash\{0\}} \bar{c}_{kl}$. The nodes are investigated according to the order of the set. When checking node k, new labels are generated only for nodes that are in the ordered set after node k. Thus, elementary paths will be created in which each node is visited at most once. In generating new labels, the weight capacity and volume of the vehicle are checked.

In label-correcting algorithms, definition of the dominance rule is essential for identifying inappropriate partial paths.

Definition 1. Let r'_{0l} and r_{0l} be two distinct paths from the depot to node l with associated labels $(k\prime, P\prime_l^1, P\prime_l^2, C\prime_l)$ and (k, P_l^1, P_l^2, C_l). So, r'_{0l} dominates r_{0l} if, and only if, $C\prime_l \le C_l, P\prime_l^1 \le P_l^1, P\prime_l^2 \le P_l^2and(k\prime, P\prime_l^1, P\prime_l^2, C\prime_l) \neq (k, P_l^1, P_l^2, C_l)$.

Claim. In LCA1, we only need to consider nondominated paths.

In LCA1, only one label is kept for each node. If two labels of a node do not dominate each other, the label with less cost will be selected; if the labels have equal cost, the label with less volume consumption will be selected. The pseudocode of LCA1 is presented in the form of Algorithm 3 in Fig. 6. In LCA1, boxes

Algorithm 1:

1 Generate initial route-vehicle pairs (columns)
2 **while** any column with negative reduced cost exists, Solve the RMP and obtain dual values
3 Solve the SP by Greedy Search Algorithm (GSA)
4 **if** No column found
5 Solve the SP by Label Correcting Algorithm 1 (LCA1)
6 **endif**
7 **if** No column found
8 Solve the SP by Label Correcting Algorithm 2 (LCA2)
9 **endif**
10 Add columns with negative reduced cost to the RMP
11 **endwhile**

Fig. 4. Column generation procedure.

Algorithm 2:

1 **for** each edge $(l,k)\in E^t$ **do**
2 Calculate $\bar{c}_{lk}$
3 **endfor**
4 **for** each node $k\in V^t$ **do**
5 Prepare the set B_k
6 **endfor**
7 $b=0$
8 $\bar{c}^{rt}=F_t$
9 Sort the set B_b in ascending order of $\bar{c}_{lb}$
10 **for** $l=1$ to $|B_b|$ **do**
11 **if** the boxes of lth customer have not been previously placed in the vehicle
12 **if** the boxes of lth customer can be packed inside the vehicle by heuristic loading algorithm
13 Load the boxes of lth customer
14 $\bar{c}^{rt}=\bar{c}^{rt}+\bar{c}_{lb}$
15 Name b the lth customer
16 Go to step 21
17 **endif**
18 **endif**
19 **endfor**
20 Go to Step 25
21 **if** $\bar{c}^{rt}+\bar{c}_{0b}<0$
22 Add the vehicle route as a new column to RMP
23 Go to step 9
24 **endif**

Fig. 5. Overview of the GSA.

Algorithm 3:

1 **for** $l=1$ to $|B|$ **do**
2 Develop the labels
3 **endfor**
4 **for** $l=1$ to $|B|$ **do**
5 **if** $C_l+\bar{c}_{l0}<0$
6 Identify the visited customers along the route with a backward movement
7 **endif**
8 **if** all boxes of visited customers can be packed inside the vehicle by heuristic loading algorithm
9 Add the vehicle route as a new column to RMP
10 **endif**
11 **endfor**

Fig. 6. Overview of LCA1.

are packed by a heuristic loading algorithm, which is described in Section 4.4. It should be noted that LCA1 may produce several columns for each type of vehicle.

4.3.3. Label-correcting algorithm 2

In LCA1, one label is kept for each node, but in label-correcting algorithm 2 (LCA2), $E > 1$ labels at most are stored for each node. Obviously, the larger the E is, the longer the running time. The labels of each node are sorted by increasing cost order; if equal in cost, the ties are broken by increasing volume consumption order; and if equal in volume, by increasing weight consumption order. Let r_{0l} be the path from the depot to node l. In LCA2, to quickly find the visited customers in backward movement, the path is indicated by a five-part label $(k, e, P_l^1, P_l^2, C_l)$ assigned to node l. This label means that path r_{0l} was obtained by extending the eth label of node k. Also, r_{0l}uses P_l^1 units of the volume resource and P_l^2 units of the weight resource. The cost of the path is C_l. When a new label is generated for a node, the dominance rule is checked. If the new label dominates one or more of the existing labels or vice versa, the dominated labels are removed. The new label is stored if any node labels do not dominate it and if the number of node labels is less than E. In LCA2, boxes are packed by a heuristic loading algorithm, which is described in Section 4.4. It should be noted that LCA2 may produce several columns for each type of vehicle.

4.3.4. Heuristic loading algorithm

We propose a heuristic algorithm for loading boxes into a vehicle called the wall-stack building algorithm (WSBA). In this algorithm, the wall building, stack building, and extreme points approaches are used for packing the boxes. Boxes are loaded such that vertical layers or walls are placed one after the other along the loading space length. The WSBA produces loading patterns that are easily understood and implemented by the operator. These patterns consist of several walls with several stacks in each wall and several layers in each stack. All boxes of a customer are placed in sequential walls, taking into account the reachability constraint, so they can be quickly loaded and unloaded.

In each wall, the boxes are stacked using the extreme points approach. Extreme points are determined using the Crainic, Perboli and Tadei (2008) approach. In Fig. 7, let the dimensions of box K_{ki} be h_{ki}, w_{ki}, l_{ki} along the *x, y, z* axes, respectively, with the left-back-bottom corner at point(x_0, y_0, z_0). The new extreme points are obtained by projecting points$(x_0 + l_{ki}, y_0, z_0)$, $(x_0, y_0 + w_{ki}, z_0)$ and $(x_0, y_0, z_0 + h_{ki})$ on the loading space's orthogonal axes. These new extreme points are shown in Fig. 7 with black circles.

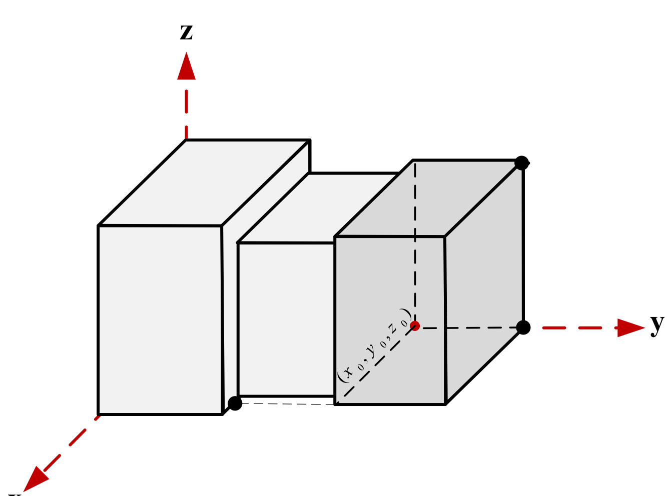

Fig. 7. Extreme points defined by an item (black circles).

Loading the customers' boxes in the vehicle is controlled by three box sorting rules and two criteria for selecting extreme points. In each sorting rule, customers are sorted by the inverse of visiting order to satisfy the LIFO constraint. The following rules are used to sort the boxes of each customer:

- Rule 1: Decreasing volume, breaking ties by decreasing height (Mahvash et al., 2017)
- Rule 2: Decreasing base area, breaking ties by decreasing height (Mahvash et al., 2017)
- Rule 3: Decreasing height, breaking ties by decreasing base area (Mahvash et al., 2017)

The following criteria select the extreme points:

- Criteria 1: The extreme point with the minimum *x* value, breaking ties with the minimum *z* value and then minimum y value (Mahvash et al., 2017)
- Criteria 2: The extreme point with the minimum *x* value, breaking ties with the minimum *y* value and then minimum z value (Mahvash et al., 2017)

The GSA is implemented for different combinations of box sorting rules and extreme points selection criteria. If a new column is found using one combination, the others are not checked. In LCA1 and LCA2, the loading of generated routes with negative reduced cost is also examined using different combinations. If the loading pattern is feasible in one combination, the others are not checked.

In the WSBA, the walls are built one after the other along the loading space's length. The length, width, and height of each wall are determined by the difference between the largest and smallest coordinates *x*, *y*, and *z* of the boxes in the wall. Let wall *w* be made with length l_w^{wall}, width w_w^{wall}, and height h_w^{wall} inside vehicle type *t*. Fig. 8 shows the empty spaces after the creation of wall *w*. As can be seen, there is an empty space in front of the wall called the length space. There is also one empty space on the side of the wall, and another on top of the wall, called the width and height space, respectively. It is not possible to use this width and height space when constructing wall *w*, but it is possible to create a new width space (height space) by merging the two width spaces (height spaces) after building the next wall, in which one or more boxes can be placed.

The pseudocode of the WSBA is given in the form of Algorithm 4 in Fig. 9. In WSBA, if the allocation-separation constraint is considered for each pair of customers within the route, the loading of boxes will be checked. Otherwise, of course, there is not a feasible loading pattern for the route.

The new wall is built in the length space. Due to the defined stability constraint, each box must be fully supported by only one bottom box. Suppose a box is placed on the vehicle floor. One or more boxes will be placed on it that are fully supported. This process can be continued. As a result, a stack is formed such that the base area of its bounding cube is base area of the box on the vehicle's floor. Therefore, each wall consists of several stacks with several layers in each stack.

The pseudocode for building a new wall is presented in the form of Algorithm 5 in Fig. 10. The first box packed in the length space determines the maximum depth of the wall. The boxes are loaded into each wall using the extreme points approach. Let *EP* be the ordered set of extreme points, and *BX* be the ordered set of boxes. In building a new wall, an attempt is made to place each box at the first extreme point of the set *EP*, subject to the geometric feasibility, stacking, and vertical stability constraints, and by examining both possible directions. After a box has been loaded at an extreme point, the set *EP* will be updated. For this purpose, new extreme points are calculated and added to the set while

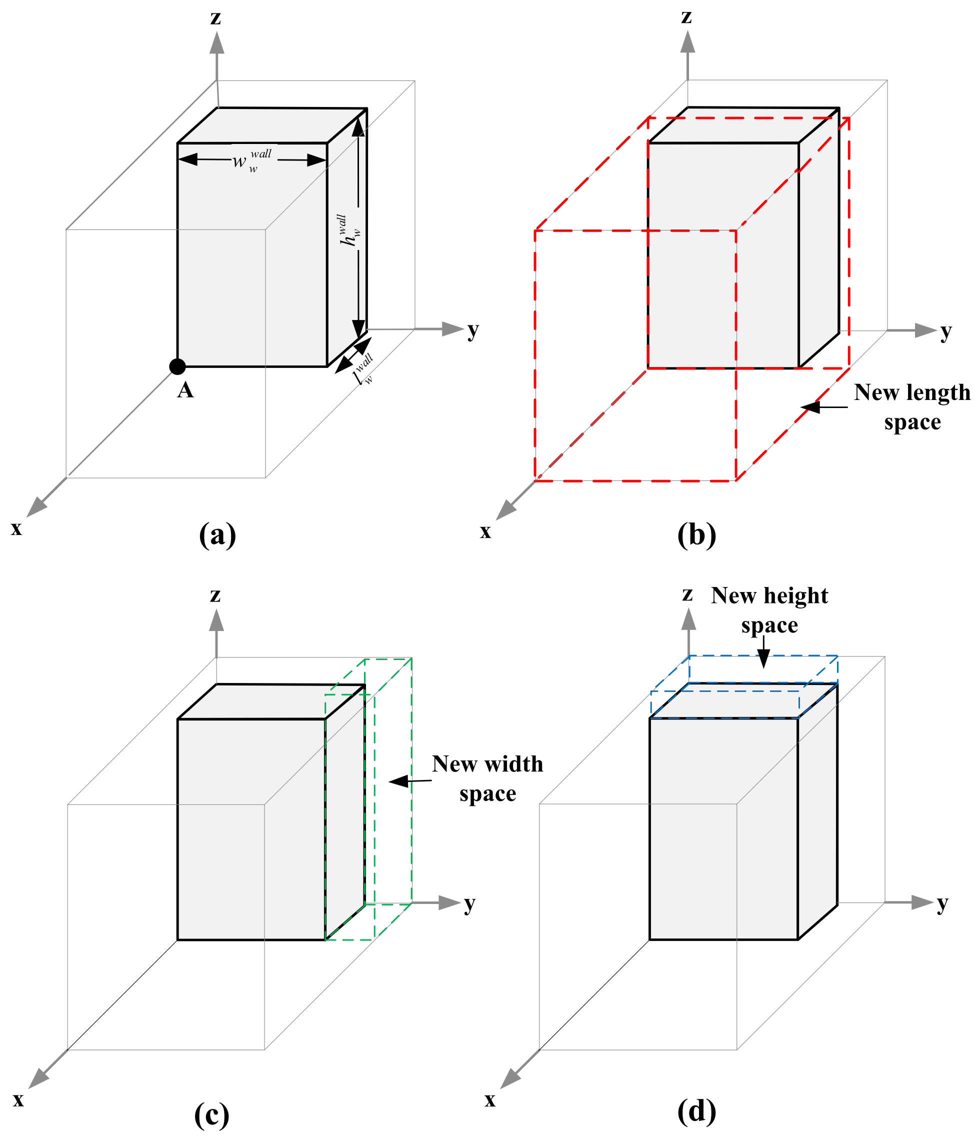

Fig. 8. (a) Wall (b, c, d) Empty space generation.

maintaining the order. The extreme points that are not compatible with loading this item are also removed from the *EP*. If it is impossible to place a box in any extreme points, the next box of the same customer will be checked.

When the number of points in the set *EP* is more than the maximum number of extreme points in the wall, or another box cannot be placed in the wall, the building of a new wall stops. The maximum number of extreme points is a parameter that we specify.

If there is at least one box in the new wall, the WSBA will be continued by loading in new empty spaces. Otherwise, it will be stopped.

After building wall w, new length, width, and height spaces are produced, as shown in Fig. 8. If the width space of wall $w-1$ is empty, it merges with the width space of wall w. It is clear that if wall w is the first wall built, the spaces will not merge. In the same way, height spaces are integrated. Only two sequential walls are merged to comply with the reachability constraint. If the first wall of customer k is wall w, the boxes of customer k will not be placed in the walls before wall w-1 to facilitate unloading for the operator. Only integrated width and height spaces are loaded.

The filling of a width space is similar to building a wall, but the packing of a height space has a few differences. Only the boxes that are designated as small boxes here are loaded in a height space. Our real-world case study has small boxes. They are products in small, light packages that can be placed on top of other boxes. To merge two height spaces, the height of the higher wall is considered the basis. Also, not all stacks in a wall are the same height. Therefore, fillers can be used in empty spaces under small boxes to support them. Small boxes can be placed in the integrated height space of the two sequential walls $w-1$ and w, which belong to the last customer in wall w. Otherwise, the LIFO constraint will not be satisfied. The stacking and stability constraints do not matter for small boxes and are not controlled.

4.3.5. Deriving an integer solution

After stopping column generation, if the RMP solution is an integer, an optimal or near-optimal solution for problem P is obtained. Otherwise, in the step of deriving an integer solution, some variables are selected with fractional values, and they are set to 1.

The pseudocode of deriving an integer solution is given in the form of Algorithm 6 in Fig. 11. In the RMP solution, each customer may be assigned to multiple routes for which the corresponding decision variables (x^{rt}) have fractional values. It is possible to se-

Algorithm 4:

1 Set status to true
2 **if** allocation-separation constraint is not satisfied
3   Set status to false and go to step 16
4 **endif**
5 **while** there are no more boxes to pack
6   Build a new wall
7   **if** no box fits inside the wall
8   Set status to false and go to step 16
9   **endif**
10   Generate empty spaces
11   Merge width spaces
12   Merge height spaces
13   Fill merged width space
14   Fill merged height space
15 **endwhile**
16 **return** status

Fig. 9. WSBA pseudocode.

lect one, and only one, route for customers belonging to several routes. Let no_r_k be the number of routes in which customer k is visited. Here, customers whose number of assigned routes is less than others are identified. For each such customer, the route with the largest value of the corresponding decision variable is selected. After choosing a route, the best sequence of visiting its nodes is obtained by solving the corresponding traveling salesman problem's mathematical model. If the loading of the best sequence is feasible, its traveling cost is considered in the total transportation cost. Otherwise, the cost of the route obtained by solving the subproblem is applied to the total transportation cost. Then the selected route is set to 1, and its customers are added to *FC,* which is the set of customers who are visited on fixed routes.

5. Computational experiments

In this section, we explain the setting of the parameters used in our algorithm. Then four sets of instances from the literature are tested. The purpose of investigating these sets is to show the proposed algorithm's speed in achieving quality solutions. Finally, we test instances derived from a real-world case study whose size is several times larger than the existing instances in the literature.

All numerical experiments are performed on a 2.3 GHz Intel (R) Xeon (R) PC with eight cores and 12.0 GB RAM under Windows 7. The solution algorithm is coded using Visual C # 2017, and mathematical models are solved with CPLEX 12.8 solver.

Algorithm 5:

1 Set status to false
2 $EP = \emptyset$
3 Add the coordinates of the left-back-bottom corner of the length space to EP
4 $i = 1$
5 **while** ($BX \neq \emptyset$ and $i < |BX|$)
6   status = Load ith box in the first possible extreme point of the set EP
7   **if** status is true
8   Remove ith box from the set BX
9     $i = 1$
10   Update the EP set
11   Remove the incompatible extreme points from the set EP
12   **if** ($|EP|$ > maximum number of extreme points in the wall)
13   Go to step 21
14   **endif**
15   **elseif** i and i+1 belong to the same customer
16     $i = i+1$
17   **else**
18   Go to step 21
19   **endif**
20 **endwhile**
21 stop

Fig. 10. Pseudocode for building a new wall.

5.1. Parameter setting

Two parameters for packing need to be determined: the maximum number of extreme points in the wall; and the maximum number of extreme points in the width space. We considered three levels (100, 150, and 200) for the first parameter and three lev-

Table 2
Results for parameter setting tests.

Maximum number of extreme points			
in the wall	in the width space	*TTC*	*Rt*(s)
100	50	4140.1	60.9
100	75	4140.1	59.4
100	100	4140.1	58.2
150	50	4135.9	62.1
150	75	4135.9	62.2
150	100	4135.9	63.6
200	50	4135.9	63.3
200	75	4135.9	63.4
200	100	4135.9	63.3

Algorithm 6:

1 **for each** $k \in V \setminus \{0\}$
2   Calculate no_r_k
3 **endfor**
4 $mr = \min\{no\_r_k \mid k \in V \setminus \{0\}\}$
5 **for each** $k \in V \setminus \{0\}$
6   **if** ($no\_r_k = mr$ & $k \notin FC$)
7   Name r the route with the largest value of the customer k
8   Solve the traveling salesman problem for the customers in route r
9   **if** heuristic loading algorithm can pack the customers in route r in the sequence of the TSP solution
10       $c^{rt} = F_t$ + the traveling cost of the TSP solution
11   **endif**
12   Set the corresponding variable of route r to one
13   Add all customers in route r to FC
14   **endif**
15 **endfor**

Fig. 11. Deriving an integer solution pseudocode.

Table 3
Parameter E for LCA2.

Condition	Value
if $N <= 250$	5
if $250 < N <= 500$	3
if $500 < N <= 1000$	2

els (50, 75, and 100) for the second parameter. Then experiments using the values obtained by combining the levels of the parameters were performed on the B-Y instances (Bortfeldt & Yi, 2020). These instances were chosen for testing because they are the closest benchmark instances to our real-world instances in terms of the variety of boxes. The response variables are total transportation cost (*TTC*) and running time, but *TTC* has higher priority. Table 2 shows the test results. The first two columns show the test. The third and fourth columns show total transportation cost (*TTC*) and running time in seconds (*Rt*). According to Table 2, the results are close to each other, and the algorithm is not very sensitive to the maximum number of extreme points. We set the maximum number of extreme points in the wall to 150 and the maximum number of extreme points in the width space to 50.

The number of labels (Parameter *E*) in LCA2 is set according to Table 3. Therefore, if the number of customers is 250, 500, and 1000, LCA2 will produce 1250, 1500, and 2000 routes at most, and will control their loading for each type of vehicle.

5.2. Instances from the literature

We test three sets of instances: the Shanghai instances (Bortfeldt & Yi, 2020): the Ceschia instances (Ceschia et al., 2013): and the B-Y instances (Bortfeldt & Yi, 2020). Bortfeldt and Yi (2020) consider these instances and their results to be the best in the literature. We compare our results with them. We also apply our algorithm to 27 3L-CVRP instances of Gendreau et al. (2006) in which the boxes are strongly heterogeneous. In all these instances, the vehicles are homogeneous, and the fixed cost is zero.

Bortfeldt and Yi (2020) presented a hybrid algorithm (SDVRLLH2) following the principle of "first packing, second routing," consisting of a genetic algorithm and several heuristic methods for loading and a local search for routing. They defined the stacking constraint such that only other fragile boxes could be placed on one fragile box, and both types of boxes could be placed on a nonfragile box. Also, a box is stable if it is placed on the vehicle floor, or a bottom box supports 75% of its base area. We considered the constraints of stacking and vertical stability in the instances from the literature similar to these authors.

These authors ran their algorithm 10 times under the conditions of forced splitting and optional splitting for each instance. They reported the average values of total transportation cost and running time (in seconds) and the best value of total transportation cost over 10 runs. Optional splitting is not desirable in our case because it may disperse a customer's boxes in more than one vehicle, which is not necessary. We compare our cost with the best ones found by the Bortfeldt and Yi (2020) algorithm in the forced-splitting variant. It should be noted that CGH only needs to be run once, while Bortfeldt and Yi (2020) ran their algorithm 10 times and reported the average running time for solving each instance. This point should be considered in comparing the running times.

As mentioned, the purpose of testing CGH on the instances from the literature is to achieve a quality solution quickly. According to the test results, it seems that CGH can solve real-world instances with several hundred customers in a reasonable time. Increasing the number of customers increases the running times exponentially. The solution methods in the literature require at least half an hour to solve instances with 100 to 200 customers and homogeneous vehicles. However, they cannot solve instances with several hundred customers and heterogeneous vehicles.

5.2.1. Shanghai instances tests

The Shanghai (Bortfeldt & Yi, 2020) instances were derived from an automotive logistics company in Shanghai. The set includes 15 instances, each containing 5 to 200 customers and 73 to 4476 boxes. The total data have 634 different packaging dimensions. The smallest box is $6 \times 9 \times 12$ centimeters, and the largest is $190 \times 155 \times 225$ centimeters. There are five different vehicles in this set, but the vehicles in each instance are homogeneous.

The results of the Shanghai instances are summarized in Table 4. The first four columns show the instance name, the number of customers (*N*), the total number of boxes (*M*), and the variety of boxes (*Bt*). The total transportation cost (*TTC*), the average filling rates of the loading space as percentages (*Fr*), and running times in seconds (*Rt*) are given for CGH as well as for SDVRLLH2. In the last line, the average results are presented.

As seen in Table 4, CGH improves the result in terms of *TTC* in 3 out of 15 instances, and while the time to run CGH once is

Table 4
Results for Shanghai instances.

				CGH			SDVRLH2		
instance	*N*	*M*	*Bt*	*TTC*	*Fr* (%)	*Rt* (*s*)	*TTC*	*Fr* (%)	*Rt** (*s*)
Sha1	5	261	26	731	41.3	2.6	**562.6**	55.1	0.1
Sha2	8	167	50	**2677.2**	58.8	0.5	2813.2	58.8	11.8
Sha3	10	73	17	499.3	42.9	0.1	**393.0**	53.6	0.4
Sha4	12	204	33	388.8	61.2	1.4	**349.7**	61.2	2.7
Sha5	12	228	59	**1433.1**	57.6	2.7	1447.6	57.6	7.5
Sha6	15	228	56	553.4	39.0	5.8	**481.3**	48.7	10.6
Sha7	16	439	79	1867.53	51.5	6.5	**1694.8**	55.8	16.0
Sha8	18	303	51	424.4	57.3	10.5	**360.9**	64.5	5.1
Sha9	27	734	98	978.2	51.4	33.3	**942.6**	55.4	12.2
Sha10	31	590	134	1824.7	54.0	27.1	**1590.1**	61.7	15.8
Sha11	46	1549	185	564.4	51.3	238.0	**512.6**	57.0	93.2
Sha12	35	623	149	2992.7	55.0	49.8	**2942.1**	55.0	35.4
Sha13	64	1494	234	3410.5	60.6	334.5	**3128.2**	62.7	424.8
Sha14	101	2659	311	1576.1	56.0	1162.8	**1481.8**	56.0	883.0
Sha15	200	4776	634	**4510.76**	55.5	5165.4	4867.8	58.6	3620.5
Average				1628.8	52.9	469.4	1571.2	57.4	342.6

**Rt* in SDVRLH2 is the average of 10 running times.

Table 5
Results for (modified) Ceschia instances.

instance	*N*	*M*	*Bt*	CGH *TTC*	CGH *Fr* (%)	CGH *Rt* (*s*)	SDVRLH2 *TTC*	SDVRLH2 *Fr* (%)	SDVRLH2 *Rt** (*s*)
SD-CSS1	11	254	36	5924.1	37.6	1.2	**5467.4**	45.2	0.2
SD-CSS2	25	350	15	13,362.0	45.6	0.8	**13,044.5**	45.6	3.0
SD-CSS3	33	285	9	17,506.3	41.6	0.7	**16,154**	45.0	21.1
SD-CSS4	37	312	13	13,751.6	42.4	0.9	**12,700.3**	48.5	13.3
SD-CSS5	41	7035	47	21,226.8	2.6	244.6	**14,513.8**	4.0	19.5
SD-CSS 6	43	8060	97	**14,812.9**	58.7	656.2	16,021.3	58.7	36.7
SD-CSS 7	45	284	14	14,013.6	40.5	1.4	**12,084.8**	49.8	47.3
SD-CSS 8	48	3275	70	**19,032.2**	55.0	149.6	19,474.4	55.0	44.3
SD-CSS 9	56	1725	45	**16,391.5**	55.3	81.2	16,585.7	58.5	133.7
SD-CSS 10	60	1840	29	**12,227.2**	46.6	309.4	16,079.4	25.9	205.1
SD-CSS 11	92	3790	34	**12,967.6**	49.1	1707.3	18,593.2	28.1	1825.5
SD-CSS 12	129	745	10	40,073.7	54.8	4.5	**36,793.9**	59.6	1135.0
SD-CSS 13	129	2880	63	23,464.5	44.4	776.8	**21,338.6**	50.7	1782.0
Average				17,288.8	44.2	302.7	16,834.7	44.2	405.1

**Rt* in SDVRLH2 is the average of 10 running times.

Table 6
Results for B-Y instances.

instance	*N*	*M*	*Bt*	CGH *TTC*	CGH *Fr* (%)	CGH *Rt* (*s*)	SDVRLH2 *TTC*	SDVRLH2 *Fr* (%)	SDVRLH2 *Rt** (*s*)
B-Y1	50	3535	3	2402.2	51.8	3.5	2402.2	51.8	40.4
B-Y2	50	1295	3	1141.1	52.7	14.6	**1019.4**	59.3	59.3
B-Y3	50	3377	20	2402.2	50.9	6.4	2402.2	50.9	41.0
B-Y4	50	1442	20	1317.7	52.4	20.5	**1163.9**	57.9	54.6
B-Y5	75	5218	3	3630.5	50.3	6.7	3630.5	50.3	270.8
B-Y6	75	2135	3	1759.7	54.0	25.4	**1573.5**	64.8	323.6
B-Y7	75	5116	20	3630.5	52.4	8.8	3630.5	52.4	271.3
B-Y8	75	2054	20	1826.0	51.2	48.5	**1643.6**	56.9	346.8
B-Y9	100	6861	3	4989.8	50.1	11.4	4989.8	50.1	1064.1
B-Y10	100	2854	3	2478.8	55.0	42.3	**2077.3**	63.1	1817.3
B-Y11	100	6866	20	4989.8	52.5	9.7	4989.8	52.5	1060.7
B-Y12	100	2680	20	2499.0	50.4	77.2	**2167.8**	59.1	1342.4
B-Y13	150	10,356	3	7360.6	51.2	15.7	**7341.9**	51.6	320.6
B-Y14	150	4508	3	3306.8	56.7	187.3	**2915.4**	63.7	1818.9
B-Y15	150	10,082	20	7365.7	52.1	24.7	**7360.5**	52.1	320.6
B-Y16	150	3904	20	3506.6	50.0	120.4	**3049.4**	58.5	1594.5
B-Y17	199	14,230	3	9610.9	51.5	40.9	**9558.3**	51.8	1098.2
B-Y18	199	5790	3	4093.8	56.8	274.5	**3665.6**	63.0	1969.6
B-Y19	199	13,058	20	9610.3	51.3	47.2	**9607.8**	51.3	1106.2
B-Y20	199	5797	20	4795.2	51.9	258.5	**4156.4**	59.6	1875.9
Average				4135.9	52.3	62.2	3967.3	56.0	839.8

**Rt* in SDVRLH2 is the average of ten running times.

about 7.3 times lower than the 10 total running times of SVRLH2, the average *TTC* increases by only 3.7%. CGH's average filling rate is 4.5% lower. It should be noted that we designed simple loading patterns that consider the complexity and reachability constraints.

5.2.2. Ceschia instances tests

Ceschia et al. (2013) presented their instances based on real-world operations. Their set includes 13 instances, each containing 11 to 129 customers and 254 to 8060 boxes. The smallest box is $10 \times 1 \times 11$ centimeters, and the largest box is $239 \times 100 \times 230$ centimeters. They considered vehicles to be homogeneous in some instances and heterogeneous in others. However, Bortfeldt and Yi (2020) changed the instances and considered an unlimited homogenous fleet in all of them. To compare our algorithm with them, we also deal with homogeneous vehicles. Table 5 demonstrates that CGH improves the *TTC* in 5 out of 13 instances. Moreover, the time to run CGH once is about 13.5 times lower than the 10 total running times of SDVRH2. The average *TTC* increases by only 2.7%, and the average filling rates are the same.

5.2.3. B-Y instances tests

Bortfeldt and Yi (2020) introduced the B-Y instances by combining five CVRP instances from Christofides (1979) and 200 CLP instances from Bischoff and Ratcliff (1995). The set includes 20 instances, each containing 50 to 199 customers and 3 to 20 boxes. They considered the fifth box of each customer to be fragile.

As seen in Table 6, CGH obtained the same SDVRH2 solution in 6 out of 20 instances, and while the time to run CGH is about 135 times lower than the 10 total running times of SDVRH2, the average *TTC* increases by only 4.2%. It should be noted that Bortfeldt and Yi (2020), in their algorithm, followed the principle of "first packing, second routing." Therefore, if their algorithm is executed for one instance and stopped after the time required by CGH to solve the same instance, their algorithm may be in the packing stage, and a feasible solution to the problem may not be obtained. The CGH average filling rate is 3.7% lower

5.2.4. Instances by gendreau et al. (2006)

The 3L-CVRP instances of Gendreau et al. (2006) have strongly heterogeneous boxes such that the number of box types equals the number of boxes.

Table 7
Results for 3L-CVRP instances.

instance	N	M	Bt	CGH TTC	Fr (%)	Rt (s)	TS TTC
3L-CVRP1	15	32	32	380.6172	35.7	0.7	**316.32**
3L-CVRP2	15	26	26	393.1436	23.7	0.5	**350.58**
3L-CVRP3	20	37	37	544.3666	28.0	0.6	**447.73**
3L-CVRP4	20	36	36	491.1278	27.6	0.5	**448.48**
3L-CVRP5	21	45	45	712.0419	29.3	0.6	**464.24**
3L-CVRP6	21	40	40	606.2166	27.9	0.6	**504.46**
3L-CVRP7	22	46	46	1144.438	31.8	0.6	**831.66**
3L-CVRP8	22	43	43	1149.736	28.3	0.6	**871.77**
3L-CVRP9	25	50	50	827.2293	27.8	0.6	**666.1**
3L-CVRP10	29	62	62	1455.063	28.7	0.6	**911.16**
3L-CVRP11	29	58	58	1317.899	29.8	0.6	**819.36**
3L-CVRP12	30	63	63	715.7799	30.5	0.6	**651.58**
3L-CVRP13	32	61	61	4316.482	31.3	0.6	**2928.34**
3L-CVRP14	32	72	72	2207.966	37.8	0.7	**1559.64**
3L-CVRP15	32	68	68	2091.8	39.5	0.7	**1452.34**
3L-CVRP16	35	63	63	850.1355	24.9	0.6	**707.85**
3L-CVRP17	40	79	79	1018.53	26.5	0.6	**920.87**
3L-CVRP18	44	94	94	1818.216	31.2	0.7	**1400.52**
3L-CVRP19	50	99	99	1186.629	32.1	0.8	**871.29**
3L-CVRP20	71	147	147	1050.466	32.3	0.8	**732.12**
3L-CVRP21	75	155	155	1772.752	35.2	1.2	**1275.2**
3L-CVRP22	75	146	146	1741.928	35.2	1.0	**1277.94**
3L-CVRP23	75	150	150	1760.111	32.5	1.1	**1258.16**
3L-CVRP24	75	143	143	1725.277	32.5	1.0	**1307.09**
3L-CVRP25	100	193	193	2277.415	35.0	1.7	**1570.72**
3L-CVRP26	100	199	199	2558.33	36.5	1.4	**1847.95**
3L-CVRP27	100	198	198	2383.985	37.6	1.7	**1747.52**
Average				1425.8	31.5	0.8	1042.26

Bortfeldt and Yi (2020) tested their algorithm using these instances to show that their approach is designed for problems with weakly heterogeneous boxes. On average, their total transportation cost increased by 55.9% compared to Gendreau et al. (2006), and the average running time per instance was 254 s. The results of the 3L-CVRP instances are summarized in Table 7. The total transportation cost (*TTC*) is given for CGH and the tabu search algorithm (TS) from Gendreau et al. (2006). The average filling rates of the loading space as percentages (*Fr*) and running times in seconds (*Rt*) are given for CGH. CGH testing results on these instances show that the total transportation cost is increased by 36.8% compared to Gendreau et al. (2006), while the average running time is 0.8 s. Thus, CGH was able to find a solution with total transportation cost 12% lower than Bortfeldt and Yi (2020). The difference between the solution times is also very large. It should be noted that CGH is also designed for weakly heterogeneous boxes.

5.3. Real-world case study

The case study is a large company distributing home appliances in Isfahan province in Iran, which sends goods all over the country. Routing and packing data are both from the real world. Our data includes 34 instances for large sizes. Instances 1 to 29 were generated using the real information for the company's shipments sent on a specific day. Also, instances 30 to 34 were generated using company data to include more customers. The total data includes 797 different products, which have 188 different packaging dimensions. The smallest box is $28 \times 9 \times 20$ centimeters, and the largest box is $100 \times 99 \times 193$ centimeter. The minimum and maximum number of customers in an instance is 230 and 850, respectively. Also, the minimum and the maximum number of boxes in an instance is 929 and 6637, respectively. Each customer has ordered an average of five goods, and the maximum number of cus-

Table 8
Vehicle types.

Type	Length (*cm*)	Width (*cm*)	Height (*cm*)	Weight capacity (*kg*)
1	150	100	210	800
2	400	210	210	4000
3	480	210	210	6000
4	580	230	210	8000
5	1360	230	210	16,000

Table 9
Results for large real-world instances.

instance	*N*	*M*	*Bt*	CGH *TTC*	*Fr* (%)	*Rt* (s)
1	353	1489	90	61,926	50.2	755.1
2	439	1361	66	75,864	45.8	1322.7
3	282	1002	75	43,554	47.3	1273.8
4	311	929	57	56,579	48.0	2104.3
5	292	994	69	57,841	45.6	277.0
6	306	1017	74	45,837	51.0	1801.8
7	266	1227	77	46,788	52.7	793.5
8	296	1409	54	60,121	54.4	1013.6
9	293	1231	73	75,578	47.1	533.2
10	323	1453	70	53,906	54.8	2219.3
11	346	1961	65	94,018	49.0	977.8
12	347	1755	76	94,560	52.2	1226.5
13	246	1221	70	47,278	55.3	966.4
14	250	1170	59	44,394	55.6	675.4
15	275	1443	60	55,082	53.9	854.8
16	371	1439	64	78,611	53.2	1169.7
17	400	2517	83	103,226	55.8	700.8
18	253	1746	66	87,807	50.3	367.9
19	239	1200	66	48,157	58.0	582.4
20	248	1061	59	49,386	53.8	688.0
21	312	1380	65	47,510	49.3	2723.4
22	230	1277	85	41,663	50.7	722.8
23	621	3347	112	146,136	52.2	2070.8
24	418	3115	112	92,812	54.5	904.0
25	559	3278	98	129,453	56.3	1671.4
26	356	2174	60	59,366	52.7	2128.7
27	311	1542	65	64,026	48.0	3292.1
28	577	3214	100	137,074	53.4	1392.2
29	534	3059	96	125,986	49.0	3251.3
30	650	3696	103	152,011	56.5	2596.5
31	700	5079	99	250,865	54.9	639.0
32	750	4004	107	234,942	51.1	1555.6
33	800	6019	107	224,242	60.4	2409.1
34	850	6637	114	269,153	60.4	1834.2
Average				95,757.4	52.5	1396.9

Table 10
Results comparison with company approach.

instance	Company Approach *TTC*	*NV*	CGH *TTC*	*NV*
3	68,393	86	43,554	40
4	77,259	90	56,579	47
5	70,852	86	57,841	62
6	72,402	82	45,837	39
7	68,857	83	46,788	41
8	84,640	93	60,121	57
9	74,530	86	75,578	72
10	86,688	102	53,906	44
11	97,386	106	94,018	84
12	81,288	93	94,560	94
13	69,189	83	47,278	36
15	77,962	85	55,082	47
16	106,856	126	78,611	73
18	75,582	88	87,807	77
19	77,135	87	48,157	37
22	70,479	87	41,663	31
26	77,189	92	59,366	61
27	85,414	97	64,026	55
29	122,006	141	125,986	147
Average	81,268.8	94	65,092.5	60

tomer orders is 356. In the instances, there are customers whose boxes cannot be placed in the largest permissible vehicle. Therefore, forced splitting is necessary. The distance between customers is extracted from a roadmap of Iran. In each instance, five different types of vehicles are available; their information is shown in Table 8. The subproblems for each type of vehicle are solved in parallel to reduce the running time.

The results of the large real-world instances are summarized in Table 9. The columns are similar to Table 4. The *Bt* column shows the packaging variation. As can be seen, CGH was able to solve all instances in less than an hour. The average filling rate by considering all loading constraints is 52.5%.

5.3.1. Comparison with the current situation

The CGH method can be compared to the current situation of the company. The information for the vehicles loaded by the company is available for some instances, as shown in Table 10. In this table, the NV column shows the number of vehicles selected. As can be seen, the CGH algorithm reduces the average total transportation cost by 20% and the average number of vehicles by 36%. It should be noted that the company does not consider the type of vehicle and allocation-separation constraints. If it wants to comply with these constraints, the difference between the proposed method's solution quality and the company's current situation will increase.

5.3.2. Replanning

Sometimes, the company faces situations where customer orders are changed or new orders are added during vehicle loading operations. In such cases, due to the changes, it is necessary to find a suitable solution for 3L-SDHVRP-f in a few minutes so that the loading operation doesn't need to stop. For this purpose, only a heuristic algorithm can be used in the CGH process to solve the subproblem, instead of using a three-stage procedure. It should be noted that such conditions will occur during the loading operation and after sending some of the boxes. In this case, it is necessary to replan the remaining boxes in a short time. Table 11 shows the CGH algorithm results in cases where 75% of the boxes are loaded and then changes occur, and only GSA or LCA1 or LCA2 is employed to find the new column. As can be seen, GSA has the best average of total transportation cost, and LCA2 has the shortest running time. Suppose only GSA is used to solve the subproblems. In this case, the average total transportation cost will increase by 12% compared to using three heuristic methods to find a new column. However, the average running time is 15.8 s.

Table 12 shows the average total transportation cost and the average filling rate of all instances for cases where changes occur after loading 25%, 50%, and 75% of boxes. As can be seen, GSA performs better. In all three cases, the more boxes packed before the changes occur, the lower the total transportation cost and the higher the filling rate.

Table 11
Results for large real-world instances in a short time.

	CGH (SP(t) is solved by GSA)			CGH (SP(t) is solved by LCA1)			CGH (SP(t) is solved by LCA2)		
instance	*TTC*	*Fr* (%)	*Rt* (*s*)	*TTC*	*Fr* (%)	*Rt* (*s*)	*TTC*	*Fr* (%)	*Rt* (*s*)
1	74,308	49.7	3.5	74,214	49.2	2.5	76,336	48.4	1.0
2	91,183	45.9	40.2	83,103	43.1	13.9	101,571	42.6	5.0
3	56,974	46.8	71.3	47,101	45.9	5.8	60,174	44.1	2.1
4	61,486	47.0	6.6	58,952	45.9	8.7	60,738	48.1	4.4
5	62,420	45.3	2.3	60,448	45.5	2.6	63,741	45.7	1.9
6	54,277	48.4	16.1	53,841	47.7	4.1	63,893	45.6	2.1
7	51,955	52.1	1.2	52,563	52.2	0.7	55,922	52.7	0.8
8	69,316	53.8	7.5	63,258	54.6	9.3	69,754	53.2	3.7
9	75,690	47.4	9.8	75,055	47.4	3.1	76,696	47.2	2.5
10	60,203	54.4	1.0	65,531	52.5	1.6	62,903	52.5	1.2
11	120,195	46.3	2.8	115,753	45.4	5.1	107,029	48.2	2.6
12	93,552	51.5	5.0	92,813	50.9	17.5	96,708	51.6	4.2
13	50,353	54.0	12.0	54,763	49.0	4.4	56,512	50.8	2.8
14	51,980	51.4	10.4	46,308	50.3	4.9	54,101	52.6	2.1
15	70,735	50.0	0.7	61,114	52.5	2.1	62,599	52.3	1.7
16	82,707	53.8	43.3	82,251	53.0	14.3	89,371	52.4	6.6
17	117,955	52.8	3.9	106,563	55.6	33.0	112,861	56.0	5.8
18	91,833	49.6	1.6	89,905	49.1	0.5	91,271	48.9	0.5
19	53,698	56.5	1.5	59,185	53.2	1.7	54,272	54.4	0.8
20	51,819	52.2	4.4	50,631	51.6	7.4	53,079	51.6	1.3
21	50,949	48.8	27.6	57,387	44.0	10.2	58,423	45.9	4.0
22	46,420	50.9	6.9	48,162	45.8	2.8	49,209	46.4	2.1
23	156,244	51.2	24.4	183,395	46.1	16.6	164,644	51.2	8.1
24	104,161	53.5	14.4	100,712	53.0	11.8	109,930	51.2	4.6
25	150,861	55.3	13.0	139,774	54.8	15.6	150,221	54.2	6.6
26	67,312	50.1	10.9	60,232	51.8	10.1	72,710	48.2	5.0
27	75,292	48.4	15.3	72,924	47.9	9.1	79,879	46.5	3.7
28	155,044	52.7	11.7	162,018	48.8	17.8	150,160	53.3	18.6
29	135,707	48.3	12.5	161,818	39.6	9.9	128,692	48.8	17.8
30	163,577	55.6	26.8	194,621	48.6	9.1	170,917	55.4	11.6
31	265,266	55.6	14.6	336,667	46.1	3.8	265,246	55.3	16.6
32	248,188	51.1	49.2	289,497	43.9	42.4	247,819	51.5	47.5
33	266,780	58.6	25.2	312,352	50.1	3.6	269,436	58.8	12.2
34	329,746	54.3	40.9	375,609	49.4	25.6	300,733	59.4	68.5
Average	107,593.2	51.3	15.8	114,368.2	49.0	9.7	108,457.9	50.7	8.2

Table 12
Results for replanning.

	CGH (SP(t) is solved by GSA)		CGH (SP(t) is solved by LCA1)		CGH (SP(t) is solved by LCA2)	
loaded boxes (%)	*TTC*	*Fr* (%)	*TTC*	*Fr* (%)	*TTC*	*Fr* (%)
25	119,199.9	48.2	176,510.5	36.1	137,245.9	47.4
50	110,631.4	49.9	142,154.3	42.2	119,809.1	49.4
75	107,593.7	51.3	114,368.2	49.0	108,457.9	50.7

6. Conclusions and suggestions for future research

We investigated the integrated vehicle routing and three-dimensional loading problem with specific characteristics. If a customer demand cannot be packed in the largest permissible vehicle, they are visited more than once. Vehicles are heterogeneous, and the customer may prohibit transporting the boxes in one or more types of vehicles. The orientation, stacking, vertical stability, LIFO, reachability, allocation-separation, allocation-connectivity, and complexity constraints are considered. The last three are considered for the first time in the integrated routing and loading problem. We introduced a CGH algorithm, which is a column generation-based heuristic. Three heuristic methods were proposed to solve subproblems. The hybrid heuristic algorithm was presented for loading, which can produce executable and straightforward loading patterns for the operator, considering all constraints, in a short time.

Computational experiments on instances from the literature showed that CGH could obtain solutions very close to those obtained by algorithm solutions in the literature with significantly shorter running times. CGH was also able to solve large-scale real-world instances with a minimum of 230 and a maximum of 850 customers in less than an hour, with an average filling rate of 52.5%, which is very useful for the distribution company. Compared to the company's current situation, CGH reduced the average total transportation cost by 20% and the average number of vehicles by 36%.

The column generation technique can also be used for integrated routing and loading problems with other practical constraints, such as time window constraints. The CGH algorithm's performance is highly dependent on the methods for solving the subproblem and loading, so it is recommended that efficient methods for solving the subproblem be developed. Future research could also provide a heuristic loading method that can produce loading patterns with more suitable filling rates in a short time. Improving the procedure for deriving an integer solution could also increase the efficiency of the algorithm. (Defination 1).

References

Bischoff, E. E., & Ratcliff, M. (1995). Issues in the development of approaches to container loading. *Omega, 23*(4), 377–390.
Bortfeldt, A. (2012). A hybrid algorithm for the capacitated vehicle routing problem with three-dimensional loading constraints. *Computers & Operations Research, 39*(9), 2248–2257.
Bortfeldt, A., & Homberger, J. (2013). Packing first, routing second—A heuristic for the vehicle routing and loading problem. *Computers & Operations Research, 40*(3), 873–885.
Bortfeldt, A., & Wäscher, G. (2013). Constraints in container loading–A state-of-the-art review. *European Journal of Operational Research, 229*(1), 1–20.
Bortfeldt, A., & Yi, J. (2020). The split delivery vehicle routing problem with three-dimensional loading constraints. *European Journal of Operational Research, 282*(2), 545–558.
Ceschia, S., Schaerf, A., & Stützle, T. (2013). Local search techniques for a routing–packing problem. *Computers & Industrial Engineering, 66*(4), 1138–1149.
Christofides, N. (1979). The vehicle routing problem. In N. Christofides, A. Mingozzi, P. Toth, & C. Sandi (Eds.), *Combined optimization*. Chichester, UK: Wiley.
Crainic, T. G., Perboli, G., & Tadei, R. (2008). Extreme point-based heuristics for three-dimensional bin packing. *Informs Journal on Computing, 20*(3), 368–384.
Desrochers, M. (1988). *An algorithm for the shortest path problem with resource constraints.*
Fuellerer, G., Doerner, K. F., Hartl, R. F., & Iori, M. (2010). Metaheuristics for vehicle routing problems with three-dimensional loading constraints. *European Journal of Operational Research, 201*(3), 751–759.
Gendreau, M., Iori, M., Laporte, G., & Martello, S. (2006). A tabu search algorithm for a routing and container loading problem. *Transportation Science, 40*(3), 342–350.
Hokama, P., Miyazawa, F. K., & Xavier, E. C. (2016). A branch-and-cut approach for the vehicle routing problem with loading constraints. *Expert Systems with Applications, 47*, 1–13.
Hu, Z.-. H., Zhao, Y., Tao, S., & Sheng, Z.-. H. (2015). Finished-vehicle transporter routing problem solved by loading pattern discovery. *Annals of Operations Research, 234*(1), 37–56.
Iori, M., & Martello, S. (2010). Routing problems with loading constraints. *Top, 18*(1), 4–27.
Junqueira, L., Oliveira, J. F., Carravilla, M. A., & Morabito, R. (2013). An optimization model for the vehicle routing problem with practical three-dimensional loading constraints. *International Transactions in Operational Research, 20*(5), 645–666.
Koch, H., Bortfeldt, A., & Wäscher, G. (2018). A hybrid algorithm for the vehicle routing problem with backhauls, time windows and three-dimensional loading constraints. *OR Spectrum*, 1–47.
Lacomme, P., Toussaint, H., & Duhamel, C. (2013). A GRASP× ELS for the vehicle routing problem with basic three-dimensional loading constraints. *Engineering Applications of Artificial Intelligence, 26*(8), 1795–1810.
Li, X., Yuan, M., Chen, D., Yao, J., & Zeng, J. (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*.
Mahvash, B., Awasthi, A., & Chauhan, S. (2017). A column generation based heuristic for the capacitated vehicle routing problem with three-dimensional loading constraints. *International Journal of Production Research, 55*(6), 1730–1747.
Mak-Hau, V., Moser, I., & Aleti, A. (2018). An exact algorithm for the heterogeneous fleet vehicle routing problem with time windows and three-dimensional loading constraints. *Data and decision sciences in action* (pp. 91–101). Springer.
Männel, D., & Bortfeldt, A. (2016). A hybrid algorithm for the vehicle routing problem with pickup and delivery and three-dimensional loading constraints. *European Journal of Operational Research, 254*(3), 840–858.
Männel, D., & Bortfeldt, A. (2018). Solving the pickup and delivery problem with three-dimensional loading constraints and reloading ban. *European Journal of Operational Research, 264*(1), 119–137.
Moura, A. (2008). A multi-objective genetic algorithm for the vehicle routing with time windows and loading problem. *Intelligent decision support* (pp. 187–201). Springer.
Moura, A. (2019). A model-based heuristic to the vehicle routing and loading problem. *International Transactions in Operational Research, 26*(3), 888–907.
Moura, A., & Oliveira, J. F. (2009). An integrated approach to the vehicle routing and container loading problems. *OR Spectrum, 31*(4), 775–800.
Pace, S., Turky, A., Moser, I., & Aleti, A. (2015). Distributing fibre boards: A practical application of the heterogeneous fleet vehicle routing problem with time windows and three-dimensional loading constraints. *Procedia Computer Science, 51*, 2257–2266.
Pollaris, H., Braekers, K., Caris, A., Janssens, G. K., & Limbourg, S. (2015). Vehicle routing problems with loading constraints: State-of-the-art and future directions. *OR Spectrum, 37*(2), 297–330.
Pollaris, H., Braekers, K., Caris, A., Janssens, G. K., & Limbourg, S. (2017). Iterated local search for the capacitated vehicle routing problem with sequence-based pallet loading and axle weight constraints. *Networks, 69*(3), 304–316.
Reil, S., Bortfeldt, A., & Mönch, L. (2018). Heuristics for vehicle routing problems with backhauls, time windows, and 3D loading constraints. *European Journal of Operational Research, 266*(3), 877–894.
Ruan, Q., Zhang, Z., Miao, L., & Shen, H. (2013). A hybrid approach for the vehicle routing problem with three-dimensional loading constraints. *Computers & Operations Research, 40*(6), 1579–1589.
Song, X., Jones, D., Asgari, N., & Pigden, T. (2019). Multi-objective vehicle routing and loading with time window constraints: A real-life application. *Annals of Operations Research*, 1–27.
Tao, Y., & Wang, F. (2015). An effective tabu search approach with improved loading algorithms for the 3L-CVRP. *Computers & Operations Research, 55*, 127–140.
Tarantilis, C. D., Zachariadis, E. E., & Kiranoudis, C. T. (2009). A hybrid metaheuristic algorithm for the integrated vehicle routing and three-dimensional container-loading problem. *IEEE Transactions on Intelligent Transportation Systems, 10*(2), 255–271.
Vega-Mejía, C. A., Montoya-Torres, J. R., & Islam, S. M. (2019). A nonlinear optimization model for the balanced vehicle routing problem with loading constraints. *International Transactions in Operational Research, 26*(3), 794–835.

Wei, L., Zhang, Z., & Lim, A. (2014). An adaptive variable neighborhood search for a heterogeneous fleet vehicle routing problem with three-dimensional loading constraints. *IEEE Computational Intelligence Magazine, 9*(4), 18–30.
Yi, J., & Bortfeldt, A. (2018). The capacitated vehicle routing problem with three-dimensional loading constraints and split delivery—A case study. In *Operations Research Proceedings 2016* (pp. 351–356). Springer.
Zachariadis, E. E., Tarantilis, C. D., & Kiranoudis, C. T. (2012). The pallet-packing vehicle routing problem. *Transportation Science, 46*(3), 341–358.
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. *Transportation Research Part B: Methodological, 82*, 20–35.
Zhu, W., Qin, H., Lim, A., & Wang, L. (2012). A two-stage tabu search algorithm with enhanced packing heuristics for the 3L-CVRP and M3L-CVRP. *Computers & Operations Research, 39*(9), 2178–2195.