THE HETEROGENEOUS FLEET LOCATION ROUTING PROBLEM WITH SIMULTANEOUS PICKUP AND DELIVERY AND OVERLOADS

XUEFENG WANG*

School of Business Administration, Jiangxi University of Finance and Economics
Nanchang 30013, China

Abstract. This paper addresses a new variant of the location routing problem (LRP), namely the heterogeneous fleet LRP with simultaneous pickup and delivery and overloads (HFLRPSPDO) which has not been previously tackled in literatures. In this problem, the heterogeneous fleet is comprised of vehicles with different capacities, and the vehicle overloads up to a specified upper bound is allowed. This paper proposes a polynomial-size mixed integer linear programming formulation for the problem in which a penalty function, allowing capacity violations of vehicles, is integrated into objective function. Furthermore, two heuristic algorithms, respectively based on tabu search and simulated annealing, are proposed to solve HFLRPSPDO. Computational results on simulated instances show the effectiveness of the proposed problem formulation and the efficiency of the proposed heuristic algorithms.

Introduction. In supply chain and logistics management, operation service often involves sending vehicles from depots to simultaneously meet the customers’ delivery and pickup demands in specific time windows. From the view of integrating strategic and tactical decisions which are closely interdependent, the academic community has studied several versions of location routing problem with simultaneous pickup and delivery, and proposed an immense list of solution approaches ranging from simple heuristics to complex meta-heuristics and exact methods. Most researchers have concentrated on simplified instances of the problem, i.e., instances that do not accommodate constraints or objectives often encountered in practice, which guide the actual solutions in real location routing problem [11]. In this paper, we address the heterogeneous fleet location routing problem with delivery and pickup and overloads (HFLRPSPDO), motivated by the fact the real-life distribution scheduling involves, to some extent, time window restrictions and small capacity violations in most distribution scenarios. The HFLRPSPDO often appears in a number of reverse logistics contexts, and has broad prospects in the theory research and practical application. One typical example in China can be described as: many third-party logistics enterprises need to locate multiple candidate depots, and dispatch heterogeneous fleet to provide bi-directional logistics service for the customers distributed in urban and rural areas. The third-party logistics enterprises

2010 Mathematics Subject Classification. 68U99.
Key words and phrases. Heterogeneous fleet, location-routing problem, simultaneous pickup and delivery, overloads.

* Corresponding author: Xuefeng Wang.
are responsible for distributing commodity from depots to customers, and collecting recycled materials or other reversed commodity from customers to depots. In practical operation, the truck overloads is often enforced by peak hours of customer demands. So the decisions involving in distribution system design include where to locate the depots and how to serve customers within the given time windows using heterogeneous fleet of vehicles.

The HFLRPSPDO can be described as follows: Consider a potential depot set and a heterogeneous fleet of vehicles with different capacities, a set of customers, geographically dispersed in given area, are required to be served by the depots and vehicles. Each customer's pickup and delivery demands must be simultaneously satisfied within a specified time window. In the distribution process, the vehicles are allowed to carry loads over capacity to meet the demand peak at the expense of penalty cost. The decision embedded in HFLRPSPDO is to determine the optimal number and location of depots and distribution routes to satisfy all customer demand under the capacity and time window constraints. The objective function of the problem is to minimize the sum of depot location cost, fixed vehicle cost, routing cost, and the penalty cost of capacity violation. The HFLRPSPDO can be considered as an extension of the LRP or LRPSPD in terms of the vehicle types, the composition of customer demand, and the capacity violation of vehicle. It implies that this more complicated integrated problem is NP-hard. So the approximation methods are more appropriate than exact methods for solving this problem. Some mathematical models and exact solution procedures have been developed for small- and medium-size LRPs in the literature [1,2,14,15]. Exact solutions obtained from these literatures can serve as lower bounds for heuristics validation. Different heuristic approaches have been also commonly proposed to solve large-size LRPs. Min [19] and Schwardt and Dethloff [26] use clustering-based method for the LRP by partitioning the customer set into one cluster per potential depot or one per vehicle route, and then solving a TSP or VRP for each cluster. Meanwhile iterative LRP heuristics have been successfully implemented for the problem through decomposing the problem into its two constituent sub-problems and iteratively solving the sub-problems. Several literature of the application of iterative heuristics include Hansen et al. [7], Wu et al. [29]. In order to realize the routing information can be utilized well in the location phase, the hierarchical heuristics are also popularly applied in solving the LRP, such as Srivastave [27], Chien [3] and Nagy and Salhi [22], and Lin et al. [16]. Comprehensive reviews of the location routing models and their applications are provided in Min et al. [20] and Nagy and Salhi [21]. To the best of our knowledge, so far, research related to the HFLRPSPDO is non-existent. Past literature mostly focused on homogeneous vehicle fleets, and the vehicular capacity is usually not allowed to violate. There only two related literatures addressing the LRP with simultaneous pickup and delivery (LRPSPD). Karaoglan et al. [12] proposed two polynomial-size mixed integer linear programming formulations, including node-based and flow-based, for the LRPSPD. A family of valid inequalities is adopted to strengthen the formulations, and a two-phased heuristic approach based on simulated annealing is proposed to solve the large-size LRPSPD instances derived from the literature. Karaoglan et al. [11] proposed an effective branch-and-cut algorithm for solving the LRPSPD strengthen by several valid inequalities adapted from the literature. However, the research on the LRP in the past seems to have ignored the fact that the fleet comprises vehicles with different capacities. The study by Salhi and Fraser [24] and Wu et al. [29] seems to be the only ones in which
the vehicles used in the LRP do not necessarily have the same capacities. In this paper, we address for the first time the location routing problem with simultaneous pickup and delivery and overloads when the fleet is heterogeneous, i.e., comprises vehicles of different capacities and associated costs. The overload is allowed up to a pre-specified bound at a penalty cost by measuring the deviation of the actual load from the nominal vehicle capacity [13].

The remainder of this paper is organized as follows. In section 1 the HFLRP-SPDO is formulated as a mixed integer programming mathematical model. This formulation extends the earlier work by considering heterogeneous fleet and time window constraints. Section 2 offers the overall structure of heuristic approach motivated by the integration of the location and routing decisions. Two heuristics approaches respectively based on tabu search (TS) and simulated annealing (SA) are proposed to solve the problem. Section 3 presents the computational experiment to test the performance of the model formulation and heuristic approaches. Finally, conclusion and future directions are discussed in the Section 4.

1. Problem definition and mathematical formulation. In this section, we first define the problem and present a polynomial-size mixed integer linear programming formulation for the problem.

1.1. Problem definition. The HFLRPSPDO is defined on a complete directed network graph \( G = (N,A) \). \( N = N_0 \cup N_C \) is the set of nodes in which \( N_0 \) and \( N_C \) represent the sets of potential depots and customers, respectively. \( A = \{(i,j) : i,j \in N\} \) is the set of arcs in graph \( G \). \( K \) is the set of vehicles or routes, and indexed by subscript \( k \). Each arc \((i,j) \in A\) has a nonnegative cost \( c_{ijk} \) and travel time \( t_{ijk} \) related to vehicle \( k \) \((k \in K)\), and triangular inequality holds \((i.e., c_{ijk} + c_{juk} \geq c_{iuk})\). A capacity \( CD_j \) and a fixed cost \( FD_j \) are associated with each potential depot \( j \in N_0 \). The customers are served with a heterogeneous fleet of vehicles with capacity \( CV_k \) and fixed operating cost \( FV_k \) \((k \in K)\). The set of customers is indexed by subscript \( i \). Each customer \( i \in N_C \) requires pickup demand \( p_i \) and delivery demand \( d_i \), with \( 0 < d_i, p_i \leq CV_k \), and must be visited once for both operations. The other parameters for each customer include earliest service start time, \( ET_i \), and latest service start time, \( LT_i \), and specified service time, \( ST_i \); which is the time spent by the vehicle to load and unload the commodity of customer \( i \). The time window \([ET_i,LT_i]\) of customer \( i \in N_C \) can be treated as soft constraint allowing the vehicle to arrive to a customer before the start of the time window but it has to wait if it arrives before the earliest service start time. The constraint of latest service start time must be strictly satisfied in any feasible solution. Hence, the total routing time of a vehicle is the sum of traveling time, waiting time and service time. Each vehicle leaves the depot carrying an amount of commodity equal to the total amount it must delivery, and returns to the depot carrying an amount of returning commodity equal to the total amount it pickup. The loading of vehicle \( k \) in route section \((i,j)\) is \( r_{ijk} \) \((i,j \in O, k \in K)\). The duration of each route is limited by forcing the last customer to be served within the specified time window. Note that capacity constraint is partially relaxed in the HFLRPSPDO with unit penalty parameter \( \lambda_k \). The problem is to determine the depot location, the assignment of customers to the opened depots and the corresponding vehicle routes with a minimum cost objective function under the following additional constraints.

- Each customer must be visited exactly once and fulfilled from a single depot.
• Each route is served by one vehicle, and each route begins and ends at the same depot.
• Multiple types of vehicles with different capacity can be used to service customers distributed in a given area.
• The capacity violation of vehicle is allowed but limited to a predefined upper bound.

1.2. Mathematical formulation. As mentioned previously, the HFLRPSPDO consists of two main problems namely the facility location problem and the vehicle routing problem with simultaneous pickup and delivery and time window. We constructed a general mixed integer linear programming formulation for the problem. The decision variables and auxiliary variables used in the formulation are given as follows:

\[ x_{ijk} : \text{traveling variable of a vehicle } k \in K, \text{ if it travels directly from node } i \text{ to node } j \ (\forall i, j \in N), x_{ijk} = 1; \text{ otherwise } x_{ijk} = 0. \]

\[ y_j : \text{opening variable of a depot } j \in N_0; \text{ if depot } j \text{ is opened, } y_j = 1; \text{ otherwise } y_j = 0. \]

\[ z_{ij} : \text{ assignment variable of a customer } i \in N_C; \text{ if customer } i \text{ is assigned to depot } j \ (\forall i \in N_C, \forall j \in N_0), z_{ij} = 1; \text{ otherwise } z_{ij} = 0. \]

\[ U_{ij} : \text{the demand delivered to customers routed after node } i \text{ and transported in arc } (i, j). \]

\[ V_{ij} : \text{the demand picked up from customers up to node } i \text{ and transported in arc } (i, j). \]

\[ Q_k : \text{the load of vehicle } k \ (\forall k \in K). \]

\[ B_i : \text{beginning time to service customer } i \ (\forall i \in N_C). \]

\[ A_i : \text{arriving time at customer } i \ (\forall i \in N_C). \]

The proposed formulation for HFLRPSPDO is as follows:

\[
\begin{align*}
\text{Min} & \quad \sum_{j \in N_0} F_D_j \cdot y_j + \sum_{i \in N} \sum_{j \in N} \sum_{k \in K} c_{ijk} \cdot x_{ijk} + \sum_{i \in N_0} \sum_{j \in N_C} \sum_{k \in K} F V_k \cdot x_{ijk} \\
& \qquad + \sum_{k \in K} \frac{1}{2} \lambda_k [(Q_k - CV_k)] + (Q_k - CV_k) \\
\text{Subject to:} & \\
\sum_{i \in N} \sum_{k \in K} x_{ijk} = 1 & \forall j \in N_C \quad (2) \\
\sum_{j \in N} \sum_{k \in K} x_{ijk} - \sum_{j \in N} \sum_{k \in K} x_{ijk} = 0 & \forall i \in N_C \quad (3) \\
\sum_{i \in N_C} \sum_{j \in N_0} x_{ijk} \leq 1 & \forall k \in K \quad (4) \\
\sum_{j \in N_0} z_{ij} = 1 & \forall i \in N_C \quad (5) \\
\sum_{k \in K} x_{ijk} \leq z_{ij} & \forall i \in N_C, \forall j \in N_0 \quad (6) \\
\sum_{k \in K} x_{ijk} \leq z_{ij} & \forall i \in N_C, \forall j \in N_0 \quad (7)
\end{align*}
\]
\[ x_{ijk} + z_{in} + \sum_{m \in N_0, m \neq n} z_{jm} \leq 2 \quad \forall i, j \in N_C, \ i \neq j, \ \forall n \in N_0, \ k \in K \] (8)

\[ \sum_{j \in N} U_{ji} - \sum_{j \in N} U_{ij} = d_i \quad \forall i \in N_C \] (9)

\[ \sum_{j \in N} V_{ij} - \sum_{j \in N} V_{ji} = p_i \quad \forall i \in N_C \] (10)

\[ \sum_{i \in N_C} d_i \cdot z_{ij} \leq CD_j \cdot y_j \quad \forall j \in N_0 \] (11)

\[ \sum_{i \in N_C} p_i \cdot z_{ij} \leq CD_j \cdot y_j \quad \forall j \in N_0 \] (12)

\[ U_{ij} + V_{ij} \leq CV_k \cdot (1 + \alpha_k) \cdot \sum_{k \in K} x_{ijk} \quad \forall i, j \in N \text{ and } 0 \leq \alpha_k \leq 1 \] (13)

\[ \sum_{j \in N_C} U_{ij} = \sum_{j \in N_C} z_{ji} \cdot d_j \quad \forall i \in N_0 \] (14)

\[ \sum_{j \in N_C} U_{ji} = 0 \quad \forall i \in N_0 \] (15)

\[ \sum_{j \in N_C} V_{ji} = \sum_{j \in N_C} p_j \cdot z_{ji} \quad \forall i \in N_0 \] (16)

\[ \sum_{j \in N_C} V_{ji} = 0 \quad \forall i \in N_0 \] (17)

\[ U_{ij} \leq \sum_{k \in K} (CV_k - d_i) \cdot x_{ijk} \quad \forall i \in N_C, \ \forall j \in N, \ \forall k \in K \] (18)

\[ V_{ij} \leq \sum_{k \in K} (CV_k - p_j) \cdot x_{ijk} \quad \forall i \in N, \ \forall j \in N_C, \ \forall k \in K \] (19)

\[ U_{ij} \geq d_j \cdot x_{ijk} \quad \forall i \in N, \ \forall j \in N_C, \ \forall k \in K \] (20)

\[ V_{ij} \geq p_j \cdot x_{ijk} \quad \forall i \in N_C, \ \forall j \in N, \ \forall k \in K \] (21)

\[ A_i = \sum_{k \in K, j \in N} x_{ijk} \cdot (B_j + ST_j + t_{ijk}) \quad \forall j \in N_C \] (22)

\[ A_i \leq B_i \leq LT_i \quad \forall i \in N_C \] (23)

\[ ET_i \leq B_i \leq LT_i \quad \forall i \in N_C \] (24)

\[ x_{ijk} \in \{0, 1\} \quad \forall i, j \in N, \ \forall k \in K \] (25)

\[ y_j \in \{0, 1\} \quad \forall j \in N_0 \] (26)

\[ z_{ij} \in \{0, 1\} \quad \forall i \in N_C, \ \forall j \in N_0 \] (27)

In the above formulation, the objective function (1) is to minimize the sum of depot fixed cost, transportation cost, vehicle acquiring cost, and capacity violation cost. The objective function allows capacity violations in some vehicles but to penalize the resulting overloads. Constraints (2) and (3) are known as degree constraints. Constraint (2) ensures each customer only appears in one route and be visited by a suitable vehicle. Constraint (3) guarantees that the same vehicle arrives and departs from each customer it serves. Constraint (4) ensures each vehicle only service for one depot. Constraint (5) ensures that each customer must be assigned to only one depot. Constraints (6)–(8) forbid the illegal routes, i.e. the routes, which do
not star and end at the same depot. Restriction (9) and (10) are flow equations for pickup and delivery demands, respectively. Constraints (11) and (12) guarantee that the total delivery and pickup loaded on any depot do not exceed the corresponding depot capacity, separately. Constraint (13) allows the total loading of each vehicle up to a percentage of the total capacity, which is determined by the scheduler through bound parameter $\alpha_k$. Constraint (14) ensures that the total delivery loads dispatched from each opened depot equals to the total delivery demand of customers, which are assigned to the corresponding depot. Constraint (15) guarantees that the total delivery loads returning to the opened depot must be equal to zero. Constraint (16) ensures that the total pickup load entering each opened depot equals to the total pickup demand of customers, which are assigned to the corresponding depot. Constraints (17) guarantees that the total pickup loads dispatched from each opened depot must be equal to zero. Constraints (18)–(21) are bounding constraints for the additional variables. The constraint (22) represents the arriving time at each customer. The constraint (23) and (24) are service time window constraints at each customer. Constraint (22), (23) and (24) guarantee that the relationship among arriving time, departure time, and service time with respect to each customer is compatible to the customer’s time window. Finally, constraints (25)–(27) are known as integrality constraints which define the nature of the decision variables. In order to deal with the absolute value existing in objective function, the objective function can be further transform into the following equation (28), and the constraints (29) and (30) are added.

$$\text{Min } \sum_{j \in N_0} FD_j \cdot y_j + \sum_{i \in N} \sum_{j \in N} \sum_{k \in K} c_{ijk} \cdot x_{ijk} + \sum_{i \in N_0} \sum_{j \in N_C} \sum_{k \in K} FV_k \cdot x_{ijk}$$

$$+ \sum_{k \in K} \lambda_k \cdot \left[(Q_k - CV_k)\right] + 2\delta$$

$$\frac{1}{2} \lambda_k \cdot (CV_k - Q_k) - \delta \leq 0$$

$$\delta \geq 0$$

2. Solution approach. Meta-heuristics such as tabu search and simulated annealing can search the solution space more efficiently than conventional approaches using different strategies. These meta-heuristics show great promises in solution of difficult combinational problems such as LRP [26, 29]. As the HFLRPSPDO proposed in this paper is a NP-hard problem, heuristic methods are necessary to quickly obtain solutions for large-size problems. The design of the proposed solution approaches aims to achieve two purposes.

- To examine the effect of capacity violations of vehicles on the problem solution.
- Explore and compare the ability of two proposed heuristic algorithms in solving the problem.

In this paper, we propose two heuristic algorithms respectively based on tabu search (TS-heuristics) and simulated annealing (SA-heuristics) which consist of three phases, called initialization, location and routing phases, to solve the HFLRPSPDO. The heuristic algorithms designed here attempt to tackle the problem hierarchically in a computationally efficient manner. It is an extension of heuristic algorithm in Lin et al. [17] for the LRP with homogeneous fleets. Our algorithm differs from the previous ones in terms of the not only the problem characteristics that it is applied but also moving strategies and constraint handling method used
in the location and routing phase. In initialization phase, a heuristic procedure is proposed to generate an initial solution for the problem. In the location phase, a hierarchical search, based on tabu search or simulated annealing, is applied to perform on the location variables to determine a preferred depot configuration. In the routing phase, another search based on tabu search or simulated annealing will run on the routing variables in order to obtain a good routing design for a given location configuration. The three search phases are coordinated so that an efficient exploration of the solution space is performed. In the following sections, the proposed TS-heuristics and SA-heuristics of our paper will be discussed in detail, including initial solution, neighborhood generation, fitness evaluation and algorithmic parameters.

2.1. Initial solution by saving algorithm. Since initial solutions obtained from heuristics are closer to the optimal solutions than the random ones, it is generally preferred to obtain initial solutions to save computational time of the search algorithm. In initialization phase, the Clarke and Wright heuristics [5], which is robust enough to give a reasonably good solution, can be easily modified to deal with vehicle capacity, time window and other constraints.

2.1.1. A lower bound on the number of depots to be established. The first procedure in the initialization phase is to determine a lower bound \( n_L \) on the number of depots to be established, which is derived from total customer demands and maximum depot capacity.

\[
n_L = \left\lceil \frac{\sum_{i \in \mathbb{N}} (p_i + d_i)}{\text{Max} \{CD_j, j \in \mathbb{N}_0\}} \right\rceil \quad (31)
\]

where \( \lceil x \rceil \) is the smallest integer larger than or equal to \( x \). After the lower bound \( n_L \) is determined, the initial clustering and routing procedures will be performed to form the customer clusters and routes.

2.1.2. Initial clustering. The second procedure in the initialization phase is to determine the initial clustering by identifying the nearest depot for each customer. A set of \( n_L \) depots will be determined in descending order of the number of its nearest customers. Then the customers can be listed in non-increasing order of their penalty costs which is calculated when the customer is assigned to the second closest depot instead of the closest one. Through sequentially assigning customers to the nearest open depot that has sufficient handling capacity, the clusters of customers can be found. If partial customers are not assigned to their closest depot because of the depot’s capacity limitation, the assignment of these customers to appropriate depot with enough capacity is tried again, and their penalty costs are recalculated. Furthermore, if any customer is left unassigned, it will be assigned to the depot with the least workload although the initial depot capacity will be exceeded.

2.1.3. Initial routing. Given the initial clustering results, a routing procedure in the initialization phase starts with constructing separate routes for each customer from its depot. Then, these routes belonging to the same depot are iteratively inserted based on cost saving obtained by their insertions. All feasible solutions in insertion process are examined in terms of cost saving obtained by the insertions, then the insertion providing the largest cost saving is adopted. This strategy ensures that a customer selected for insertion attempts to maximize the benefit derived from inserting this customer on the partial route being constructed, rather than
commencing a new route. This strategy is repeated until no-capacity-feasible or no-time window-feasible combination is found. This procedure terminates with a complete initial routing solution that satisfies all problem constraints. The elements of the proposed solution method are examined in detail below.

**Cost evaluation of inserting customer:**

To evaluate the cost saving obtained by insertion, a list of cost criteria proposed by Golden et al. [8] and Kritikos et al. [13] can be applied in the evaluation process. Let \( i \) and \( j \) be two consecutive customers in a partially constructed route \( R \), and \( u \) be another uncombined route (customer). The first and second sub-criterion express the time window compatibility of the selected customer with the specific insertion place in the current route (see Loannou et al. [9]). The first sub-criteria is the time gap between the service starting time \( B_u \) and the lower bound of time window \( ET_u \), denoted by \( C_1(i, u, j) = B_u - ET_u \). The second sub-criteria is the time gap between the service starting time \( B_u \) and the upper bound \( LT_u \) of time window, denoted by \( C_2(i, u, j) = LT_u - B_u \). In addition, the insertion of \( u \) between \( i \) and \( j \) will result in a distance increase. So the third sub-criteria \( C_3(i, u, j) = c_{iuk} - c_{ujk} - c_{ijk} \) can also included in \( C(i, u, j) \). The fourth sub-criteria \( C_4(i, u, j) \) models the different of vehicle fixed cost and capacity penalty transferred by vehicle \( k \) prior and after the insertion of customer into the route. If a vehicle of nominal capacity of \( CV_k \) is loaded over this capacity, then this is allowed at a penalty, determined together with the vehicle’s fixed cost by the relationship: \( F(Q_k) = FV_k + \lambda_k (Q_k - CV_k) \). If the vehicle is loaded up to \( CV_k \), while exceeding the bound imposed by the vehicles of smaller capacity, then the penalty is zero and the following holds: \( F(Q_k) = FV_k \). The fourth penalty component is, thus, defined as \( C_4(i, u, j) = P(Q_k) \) for the active vehicle \( k \) for which the current route is constructed. The function \( C(i, u, j) \) is a composite criterion by appropriately weighting several sub-criteria, which can measures the cost of inserting \( u \) between \( i \) and \( j \). Thus, the cost function \( C(i, u, j) \) is given by the following relationship:

\[
C(i, u, j) = \sum_{\xi=1,...,4} \gamma_{\xi} \cdot C_{\xi}(i, u, j)
\]  

(32)

where \( \gamma_{\xi} \) weights define the relative contribution of each individual sub-criterion to overall selection criterion. We can define the customer insertion criterion, i.e., the customer that is best according to the selection criterion: \( \max_u [C(i, u, j)] \).

The general steps of the Initial routing procedure in initialization phase could be summarized as follows.

**Step 1:** Put all customers into each set of depots according to the initial customer assignment. **Step 2:** Open and sort lists for customers and vehicles.

**Step 2.1:** Open a customer list \( L \), and sort customers by increasing values of the lower bound \( ET_i \) of time window. If several nodes have the same \( ET_i \), arrange them by increasing values of the upper bound of time window \( LT_i \).

**Step 2.2:** Open a list of available vehicles \( V \) and sort them by decreasing values of the ratio \( (CV_k/FV_k) \).

**Step 3:** Open an empty list \( K_n \) linked to the next cluster (route) \( C_n \) to be created. Assign the top vehicle in list \( V \) to serve the customers in \( C_n \) and delete it from \( V \).
Step 4: Pick up the top node on the customer list; initialize the parameters of cluster (route), and delete it from list \( L \) and make a copy of the current list \( L \) and call it \( L^* \).

Step 5: Randomly choose customer \( u \) from list \( L^* \) to the best insertion location on the current route, If the vehicle capacity and the time window constraints are met; If not, delete the customer from list \( L^* \), repeat step 4. Otherwise, proceed to step 6.

Step 6: If the list \( L^* \) is empty, compute the \( C(i,u,j) \) and save the list defining the cluster (route) \( C_n \), and proceed to Step 7. Otherwise return to step 4.

Step 7: Choose the customer with the minimal value of \( C(i,u,j) \). Delete it from the list \( L^* \).

Step 8: If the list \( L \) is empty, save the list \( K_n \) defining the cluster \( C_n \), and proceed to step 9. Otherwise return to step 4.

Step 9: Repeat the step 3-8 until the list \( L \) is empty.

Step 10: Terminated; output the cluster set including the active vehicles, the routes, sequence of customers visited by each vehicle, total distance and total cost.

2.2. Neighborhood search.

2.2.1. Neighborhood search in location phase. In location phase, a neighborhood search operation is randomly selected in add or swap move, and it is applied to the best solution obtained in the routing phase to determine a new location configuration.

**Swap move:** This operation keeps the number of opened depots in the solution constant, and searches for a good configuration for a certain number of depots. For a given depot set in the current solution, a depot, which is randomly selected from the set of opened depot, is closed, and another one which is randomly selected from the set of closed depots is open simultaneously. Then the routes belonging to the closed depot are reassigned to opened depots with shortest path to the centroids of these routes. The difference in routing cost can be estimated using the difference in the direct distance between the centroid of route and the depot according to the new and old assignment. When the swap move yielding the largest fitness evaluation value is performed, both the move and its reverse are declare tabu for a given number of iterations, and the search resumes to the routing phase in order to update the routings according to the swap move. Swap moves are terminated after a \textit{max-swap} number of iterations are preformed without improvement over the best solution found.

**Add move:** This operation opens one of closed depots in the current solution, and therefore increases the number of opened depots by one. Some of the routes in the opened depots are reassigned to the recently opened depot with shortest path to the centroids of these routes. As in the swap move, the search again returns to the routing phase in order to update the routing after the add move. Before accepting any move, restrictions from the depot and vehicle capacity limitation, time window constraint, and the tabu list must be checked. The process is repeated until \textit{max-add} threshold for termination are performed without any improvement over the best fitness evaluation value.

2.2.2. Neighborhood search in routing phase. To improve on the routing costs in the initial HFLRPSPDO solution obtained in the initialization phase and the location phase, the routing phase starts from the best routing result found for the previous depot configuration, and improves the solution within routes and between routes until a stopping criterion is met. Before proceeding the routing phase, the vehicles
in the heterogeneous fleet are assigned to customer clusters according to the order from large to small vehicle capacities. After assigning vehicles, insert and swap moves are employed sequentially in the routing phase. The search strategy used in this phase can refer to the two-phase tabu search approach proposed by Tuzun et al. [28].

**Insert move:** This operation only inserts a customer, which is assigned to a depot that with customer assignment change, to a new position on the other routes originating from its current depot, or any other open depot that is close enough to the customer. Firstly, randomly select a route from the inserted customer’s current depot or other open depots, then to find two customers in the route that is nearest to the selected customer, and insert the selected customer into a position between the two customers. In order to evaluate the routing move, the actual cost change of each move is calculated using the metric $C(i, u, j)$ defined in the previous section 2.1.3. During the evaluation process, as soon as a move with better evaluation value is found, it is performed, and applying an insert move to this customer is declared tabu for a number of iterations. Insert moves are terminated after a $max$-route number of iterations are performed without improvement over the best solution found for the current depot configuration. This is followed by a set of swap move.

**Swap move:** This operation swaps the position of any two customers that are currently assigned to a depot that with customer assignment change. Two customers, which are in the same route or in two different routes belonging to the same depot or two different depots, are exchanged. After the swap move is done, swapping these two customers is declared tabu for a number of iterations. Swap moves are also terminated after a $max$-route number of routing iterations are performed without improvement. The solution obtained at routing phase is then input to location phase module described in section 2.2.1.

In the above four move operation, the time window and capacity restrictions of neighbor solutions and the tabu list must be checked before accepting any move. As soon as the check is passed, feasible solutions generated by the above procedure are then recorded in the tabu list. The process is repeated until convergence or the stopping criterion is met. In neighborhood operation, every candidate solution is evaluated, and is assigned a fitness value which drives the search process. The fitness value of the HFLRPSpDO can be easily be assigned to an individual solution according to its objective function evaluation.

2.3. **Tabu search and simulated annealing.** In this section, the parameters of the proposed heuristics algorithms will be discussed, and the main scheme of the algorithms are presented in Figure 1.

2.3.1. **Tabu Search.** The general tabu search design, which has been effectively applied to single-objective optimization problem, is adopted to constitute search mechanism of the TS-heuristics for solving the HFLRPSpDO. Four tabu lists are maintained with respect to four neighborhoods in location and routing phases. As for the tabu durations to avoid cycling, there were respectively generated uniformly from intervals [5, 10] and [10, 15] for the location and routing attributes. When the move generates a solution with fitness evaluation value better than the best known one, an aspiration criterion is set to revoke a tabu. For reducing the exploration of many unfavorable configurations, threshold parameters are applied with respect to $max$-swap and $max$-add in location phase, and $max$-route in routing phase. So the neighborhood search is terminated when given number of moves in location or routing phase are performed without any improvement over the best fitness evaluation.
value. The algorithm terminates if the best found during the search process is not improved in a predetermined number of successive implementations of the routing phase.

2.3.2. **Simulated annealing.** Simulated annealing (SA) is a local search mechanism which is capable of exploring the solution space stochastically and effectively. It tries to escape from local optimum by accepting worse solution during its search with probability which is monotonically decreasing by temperature. The design of SA is similar to TS, apart from the acceptance of new solutions based on the temperature parameter. In this paper, the general SA design can also be adopted.
to constitute search mechanism of SA-heuristics for solving the HFLRPSPDO. The cooling schedule is adopted from Lundy and Mees [18], where the temperature sequence of length $K$ is recursively derived from the following:

$$T_{i+1} = \frac{T_i}{1 + \beta T_i}, \quad i = 1, \ldots, K - 1 \quad (33)$$

where $\beta$ is the cooling rate parameter expressed in terms of the input parameter values $T_1, T_K$ and $K$. The performance of SA-heuristics will be tested using various combinations of parameters: the cooling rates in location and routing phase; initial temperature and final temperature in location and routing phase; maximum number of cycles with no improvement ($\text{max-swap}$ and $\text{max-add}$ in location phase; $\text{max-route}$ in routing phase).

3. Computational results. This section presents the generation of the simulated instances used in test, and specifies the associated experimental designs, which are organized into two stages. The first stage investigates the performance of the formulations in terms of valid inequalities and relaxation schemes on the small-size problems, and evaluates the effect of vehicle capacity violation on the results. The second stage investigates the performance of the proposed TS-heuristics and SA-heuristics on large-size problems.

3.1. Simulated instances. Numerical experiments are conducted on a test set of 22 simulated instances corresponding to the different number of potential depots and customers. These test instances are derived from the LRP test set proposed by Prodhon [10, 23]. Test instances can be divided into two classes: small-size instances where $N_C$ in (15, 20, 30, 40, 50) and $N_0$ in (3, 6), large-size problems where in (50, 80, 100, 120, 150, 200) and $N_0$ in (8, 9, 10). The detailed settings for each test instance are as follows: The geographical scenarios of instances are generated by the CON method proposed by Dethloff [4, 6, 30]. For the sake of reflecting the practical customer distribution in which larger populations of customers are concentrated into a certain region, first the coordinates for half of the customer population are generated in [0, 100] according to a uniform distribution; then the coordinates of the other half of the customer population and candidate depots are uniformly distributed within $[100/3, 200/3]$. Based on the customer distribution above, the distance matrix of each instance can be obtained by calculate the Euclidean distances of the nodes. In order to generate the delivery and pickup demands of the customers in each test instance, the demand separation approach, introduced by Salhi and Nagy [25], can be used to generate delivery and pickup demands of the customers. The original demand $\text{Dem}_i$ of customer $i$ ($i \in N_C$) is stochastically generated from the interval [20, 40]. Then the delivery demand $d_i$ and pickup demand $p_j$ are evaluated as $r_i \cdot \text{Dem}_i$ and $(1 - r_i) \cdot \text{Dem}_i$, respectively. The ratio $r_i$ is calculated as $\{(x_i/y_i), (y_i/x_i)\}$, where $x_i$ and $y_i$ denote the coordinate of the customer $i$. The depot capacities have been determined in such as way that at least two or three depots are opened. The lower bound $\text{ET}_i$ of time window are uniformly distributed over the interval [0, 10], and the upper bounds $\text{LT}_i$ of time window are computed by using a random number $\sigma_i$ that is uniformly distributed over the interval [0, 1]; $\text{LT}_i = (1 + \sigma_i) \cdot \text{ET}_i$. The specified service time $\text{ST}_i$ of each customer is a random number that is uniformly distributed over the interval [0, 10]. The unit penalty parameter $\lambda_k = 0.1$. Three types of vehicles with different characteristics can be used in the problem. In order to further constraint the capacity.
violation area, we employ a factoring metric $\alpha_k$ that scales the capacity of each vehicle, and up to satisfy the $FV_{k+1} \geq (1 + \alpha_k) FV_k$, $\alpha_k = 0.2$. The other problem parameters are shown in Table 1. As a result, we generate 50 small-size and 60 large-size test instances for the HFLRPSPDO considering five instances for each problem. Note that the results are computed for each instance from the mean value of five repetitional computations with different parameters.

The stage-of-the-art LP/MIP solver CPLEX (version 10.1) was used to solve the formulations. A running time of 1200 CPU seconds is allowed on each instance. The proposed heuristic algorithms were coded using Visual C++ programming language, and stochastic simulations are performed on a PC with Inter 2.6GHz processor and 8GB RAM). For each heuristics, initial experiments were run to find the best algorithmic parameter values. Based on preliminary experiments, we implement the following combinations of parameter values as shown in Table 2.

### 3.2. Performance of the formulations in small-size problems

Table 3 summarizes the computational results of the original formulation and the strong formulation with inequalities of HFLRPSPDO. The first two columns in the table show the number of customer $N_C$ and the number of depot $N_0$, respectively. In the next three columns of the table show the result of the original problem formulation, and successive three columns report the result of the strong problem formulation with two valid inequalities. The following performance measures are used: the lower bound percentage gap, CPU time in seconds and the number of optimal solutions (# OP) obtained within 2h of CPU time. The lower bound percentage gap is the gap between the average lower bound (LB) and the average upper bound (UB) when the corresponding strong or original formulation is solved by CPLEX within 2h of CPU time. In order to obtain the strong lower bound of the computational results,
two valid inequalities on the original formulation can be adopted to strengthen the linear relaxations of the original formulation. The first valid inequality ensures that any feasible route does not contain a subtour with only two customers.

\[ x_{ijk} + x_{jik} \leq 1 \quad \forall i, j \in N_C \quad (34) \]

The second valid inequality imposes that customer \( i \) is not assigned to the depot \( j \), which is closed.

\[ z_{ij} \leq y_j \quad \forall i \in N_C, \forall j \in N_0 \quad (35) \]

Observing the results in table 3, we can conclude that the valid inequalities are very useful to obtain tight bounds for the formulations. While the LB percentage gaps are 38.06% on average for the original problem formulation, the inclusion of the valid inequalities in the corresponding formulation improves the lower bounds and the percentage gaps reduce to 8.93% on average. The strong problem formulation provides better lower and upper bounds than those of the original problem formulation (the improvement of the LB gap is around 58.14% on average). Concerning the CPU time, we can observe that appending two inequalities to the original formulation does not affect the CPU times so much in solving the LP relaxations of the formulations. The strong problem formulation spends 12.18% less CPU time than the original problem formulation to solve all the instances. The original and strong formulations are further examined in terms of their ability to solve the HFLRP-SPDO to optimality on small-size problems. From Table 3, we can also observe that the original formulation produces optimal solution for 42 out of 100 instances, but the strong problem formulation produces optimal solution for 48 out of 100 instances in a given computational time limit. To conclude, the strong problem formulation can be used to obtain optimal or near optimal solutions for the small-size problems.

As the HFLRPSPDO is an NP-hard problem, the original and strong problem formulations are not directly applicable to find optimal solutions for the large-size problems. In this section, the best LP relaxation bound of the formulations can be chosen as lower bound to evaluate the performance of our heuristic approach for
the HFLRPSPD. In order to demonstrate the improving effects of forcing the location variables to be integer, and the effect of inclusion of valid inequalities in the formulations on large-size problems, the following LP relaxations of formulations are considered in the comparison: LP relaxations of the original and strong formulations and, also semi-LP (SLP) relaxations of the strong formulations in which decision variables related with location are kept as binary while remaining decision variables are relaxed. Table 4 summarizes the computational results of LP relaxed formulations. The first two columns of table 4 are the same as in the table 3, and every successive two columns of the table report the lower bound percentage gap and CPU tie for the corresponding relaxation types of each formulation.

As seen in table 4, the strong formulation produces better LP lower bound than those of the original formulation in small computation times. While the strong formulation improves lower bound 65% on average, their CPU times in solving the LP relaxations are only 28% more than those of the original formulation. This result also confirms that appending valid inequalities to the formulations improves the LP bounds at expense of slightly increased computational time. In addition, the SLP relaxation of the strong formulation leads to better lower bounds than those of the LP relaxation of the strong formulation. The percentage gaps reduce to 5.47% on average, such that their improvements according to the LP relaxations of the strong formulation are around 51% on average. As it is expected, imposing binary variables related with the location decision to strong formulation increases the computation times of the LP relaxations for both formulations. The results in table 4 also reveal the CPU times of the LP and SLP relaxations for the formulations increase with the number of customers and depots in the problem, whereas the changes of LB percentage gap values in these formulation are not obvious. From the results in table 4, it can be conclude that the SLP relaxation of the strong formulations produce better lower bounds than those of the LP relaxations of the strong formulation at the expense of the increase of computational time.

3.3. The analysis of the effect of capacity overloads. To illustrate the idea behind the capacity violations of vehicles, we can further analyze the effect of vehicle

### Table 4. Computational results for relaxations of the formulations

| N<sub>C</sub> | N<sub>0</sub> | LP of original formulation | SLP of strong formulation |
|----------------|----------------|---------------------------|---------------------------|
|                |                | Gap% | CPU | #OP | Gap% | CPU | #OP |
| 15             | 3              | 45.82 | 0.03 | 18.63 | 0.02 | 4.75 | 0.43 |
| 20             | 3              | 36.02 | 1.84 | 1.76 | 1.59 | 2.04 | 15.71 |
| 20             | 4              | 51.80 | 2.53 | 1.09 | 2.37 | 1.84 | 1.68 |
| 30             | 4              | 26.39 | 6.85 | 16.39 | 5.89 | 4.73 | 184.93 |
| 40             | 4              | 38.74 | 6.52 | 2.54 | 7.41 | 5.91 | 347.55 |
| 30             | 5              | 33.90 | 12.48 | 11.64 | 11.91 | 2.43 | 194.69 |
| 40             | 5              | 46.29 | 11.53 | 10.83 | 10.53 | 3.74 | 842.68 |
| 50             | 5              | 34.62 | 14.83 | 18.46 | 13.96 | 6.47 | 1043.79 |
| 40             | 6              | 29.43 | 17.35 | 13.58 | 15.37 | 10.25 | 357.73 |
| 50             | 6              | 37.58 | 28.59 | 17.36 | 22.54 | 12.54 | 1074.51 |
| **Average**    |                | 38.06 | 10.26 | 11.23 | 13.13 | 5.47 | 406.37 |
Table 5. Perform comparison of HFLRPSPDO on the effect of capacity violation

| $N_C$ | $N_0$ | Cost without violation | Cost with violation | Improvement on cost (%) | CPU times (sec) | Capacity violation (%) |
|-------|-------|------------------------|---------------------|------------------------|----------------|------------------------|
| 15    | 3     | 29222                  | 24731               | 15.37                  | 86.29          | 6.82                   |
| 20    | 4     | 30325                  | 27387               | 9.69                   | 1973.64        | 9.65                   |
| 30    | 4     | 51172                  | 48373               | 5.47                   | 3729.65        | 4.27                   |
| 40    | 4     | 66101                  | 61382               | 7.14                   | 5183.59        | 6.49                   |
| 30    | 5     | 51798                  | 46183               | 10.84                  | 4734.64        | 3.85                   |
| 40    | 5     | 62976                  | 53284               | 15.39                  | 5374.68        | 6.58                   |
| 50    | 5     | 64828                  | 59337               | 8.47                   | 6739.52        | 9.83                   |
| 40    | 6     | 59014                  | 55833               | 5.39                   | 7200.00        | 5.48                   |
| 50    | 6     | 64188                  | 57821               | 9.92                   | 7200.00        | 6.29                   |
|      |       | **Average**            |                     |                        | **51225**      | **6.66**              |

overloads in the computation results of table 5. Specifically, table 5 presents the cost without capacity violation and the cost with capacity violation, the percentage improvement of total cost due to the capacity violation and the CPU times. Table 5 also provides additional data concerning the average capacity violations of routes. The average cost without violation is 51225, whereas the one with violation is 46407. The implement of average capacity violation 6.66% can improves cost 9.65% on average. Observing the results in table 5, it is evident that small capacity violations of vehicles will result in significant cost decreases. Thus, the controlled capacity violations of the vehicles results in reduced total cost for routing and scheduling problems. This is exactly the goal of the proposed formulation of HFLRPSPDO and the fact that is can be demonstrated in most simulated instances. Note that for all instances reported in table 5, the maximum violation is less than 10% of the nominal capacity in all cases, while the parameter of our algorithm was set $\alpha_k = 0.2$.

3.4. Performance of the heuristic algorithms. Table 6 summarizes the computational results of the heuristic algorithms on small-size problems. The first two columns of the table are the same as in the previous tables. In the following columns, six statistics for the TS-heuristics and SA-heuristics are presented: the average percentage gap, the average cost and the average CPU time over all instances for each test problem. Both heuristic algorithms are run ten times with different random number seeds for each instance and the best of ten runs for each instance is considered as the heuristic solution of the corresponding heuristic algorithm. It should be noted that the average percentage gaps of the heuristic solutions are calculated by considering optimal solutions, or if unknown, the best lower bounds, which are obtained by solving the formulation with valid inequalities for a maximum of $2h$.

The average percentage gaps obtained by the TS-heuristics and SA-heuristics are 0.66% and 0.75% respectively, and the maximum percentage gaps for the TS-heuristics and SA-heuristics are 1.93% and 1.82% respectively. This result indicates the capability of the two heuristic algorithms in searching solution space to reach good quality solution. The results in table 6 reveal that the TS-heuristics performs slightly better than the SA-heuristics in terms of the solution quality, i.e. the average cost for TS-heuristics is 46407, whereas the average cost for SA-heuristics is
Table 6. Computational results of the TS and SA on small-size problems

| $N_C$ | $N_0$ | TS-heuristics | SA-heuristics |
|-------|-------|---------------|---------------|
|       |       | Gap% | CPU  | #OP | Gap% | CPU  | #OP |
| 15    | 3     | 0.00 | 24731| 38.13| 0.00 | 24731| 40.57|
| 20    | 3     | 0.00 | 29742| 63.59| 0.00 | 29742| 54.13|
| 20    | 4     | 0.01 | 27387| 73.82| 0.02 | 27390| 80.53|
| 30    | 4     | 0.23 | 48373| 102.43| 0.28 | 48397| 138.62|
| 40    | 4     | 0.91 | 61382| 162.57| 0.73 | 61273| 147.76|
| 30    | 5     | 0.35 | 46183| 90.37| 0.34 | 46178| 128.54|
| 40    | 5     | 0.97 | 53284| 194.63| 1.14 | 53307| 251.72|
| 50    | 5     | 1.24 | 59337| 288.79| 1.45 | 59460| 300.05|
| 40    | 6     | 0.99 | 55833| 239.40| 1.06 | 55872| 207.24|
| 50    | 6     | 1.93 | 57821| 247.56| 1.82 | 57759| 277.42|
|       |       | Average| 0.66 | 46407| 150.13| 0.75 | 46411| 162.66|

Table 7. Computational results of the TS and SA on larger-size problems

| $N_C$ | $N_0$ | TS-heuristics | SA-heuristics |
|-------|-------|---------------|---------------|
|       |       | Gap% | CPU  | #OP | Gap% | CPU  | #OP |
| 50    | 8     | 2.41 | 54823| 234.65| 2.08 | 54711| 208.40|
| 80    | 8     | 1.73 | 113897| 383.59| 2.36 | 114602| 361.47|
| 100   | 8     | 0.96 | 137254| 437.42| 1.53 | 138029| 472.43|
| 80    | 9     | 1.24 | 100286| 369.38| 2.37 | 101405| 390.22|
| 100   | 9     | 0.72 | 130287| 482.36| 1.24 | 130960| 538.52|
| 120   | 9     | 0.92 | 157239| 501.36| 0.83 | 157099| 409.25|
| 150   | 9     | 1.09 | 186275| 472.17| 0.95 | 186117| 463.47|
| 80    | 10    | 3.73 | 983673| 302.54| 2.54 | 982388| 378.49|
| 100   | 10    | 2.27 | 125362| 261.52| 2.36 | 125472| 330.52|
| 120   | 10    | 1.34 | 139927| 289.55| 2.03 | 140080| 375.38|
| 150   | 10    | 1.46 | 173845| 573.82| 3.41 | 174186| 593.54|
| 200   | 10    | 2.03 | 237419| 479.53| 3.56 | 238279| 636.39|
|       |       | Average| 1.66 | 211690| 398.99| 2.11 | 211944| 429.84|

Concerning the CPU time, the average CPU times required by two heuristics are about 150s and 162s respectively. The results obtained indicate the proposed heuristic algorithms obtain optimal or near optimal solution by consuming much less CPU time than exact algorithm.

Finally, we investigate the performance of the proposed heuristic algorithms on large-size problem in Table 7. Each column of the table is same as in the table 6. The average percentage gaps of the heuristic solution are calculated by considering optimal solution, or if unknown, the best lower bounds, which are obtained by solving the SLP formulation with valid inequalities for a maximum of 2h. From table 7, it is seen that the average percentage gap for the TS-heuristics is 1.66%, 46411.
while the value increases to 2.11% for the SA-heuristics. The TS-heuristics performs slightly better than the SA-heuristics in terms of the solution quality. It is found that when the number of candidate depots increases, the total system cost for heuristic solution decrease as to given number of customers. Concerning the computational burden, although the CPU times of two heuristic algorithms increases when the problem size (number of candidate depots and number of customers) increases, the computational times grow slowly. The average CPU times required by two heuristic algorithms are about 399s and 430s respectively. These results reveal that both heuristic algorithms can yield the heuristic solutions, which are close to the optimal solution of the large-size problem, at the similar computational times. From table 6 and 7, we conclude that the proposed heuristic algorithms are computationally efficient algorithm for solving the HFLRPSPDO.

4. Conclusion and future directions. In this paper, we addressed for the HFLRPSPDO when the fleet is heterogeneous, i.e., comprises vehicles of different capacities and associated costs. The overloads of vehicles are allowed up to a pre-specified bound in the problem, at a penalty embedded in the objective function. A polynomial-size mixed integer linear programming formulation is proposed for the problem. Because of the NP-hardness of the HFLRPSPDO, we have proposed two heuristic algorithms based on tabu search and simulated annealing to solve the large-size problems. The computational results on instances derived from the literature show that inclusion of valid inequalities in the original formulation can achieve competitive lower bounds than those of the original formulation on small-size problems. As to the effect of capacity violation of vehicle, the computational results on simulated problem instances showed that the overloads of vehicles will result in significantly cost decreases. Computational results over large-size problem also reveal that the proposed heuristic algorithms are computationally efficient approach to solve the HFLRPSPDO.

However, there is still much opportunity to extend out work in some respects. A valuable avenue for future research is to develop new valid inequalities for the problem in order to strengthen the lower bounds of the formulation. Although SA-heuristics and TS-heuristics have shown their outstanding ability to solve the problem at hand, there is a possibility to use other heuristics or metaheuristics to solve the same problem or to conduct an empirical study to compare the strengths of various approaches in solving the problem of this paper. Finally, there remains the need to improve further the proposed heuristics in this paper to solve larger problem instances and different variants of the problem.

Acknowledgments. The authors are indebted to the editor and anonymous referees for their helpful comments and suggestions. This work was supported by 2017 Humanities and Social Science Foundation of Chinese Ministry of Education [grant number: 17YJA630100]; 2018 National Natural Science Foundation of China [grant number: 71862016]; 2016 College Humanities and Social Science Foundation of Jiangxi Province [grant number: GL161035].

REFERENCES

[1] D. Ambrosinoa, Distribution network design: New problems and related models, European Journal of Operational Research, 165 (2005), 610–624.
[2] R. T. Berger, C. R. Coullard and M. S. Daskin, Location-routing problems with distance constraints, Transportation Science, 41 (2007), 29–43.
[3] T. W. Chien, Heuristic procedures for practical-sized uncapacitated location-capacitated routing problems, *Decision Sciences*, 24 (1993), 995–1021.

[4] C. H. Chu and J. Hopscotch, Further discussion for transit system of chicago, *Journal of Discrete Mathematical Sciences & Cryptography*, 20 (2017), 717–724.

[5] G. Clarke and J. W. Wright, Scheduling of vehicles from a central depot to a number of delivery points, *Operations Research*, 12 (1964), 568–581.

[6] J. Dethloff, Vehicle routing and reverse logistics: The vehicle routing problem with simultaneous delivery and pick-up, *OR-Spektrum*, 23 (2001), 79–96.

[7] J. Geunes and B. M. Chang, *Operations Research Models for Supply Chain Management and Design*, vol. 76, Springer US, 1994.

[8] B. Golden, A. Assad, L. Levy and F. Gheysens, The fleet size and mix vehicle routing problem, *Computers & Operations Research*, 11 (1984), 49–66.

[9] G. Ioannou, M. Kritikos and G. Prastacos, A greedy look-ahead heuristic for the vehicle routing problem with time windows, *Journal of the Operational Research Society*, 52 (2001), 523–537.

[10] H. Kamankesh and V. G. Agelidis, A sufficient stochastic framework for optimal operation of micro-grids considering high penetration of renewable energy sources and electric vehicles, *Journal of Intelligent & Fuzzy Systems*, 32 (2017), 373–387.

[11] I. Karaoglan, F. Altiparmak, I. Kara and B. Dengiz, A branch and cut algorithm for the location-routing problem with simultaneous pickup and delivery, *European Journal of Operational Research*, 211 (2011), 318–332.

[12] I. Karaoglan, F. Altiparmak, I. Kara and B. Dengiz, The location-routing problem with simultaneous pickup and delivery: Formulations and a heuristic approach, *Omega*, 40 (2012), 465–477.

[13] M. N. Kritikos and G. Ioannou, The heterogeneous fleet vehicle routing problem with overloads and time windows, *International Journal of Production Economics*, 144 (2013), 68–75.

[14] G. Laporte, Y. Nobert and D. Arpin, An exact algorithm for solving a capacitated location-routing problem, *Annals of Operations Research*, 6 (1986), 293–310.

[15] G. Laporte, Y. Nobert and S. Taillefer, Solving a family of multi-depot vehicle routing and location-routing problems, *Transportation Science*, 22 (1988), 161–172.

[16] C. K. Y. Lin, C. K. Chow and A. Chen, A location-routing-loading problem for bill delivery services, *Computers & Industrial Engineering*, 43 (2002), 5–25.

[17] C. K. Y. Lin and R. C. W. Kwok, Multi-objective metaheuristics for a location-routing problem with multiple use of vehicles on real data and simulated data, *European Journal of Operational Research*, 175 (2006), 1833–1849.

[18] M. Lundy and A. Mees, Convergence of an annealing algorithm, *Mathematical Programming*, 34 (1986), 111–124.

[19] H. Min, Consolidation terminal location-allocation and consolidated routing problems, *Journal of Business Logistics*, 17 (1996), 235–263.

[20] H. Min, V. Jayaraman and R. Srivastava, *Combined Location-Routing Problems: A Synthesis and Future Research Directions*, vol. 108, Springer Berlin Heidelberg, 1998.

[21] G. Nagy and S. Salhi, Location-routing: Issues, models and methods, *European Journal of Operational Research*, 177 (2007), 649–672.

[22] G. Nagy and S. Salhi, Nested heuristic methods for the location-routing problem, 47 (1996), 1166–1174.

[23] C. Prodhon, . in *http://prodhonc.free.fr/homepage*, 2016.

[24] S. Salhi and M. Fraser, An integrated heuristic approach for the combined location vehicle fleet mix problem, *Studies in Locational Analysis*, 8 (1996), 3–21.

[25] S. Salhi and G. Nagy, A cluster insertion heuristic for single and multiple depot vehicle routing problems with backhauling, *Journal of the Operational Research Society*, 50 (1999), 1034–1042.

[26] M. Schwarzdt and J. Dethloff, Solving a continuous location-routing problem by use of a self-organizing map, *International Journal of Physical Distribution & Logistics Management*, 35 (2005), 390–408.

[27] R. Srivastava, Alternate solution procedures for the location-routing problem, *Omega*, 21 (1993), 497–506.

[28] D. Tuzun and L. I. Burke, A two-phase tabu search approach to the location routing problem, *European Journal of Operational Research*, 116 (1999), 87–99.
[29] T. H. Wu, C. Low and J. W. Bai, Heuristic solutions to multi-depot location-routing problems, *Computers & Operations Research*, **29** (2002), 1393–1415.

[30] X. Zhang, Z. B. Zhang, H. Broersma and X. Wen, On the complexity of edge-colored subgraph partitioning problems in network optimization, *Discrete Mathematics & Theoretical Computer Science Dmtcs*, **17** (2016), 227–244.

Received June 2017; revised December 2017.

*E-mail address: wxfyf0163.com*