Applied Soft Computing xxx (2016) xxx–xxx

Contents lists available at ScienceDirect

Applied Soft Computing

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

A label based ant colony algorithm for heterogeneous vehicle routing with mixed backhaul

Q1 Weiqin Wu[a,b], Yu Tian[a,c,*], Tongdan Jin[d]

[a] School of Business, Sun Yat-sen University, Guangzhou 510275, China
[b] China Post Group Corporation, Ltd., Guangdong Branch, Guangzhou 510898, China
[c] School of Business, Jishou University, Jishou 41600, China
[d] Ingram School of Engineering, Texas State University, San Marcos, TX 78666, USA

ARTICLE INFO

Article history:
Received 20 August 2015
Received in revised form 12 March 2016
Accepted 9 May 2016
Available online xxx

Keywords:
Ant colony system
Label matching
Heterogeneous vehicle
Mixed delivery and pickup
Multi-criteria decision

ABSTRACT

Vehicle heterogeneity and backhaul mixed-load problems are often studied separately in existing literature. This paper aims to solve a type of vehicle routing problem by simultaneously considering fleet heterogeneity, backhaul mixed-loads, and time windows. The goal is to determine the vehicle types, the fleet size, and the travel routes such that the total service cost is minimized. We propose a multi-attribute Label-based Ant Colony System (LACS) algorithm to tackle this complex optimization problem. The multi-attribute labeling technique enables us to characterize the customer demand, the vehicle states, and the route options. The features of the ant colony system include swarm intelligence and searching robustness. A variety of benchmark instances are used to demonstrate the computational advantage and the global optimality of the LACS algorithm. We also implemented the proposed algorithm in a real-world environment by solving an 84-node postal shuttle service problem for China Post Office in Guangzhou. The results show that a heterogeneous fleet is preferred to a homogenous fleet as it generates more cost savings under variable customer demands.

1. Introduction

Q3 When we address a vehicle routing problem (VRP), apart from time windows and vehicle capacity, other operational variables, such as vehicle types and backhauls should be taken into account as well. Backhauls mean a vehicle can carry out pick-up tasks while delivering the goods (i.e., linehauls). Vidal et al. [41,42] treat the vehicle routing problem with multiple constraints as a multi-attribute vehicle routing problem. Though more realistic, the accommodation of these operation constraints makes the vehicle routing problem more complex and difficult to solve. In this paper, we focus on a type of vehicle routing problem considering backhauls, mixed-load, and time windows served by a fleet of heterogeneous vehicles. Our study is motivated by the vehicle routing problems arising from mail delivery and pick-up service business. Postal shuttles usually deliver outbound mails from the post offices to the mail processing center and also take inbound mails back to different delivery stations. Both linehauls and backhauls can be served in a random order. They have their own time windows, and different types of vehicles are often used in order to meet the requirements of various customers. Q4

In general, vehicle routing problems with backhauls (VRPB) can be classified into three categories [33]: (1) linehauls must be served before backhauls; (2) linehauls and backhauls can be served in a random order; and (3) simultaneous pick-up and delivery. The first category problem is often named as vehicle routing problem with delivery before pick-up (VRPDBP) because pick-up tasks cannot be executed prior to the completion of all delivery jobs [1,40,16,34]. The second category captures the situation that pick-up tasks can be inserted into the routes constructed by delivery, and it is also referred to as vehicle routing problem with backhauls and mixed-load (VRPBM). Some researchers also call it as vehicle routing problem with mixed pickup and delivery (see Refs. [33,20,8]). Simultaneous pick-up and delivery means a customer requests both pick-up and delivery services at the same time. This situation can be called the vehicle routing problem with simultaneous pick-up and delivery (VRPSPD) (see Refs. [31,7,21,37,18,29]). In early literature, the first category problems attracted more attention than the other two categories. This is because the vehicle compartment at that time can only be loaded from the rear door, making the "mixed" service too costly to perform. Driven by the market com-

Q2 * Corresponding author at: School of Business, Sun Yat-sen University, Guangzhou 510275, China.
E-mail address: mnsty@mail.sysu.edu.cn (Y. Tian).

http://dx.doi.org/10.1016/j.asoc.2016.05.011
1568-4946/© 2016 Elsevier B.V. All rights reserved.

petition, today's vehicle compartment can be loaded from multiple sides. Hence, both the pick-up and the delivery tasks can be carried out independently. As such, the stream of research on VRPBM and VRPSPD is growing quickly in recent years. Although VRPBM can be treated as a special case of VRPSPD [31,15], it still possesses its own modeling characteristics and unique solution techniques. Take the mail service shuttle as an example. The shuttle delivers outbound mails to the central processing center, and then brings the inbound mails back to the post offices. This is a typical VRPBM problem which is quite difficult to solve with the conventional VRPSPD framework.

Various optimization algorithms have been proposed to solve the VRPBM problem. Golden et al. [20] proposed an insertion algorithm with which both the linehaul and the backhaul customers can be served in an arbitrary order. Casco et al. [6] used the Clark–Wright algorithm to construct a linehaul route, and applied the insertion technique to generate the entire path. Salhi and Nagy [33] developed a different insert algorithm which is tested on various data sets consisting of single and multiple depots. They showed that their method does not increase additional computational cost compared to other insertion techniques. Chen and Wu [7] proposed a simulated annealing algorithm to solve a modified vehicle routing problem with backhaul services. The algorithm is further verified on the Solomon benchmark data sets [36], and a lower transportation cost is claimed to be found. Cheung and Hang [8] formulated the VRPBM problem under a label matching framework where a label contains multiple attributes each representing the vehicle and the route state. Two heuristic algorithms, i.e., simultaneous assignment and sequential assignment, are used to search for the optimal routes. They showed that the label matching algorithm yields a better solution with a shorter computational time. Later, an adaptive labeling algorithm was further proposed by Cheung et al. [9] to solve a cross-border drayage container transportation problem. Küçükoğlu and Öztürk [24] constructed a hybrid meta-heuristic algorithm that integrates simulated annealing and Tabu search to obtain more effective solutions for the VRPBM models with time windows. Brito et al. [5] and Reed et al. [32] solved the capacitated vehicle routing problems using ant colony algorithm. In the former study, the vehicle capacity and time windows were modeled as fuzzy constraints due to the uncertain customer demands and imprecise information.

In recent years, the heterogeneous fleet vehicle routing problem (HFVRP) is gaining popularity due to the increased service customization in practice. Taillard [38] presented a three-step method to solve a multi-capacity vehicle routing problem based on column generation method. First, vehicles are grouped according to the capacity or type. Then, the initial solution is obtained by solving the routing problem for each vehicle type. Finally, the optimal solution is generated by aggregating all the solutions. Li et al. [26] developed a variant of record-to-record travel algorithm for the standard vehicle routing problem by taking into account the heterogeneity feature of the fleet, and further reported the computational results from eight benchmark problems. Brandão [4] divided different types of vehicles into two groups: fixed cost and variable cost fleets. The objective is to minimize the total fleet transportation costs subject to one-time customer visit and other operating requirements. A Tabu algorithm was developed to solve the Taillard examples for the model verification, and a satisfactory result was claimed to be achieved.

Remarkable achievements have been made in the field of vehicle routing. Laporte [25] summarized the developments of the solution approaches for VRP in the last five decades. Eksioglu et al. [14] provided a taxonomic review on the vehicle routing optimization problems. Albeit a large body of literature on VRP and its variants, the backhaul mixed-loads and the heterogeneous fleet assignment problems are often studied separately. Only recently some researchers started to formulate and solve the vehicle routing problem by jointly considering fleet heterogeneity, backhauls, mixed-load, and time windows (VRPHBMTW). For instance, Belmecheri et al. [2] formulated a single-objective VRPHBMTW model to minimize the travel distance, and a particle swarm optimization algorithm was implemented to search for the optimal solution.

This paper aims to solve a type of VRPHBMTW problem arising from mail delivery and pick-up service industry. Our problem setting is similar to Belmecheri et al. [2] in that both consider the fleet heterogeneity and mixed backhauling tasks. The main difference is that we formulate VRPHBMTW under a bi-objective optimization framework where both the number and type of transport vehicles and the aggregate travel cost are jointly minimized. In addition, our model also imposes the penalty costs on extended customer waiting time when vehicles arrive either early or late. To solve the problem, we combined the two objectives as a single objective optimization using weighted sum. A two-stage optimization algorithm is then proposed to search for the optimality. Namely, we first determine the vehicle type and minimize the vehicle quantity, and then we optimize the traveling routes based on the results from the previous stage. To tackle this complex optimization problem, we develop a Label-based Ant Colony System (LACS) algorithm to jointly minimize the travel cost and the vehicle quantity and type. The LACS algorithm synthesizes the multi-attribute labeling (MAL) technique [8] with the multi-ant colony system (ACS) algorithm [17]. Hence, it possesses the fast, flexible, and accessible feature of MAL as well as the strong robustness and global search ability of ACS. In particular, the LACS algorithm enables us to resolve complex operational constraints, such as fleet heterogeneity, time windows, and multiple periods that are encountered in practical vehicle routing decisions.

Following the introduction, the rest of this paper is organized as follows. Section 2 presents the mathematical formulation of the VRPHBMTW problem. Section 3 provides an overview of the LACS framework that comprises the label matching technique and the ant colony system. Section 4 describes the detailed implementation procedure of the LACS algorithm. In Section 5, the performance of the proposed algorithm is analyzed and compared using Solomon benchmark data set and an 84-node shuttle routing problem from China Post. Section 6 concludes the paper.

2. Problem formulation

2.1. Problem description and assumptions

To simplify our presentation, we use the word "demand" to represent a delivery or a pick-up request from a customer. Without loss of generality, we also use the phrase "at a demand" to represent "at the location of a customer demand".

In VRPHBMTW problem, $D = \left\{1, 2, \ldots, |D|\right\}$ is the customer set in which a customer must be served by a unique depot. Each customer requires q_i amounts of goods for $i = 1, 2, \ldots, n$. A positive q_i means a delivery request while a negative value indicates a pick-up task. In addition, $K = \left\{1, 2\ldots, |K|\right\}$ vehicles with capacity C_k for $k = 1, 2, \ldots, |K|$ are available to serve D customers. Each vehicle is available at its earliest time s_k for $k \in K$ and must return to the depot before the latest time e_k for $k \in K$. Meanwhile, the customer must be served within a required time windows. Here is the summary of these key assumptions:

- Given a set of vehicles K with different capacity C_k, the duty hours and available times fall in $[s_k, e_k]$. Each vehicle can make only one trip departing from the depot D_0 and must return to the depot within the duty hours.

- Each customer has a fixed service time window $TW_{d_i} \in [a_i, b_i]$, and the linehaul and the backhaul customers can be served in an arbitrary order on a route.
- Each customer can be served only by one vehicle for one time.

2.2. Notations

In our problem setting, each vehicle must finish all the tasks before returning to the depot. To accommodate this condition, we define a notation table consisting of set parameters, time parameters, cost and capacity parameters, and decision variables as follows:

(1) Set parameters

K	Set of vehicles, and $K = \{1, 2, \ldots, \lvert K\rvert\}$
D	Set of customer demands, and $D = \{1, 2, \ldots, \lvert D\rvert\}$
$\bar{D}$	Extended set of customer demands including depot D_0 where the virtual demand is zero, and $\bar{D} = D \cup \{0\}$

(2) Time parameters

a_i	Earliest start time for serving demand i, and $i \in D$
b_i	Latest start time for serving demand i, and $i \in D$
τ_i	Service time for demand i, and $i \in D$
t_{ij}	Traveling time from demand i to demand j, and $i, j \in D$
s_k	Earliest time for vehicle k to start service, and $k \in K$
e_k	Latest time for vehicle k to finish the service, and $k \in K$

(3) Cost and capacity parameters

d_{ij}	Traveling distance from demand i to demand j, and $i, j \in \bar{D}$
c_k	Traveling cost per unit distance for vehicle k
c_k^w	Waiting cost per unit time for vehicle k
q_k^F	Full capacity of vehicle k, and $k \in K$
q_i	Demand for customer i, and $i \in D$

(4) Decision variables and dependent variables

x_{ijk}	t_i Start time for servicing demand i, and $i \in D$
t_i^w	Waiting time at demand or customer site i, and $i \in D$
q_{ki}^D	Empty capacity of vehicle k when departing from site i, $i \in D, k \in K$
q_{ki}^A	Load for vehicle k when it arrives at demand site i, and $i \in D, k \in K$

2.3. Mathematical formulation

Our goal is to minimize the number of required vehicles as well as to minimize the total travel cost. Denoted as Model 1, the following multi-objective optimization problem (MOOP) is formulated to Q5 accommodate both design criteria as follows,

Model 1:

$$\text{Min} : f_1(\mathbf{x}) = \sum_{j \in D} \sum_{k \in K} x_{0jk} \tag{1}$$

$$\text{Min} : f_2(\mathbf{x}) = \sum_{i \in \bar{D}} \sum_{j \in \bar{D}} \sum_{k \in K} x_{ijk}(d_{ij} c_k + t_j^w c_k^w) \tag{2}$$

Subject to:

$$\sum_{k \in K} \sum_{j \in \bar{D}} x_{ijk} = 1 \quad \forall i \in D \tag{3}$$

$$\sum_{j \in \bar{D}} x_{0jk} = \sum_{i \in \bar{D}} x_{i0k} = 1 \quad \forall k \in K \tag{4}$$

$$\sum_{i \in \bar{D}} x_{ijk} = \sum_{l \in \bar{D}} x_{jlk} \quad \forall k \in K, \forall j \in D \tag{5}$$

$$x_{ijk} = 1 \Rightarrow t_i + t_{ij} + \tau_i \le t_j \quad \forall k \in K, \forall i, j \in D \tag{6}$$

$$x_{0jk} = 1 \Rightarrow s_k + t_{0j} \le t_j \quad \forall k \in K, \forall j \in D \tag{7}$$

$$x_{i0k} = 1 \Rightarrow t_i + \tau_i + t_{i0} \le e_k \quad \forall k \in K, \forall i \in D \tag{8}$$

$$a_i \le t_i \le b_i \quad \forall i \in D \tag{9}$$

$$x_{ijk} = 1 \Rightarrow t_j^w = \max\left\{a_j - t_i - \tau_i - t_{ij},\ 0\right\} \quad \forall j \in D \tag{10}$$

$$x_{ijk} = 1 \Rightarrow q_{ki}^D = q_{kj}^A \quad \forall k \in K, \forall i, j \in D \tag{11}$$

$$Q_{ki}^D = Q_{ki}^A - q_i \quad \forall k \in K, \forall i \in D \tag{12}$$

$$x_{ijk} = 1 \Rightarrow q_{kj}^A = q_k^F - \sum_{m \in s(i)} \max(q_m, 0) \quad \forall k \in K, \forall i, j \in D \tag{13}$$

$$0 \le q_{ki}^A \le q_k^F, \quad \forall i \in D, \forall k \in \in K \tag{14}$$

$$0 \le q_{ki}^D \le q_k^F, \quad \forall i \in D, \forall k \in \in K \tag{15}$$

$$x_{ijk} \in \{0,\ 1\} \quad \forall k \in K, \text{and} \forall i, j \in D \tag{16}$$

Note that vehicle k travels directly from demand i to j (where $i \neq j$), $x_{ijk} = 1$. Otherwise, $x_{ijk} = 0$. Object functions (1) and (2) minimize the number of service vehicles and the total traveling cost respectively. The second objective is achieved by reducing the travel distance and the vehicle waiting time. Constraint (3) ensures that each demand must be satisfied by only one vehicle at once. Constraint (4) states that each vehicle must leave from and return to the depot. This constraint also implies that vehicle k may stay at the depot with $x_{00k} = 1$ if no dispatch is required. Constraint (5) states that once vehicle k has visited a customer, it must leave the customer as well. Constraint (6) prescribes the condition that the time to serve demand j should not be earlier than the vehicle arrival time at that customer site. Constraints (7) and (8) define the available time and duty hours of vehicles respectively. Constraint (9) ensures that the time windows of all demands are satisfied. Constraint (10) calculates the waiting time for each customer. Constraint (11) states that if a vehicle travels directly from demand i to j, the load of the vehicle on departure from demand i is equal to the amount when it arrives at demand j. Constraint (12) means the available capacity of a vehicle increases with linehaul and decreases with pick-up tasks. Constraints (11) and (12) can be transformed into (13) to characterize the changing volumes of liehauls. Note that $s(i)$ is the set of customer who have been served by vehicle k. Constraints (14) and (15) define the maximum capacity of a vehicle. Finally, constraint (16) captures the status of the vehicle arriving at and leaving from the same customer.

In general, it is quite difficult, if not impossible, to directly search for the optimal solution of the MOOP model as it requires the generation of a non-dominant solution set. In general a multi-objective programming model can be transformed into a single objective problem by assigning appropriate weight to individual objective functions. If the values of these objective functions are close among each other, assignments of linear weights are preferred as it facilitates the construction of the solution algorithm [10,35]. Hence this paper adopts the linear assignment by combining two objective functions, i.e., Eqs. (1) and (2), to form a single-objective function denoted as $g(\mathbf{x})$ as follows:

Model 2:

$$g(\mathbf{x}) = \min\{p_1 f_1(\mathbf{x})\} + \min\{p_2 f_2(\mathbf{x})\}, \quad p_1, p_2 \text{ are priority factors} \tag{17}$$

Subject to:

Constraints (3)–(16), where p_1 and p_2 are priority factors or weights. The studies by Bent and Van Hentenryck [3] and Li and Chang [27] show that by adjusting the values of p_1 and p_2, we can always prioritize the minimization of vehicle quantity over the minimization of the travel distance. In other words, Model 2 can be solved using a two-stage optimization method. In stage 1, we determine the vehicle types and the number per type, and in stage 2, the total travel cost is minimized based on the vehicle quantity and types determined in previous stage.

3. Overview of labeling technique and ant colony system

In literature, two-stage heuristic algorithms consisting of initialization and optimization are often used to tackle large and complex VRP issues. In general, the performance of the algorithm determines the computational efficiency and the solution quality. Cheung and Hang [8] developed a two-stage search algorithm within the multi-labeling framework. In stage 1, they generated the initial solution under the label matching framework, and in stage 2 the initial solution was further refined via simultaneous assignment (MA) and sequential assignment (SA) algorithms respectively. In this paper, we adopt the label-based initialization framework for stage 1 decision, but use the nearest neighborhood algorithm to find the initial solution. In stage 2, we develop an ant colony system algorithm to search for the global minimization of the cost. The details of our algorithm are elaborated in the following paragraphs.

3.1. Labels and label creation

Based on the object-oriented programming concept, Powell et al. [30] proposed an adaptive labeling approach that turns mathematical constraints into label mapping rules. In their algorithm, multi-attribute labels are created for customers and vehicles through which a set of labels is dynamically updated to form a feasible route. One advantage of the labeling approach is that it can easily handle a variety of operational constraints, such as heterogeneous vehicles, different compartments, and multiple periods. When these constraints change, only small adjustments to the algorithm are needed. Before starting the label creation, we describe three types of labels: demand label, vehicle label, and route label.

3.1.1. Demand labels

The demand label is created once the vehicle completes a demand request. Let D be a set of demand labels. For each demand i for $i \in D$, we define a demand label $d \in D$ that contains the following attributes:

$$d = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \\ d_6 \end{bmatrix} = \begin{bmatrix} q_i \\ a_i \\ b_i \\ \tau_i \\ i \\ m \end{bmatrix} \left\{ \begin{array}{c} \text{Amount of demand} \\ \text{Start of time window} \\ \text{End of time window} \\ \text{Service time} \\ \text{Index of the demand} \\ \text{Requirement for the vehicle type} \end{array} \right\}$$

Because there exists a one-to-one mapping between customer i and the corresponding label (i.e., $d_5 = i$), we use label d to denote demand i for notational simplicity. Attribute d_1 is defined as the demand quantity q_i for customer i with $|q_i| < q_k^F$. The definitions of other attributes are straightforward. For instance, d_2 is the start service time, d_3 is the end service time, d_4 is the actual service time, d_5 is the demand site or the customer index, and d_6 defines what type of vehicle is available for this demand.

3.1.2. Vehicle labels

A vehicle can own multiple labels, and the values of the attributes within a label depend on when, where, and how the vehicle arrives at a customer site. Let V be a set of vehicle labels. For a vehicle label $v \in V$, its attributes are defined as:

$$v = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \\ v_8 \\ v_9 \end{bmatrix} \left\{ \begin{array}{c} \text{Available time} \\ \text{Off duty time} \\ \text{Full capacity} \\ \text{Immediate delivery goods} \\ \text{Immediate pickup capacity} \\ \text{Set of demands in the same route} \\ \text{Index of the demand where the vehicle is available} \\ \text{Identity of the vehicle} \\ \text{Type of the vehicle} \end{array} \right\}$$

Here v_1 is the vehicle available time at which the vehicle satisfies the current demand. v_4 is the immediate quantity of delivery goods. Namely, it is the maximum amount of goods that the vehicle can deliver at the next customer site if it is a linehaul request. Similarly, v_5 is the immediate pick-up capacity. It is the maximum amount of goods a vehicle can collect for the next customer under a backhaul request. v_6 is the set of demands in the same route, v_7 is the index of demand where the vehicle is available, v_8 is the identity of the vehicle, and v_9 is vehicle type. Explanations on v_2 and v_3 are ignored because they are self-explanatory.

3.1.3. Route label

When a vehicle serves a customer, a route label is automatically created. If a vehicle cannot visit any customer, a complete route is constructed. Let R be a set of route labels. A route label $r \in R$ is defined as follows:

$$r = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \end{bmatrix}$$

$$\left\{ \begin{array}{l} \text{The total amount of goods delivered on the current route} \\ \text{The total amount of goods picked-up on the current route} \\ \text{Set of demands on the current route} \\ \text{Length of the current route} \\ \text{The index of the vehicle which finishes the current route} \end{array} \right\}$$

Attributes r_1 and r_2 are defined respectively as the total amount of goods delivered and collected on the current route r_3 with the traveling distance r_4. Note that r_5 is the vehicle identification number.

3.1.4. Label creations and matching

Demand labels remain the same throughout the computation, but vehicle and route labels may change with the service path. The label creation rules are given as follows:

1) Vehicle label creation conditions

For a given vehicle label $v \in V$ and a demand label $d \in D$, let $i = v_7$ (i.e., the vehicle is currently at demand i) and $i = d_5$ (i.e., index of the demand to be covered). The vehicle with label v can cover demand d with a label if the following constraints hold.

$$v_1 + t_{ij} \le d_3 \tag{18}$$

$$\max\left\{v_1 + t_{ij}, d_2\right\} + d_4 + t_{j0} \le v_2 \tag{19}$$

$$\text{if} \quad d_1 \ge 0 \quad v_4 \ge d_1 \tag{20}$$

$$\text{if} \quad d_1 < 0, \quad v_5 \le d_1 \tag{21}$$

$$j \notin v_6, \quad \forall v \in V \tag{22}$$

All these constraints are associated with the vehicle-demand assignment feasibility condition. Constraint (18) represents the time window constraints, namely all customers must be served prior to the end of their window time. Constraint (19) ensures that the vehicle can go back to the depot prior to the off-duty time upon the completion of all required demands. Constraints (20) and (21) define the capacity requirements of linehauls and backhauls, respectively. Constraint (22) ensures that demand j must be a new demand which has not been served by any other vehicles.

2) Vehicle label creation

Considering a scenario that a vehicle just fulfilled a demand (at a demand location with vehicle label v) and is going to serve a new demand with label d. We will create a new label v' to capture the state of the vehicle when it is serving the new customer. This new label is created by mapping $F: V \times D \to V$ as follows:

$$v' = f(v, d) \begin{Bmatrix} v'_1 = \max(v_1 + t_{ij}, d_2) + d_4 \\ v'_2 = v_2 \\ v'_3 = v_3 \\ v'_4 = v_4 - \max(d_1, 0) \\ v'_5 = \min(v_3, v_5 + d_1) \\ v'_6 = v_6 \cup \left\{j\right\} \\ v'_7 = j \\ v'_8 = v_8 \\ v'_9 = v_9 \end{Bmatrix} \tag{23}$$

Eq. (23) shows that when constraint (18) is satisfied, the service starts at $\max\left\{v_1 + t_{ij}, d_2\right\}$ and the vehicle becomes available at time v'_1 which is equal to the completion time of demand d. Here $v'_1 = \max\left\{v_1 + t_{ij}, d_2\right\} + d_4$, and v'_2, v'_3, v'_8, and v'_9 are constant parameters. Two cases should be considered upon the completion of the current demand. If the previous service is a backhaul task, the immediate pick-up capacity v'_5 decreases, while the quantity of the delivery goods v'_4 remains the same. On the other hand, if the previous service is a linehaul task, the immediate quantity of delivery goods v'_4 decreases, and the pick-up quantity v'_5 remains constant. In either case, the total pick-up capacity cannot exceed the current vehicle capacity. From this matching process, we can see that a vehicle can create different vehicle labels corresponding to different customers.

3) Route label creation

Suppose a vehicle with label v cannot cover any demands, then a route label r is created using a mapping $R: V \to R$ such that

$$r = f(v) = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^{\tilde{n}} d_i.d_1, \ \text{for} d_i \in v_6, \ d_i.d_1 > 0 \\ \sum_{i=0}^{\tilde{n}} d_i.d_1, \ \text{for} d_i \in v_6, \ d_i.d_1 < 0 \\ v_6 \cup v_7 \\ \sum_{i=1}^{\tilde{n}} l_{d_{i-1}d_i} + l_{d_0 d_i}, \ \text{for} d_i \in v_6 \\ v_8 \end{bmatrix} \tag{24}$$

where $\tilde{n}$ is the number of customers a vehicle served. r_1 is the sum of demands of linehaul customers, r_2 is the sum of the demands of backhaul customers, r_3 indicates that a trip starts at the depot, and must terminate at the depot, r_4 is the total distance of the current route, and r_5 indicates which vehicle completes the current route.

3.2. Principle of ant colony system

Ant colony system (ACS) was originally introduced by Dorigo and Gambardella [12] for the purpose of improving the performance of ant system in solving traveling salesman problems. We leverage ACS algorithm to generate the optimal vehicle number and the travel distance to minimize the objective function in Eq. (17). In particular, two artificial ant colonies are used: one ant colony optimizes the vehicle number and the other minimizes the total travel distance. In this section we describe the working principle of this optimization method.

Assume that there are m ants and n nodes (i.e., customers). The distance between nodes i and j is d_{ij}. Let η_{ij} be the inverse of the distance of edge (i, j) which implies that the corresponding heuristic value, τ_{ij} is the pheromone intensity of the edge. $\Delta\tau_{ij}$ is the pheromone of ant k left on edge (i, j). Let P_{ij}^k be the probability that ant k moves to j from i. The pheromone decay is denoted as α for $0 < \alpha < 1$. Finally, β is a positive parameter characterizing the relative importance of pheromone versus distance. When ant k departs from node i, it chooses node j as the next destination with probability P_{ij}^k as follows

$$P_{ij}^k = \begin{cases} \dfrac{(\tau_{ij})^\alpha \times (\eta_{ij})^\beta}{\sum_{s \in N_i^k} (\tau_{is})^\alpha \times (\eta_{is})^\beta}, & \text{if} j \in N_i^k \\ 0 & \text{otherwise} \end{cases} \tag{25}$$

where $N_i{}^k$ is the set of the feasible nodes. Dorigo and Gambardella [12] proposed three rules for the ACS algorithm to improve the traveling salesman problem solution. We adopt and incorporate these rules into the LACS algorithm for vehicle label matching and route optimization as well.

1) State transition rule

Based on the pseudo-random-proportional rule, LACS leverages exploration and exploitation to determine the next feasible node. The state transition rule is defined based on the following criterion:

$$p = \begin{cases} \underset{s \in N_i^k}{\text{argmax}} \left\{(\tau_{is})^\alpha.(\eta_{js})^\beta\right\} & \text{If} q \le q_0, \ \text{use exploration} \\ P_{ij}^k & \text{otherwise, use exploitation} \end{cases} \tag{26}$$

The state transition rule is equivalent to the random-proportional rule of the original ant system. It shows that an ant tends to choose a short edge with more pheromone as the next tour.

2) Local updating rule

In LACS, pheromone trail is updated locally and the local updating rule is applied to the process when an ant searches for a single route. The following formula is applied to update the path pheromone upon the traversing of the ant,

$$\tau_{ij} = (1-\rho).\tau_{ij} + \rho\Delta\tau_{ij} \tag{27}$$

where ρ for $0 \leq \rho \leq 1$ is the pheromone released by the ant. In fact, $1-\rho$ is often used to represent the decay degree of pheromone. τ_0 is the initial value of trails. It has been shown that $\tau_0 = (nL_{nm})^{-1}$ is a good estimate as reported by Gambardella et al. [17]. Here L_{nm} is the length of the initial solution generated by the nearest neighbor heuristic. The local updating rule in Eq. (27) is able to decrease the pheromone of the selected edge, or increase the probability of choosing alternative paths. Hence, ants are prevented from selecting the same edge.

3) Global updating rule

After the current best solution is found, the global updating rule is applied to adjust the pheromone across the route. This updating strategy is proved to be more efficient than the one based on all the constructed solutions by Dorigo and Gambardella [12]. After an ant has constructed its route, a global update is performed based on the following rule:

$$\tau_{ij} = (1-\alpha).\tau_{ij} + \alpha\Delta\tau_{ij} \tag{28}$$

with

$$\Delta\tau(i,j) = \begin{cases} \left(L_{gb}\right)^{-1} & \text{if}(i,j) \in \text{global best solution} \\ 0 & \text{otherwise} \end{cases}$$

where L_{gb} is the current best global route. It is worth mentioning that the global updating rule is only applied in the process of constructing L_{gb}.

4. Detailed design of Label-Based Ant Colony System algorithm

Recall that LACS is a two-stage heuristic optimization method. First, it constructs an initial solution using the nearest neighbor search within the label matching framework. Second, it improves the initial solution using ant colony system algorithm to minimize the tours and the total route length. The pseudo code of the two-stage optimization is summarized below:

```
While($i <= $repeattime; $i = i + 1$)
Initialsolution();
For(int$j = 1$; $j <= $extendcycle; $j = j + 1$)
        LACS_Vehicle();
        LACS_Distance();
        Savecourrentresult();
Endfor
Savecourrentbestresult();
Endwhile
Getbestresult();
Printbestresult();
```

The flowchart of the algorithm is depicted in Fig. 1.

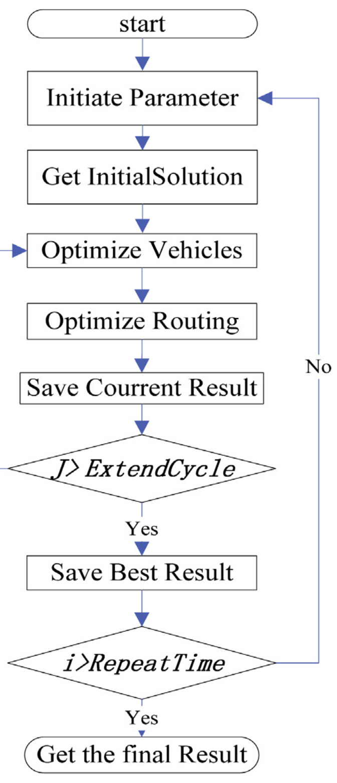

Fig. 1. The general flowchart of LACS algorithm.

4.1. Initial solution construction

The solution produced by the nearest neighbor heuristic is a good starting point to initialize the path pheromone. A 6-step procedure for constructing a solution based on the nearest neighbor heuristic algorithm is presented below.

Step 1: Create labels for all customers in D, i.e., generate a set for D.

Create labels for all vehicles in K according to their initial states and create $Q = K$.

Step 2: Let v be the first element of Q, $v = Q$; and delete v from Q, $Q = Q\backslash v$.

If $v = \phi$ and all customers are satisfied, it means the initial solution is obtained, go to **step 6.**

Step 3: Obtain the feasible demand set D^v for label v according to formula (18)–(22);

If $D^v = \Phi$, it implies that that label v cannot visit any demand sites, go to **step 5**.

Else, run *computeDv*(v, d) to find the best demand d from set D^v.

Step 4: Create new vehicle label $v' = f(v, d)$ according to formula (23), and set $v = v'$, and go to **step 3**.

Step 5: Create route label $r = f(v)$ using formula (24), and put it back the route set R.

Go to **step 2**.

Step 6: End the procedure, and a feasible solution $R = \Psi^{ini}$ is obtained. Complete procedure.

4.2. Improving the initial solution

Similar to Duhamel et al. [13], the minimization of the vehicle number takes precedence over the total route length in this paper. In other words, for a given number of customers and constraints, the fewer the vehicles are required, the better the algorithm is. Under the circumstances of the same vehicle quantity, a shorter travel distance is further exploited to lower the overall service cost.

4.2.1. *Vehicle minimization*

In this section, we improve the initial solution by minimizing the vehicle number using the ant colony algorithm. This involves the global route Ψ^{gb} construction and the single path construction Ψ^{m}. The working principle of vehicle minimization algorithm is described as below.

Step 1: If $k \le 1$, only one tour exists in the solution, go to **step 4**;

With initialization of the pheromone trails and insertion matrix (record the times a customer has been visited), the best solution ψ^{asc_vei} is obtained by the present ant colony with k vehicles, and the best solution Ψ^{gb} is obtained up to now. Let $k = k - 1$;

Q6 **Step 2:** l = *AntsBatch*, difinesants computation iterated times.

For ant m of batch l, construct a solution from:

Insert the missing demands $\Psi^{m} = Ant\_Solution\_Vehicle(m, Insert)$; the insertion procedure (insert the unvisited customers into solution ψ^{m}) will be conducted if not all customers have been served; and now if ψ^{m} is feasible, update the best solution.

If Distance(Ψ^{m}) < Distance(Ψ^{asc_vei}) where Distance(Ψ^{m}) is the total length of solution ψ^{m}, update the best solution $\psi^{asc\_vei} = \psi^{m}$, and also update the pheromone matrix according to $\tau_{ij} = (1-\rho)\tau_{ij} + \rho/\text{distance}(\psi^{asc\_vei}), \forall (i, j) \in \psi^{asc\_vei}$. If ψ^{m} is not feasible, let $m = m + 1$ and go to **step 2**.

Step 3: If Distance$\left(\psi^{asc_vei}\right)$ < Distance$\left(\psi^{gb}\right)$, let $\psi^{gb} = \psi^{asc\_vei}$, and also update

pheromone according to $\tau_{ij} = (1-\rho)\tau_{ij} + \rho/\text{Distance}(\psi^{gb}), \forall (i, j) \in \psi^{gb}$. If $l > 0$, let $l = l + 1$, and go to **step 2**. If all ψ^{m} are feasible, go to **step 1**; otherwise go to **step 4**.

Step 4: End the procedure, print ψ^{gb}. Complete procedure.

Although the approach to constructing the ψ^{m} solution is similar to the construction of the initial solution, there are two major distinctions:

1) Exploration and exploitation are applied

We use the pseudo random proportional principle to construct solution ψ^{m} for ant m, and create a random constant q between [0,1] to comply with the average distribution rule.

If $q > q_0$ (q_0 is a parameter to be set as $0 < q_0 < 1$), we create a random number $\bar{q}$ from Eq. (25), and rank the demands D^{v} in ascending order according to $P_{ij}^{k} = \left((\tau_{ij})^{\alpha}.(\eta_{ij})^{\beta}\right)/\left(\sum_{j \in D^{v}} (\tau_{ij})^{\alpha}.(\eta_{ij})^{\beta}\right)$. Then, we sum the data one at a time until $\sum_{j \in D^{v}} P_{ij}^{k} > \bar{q}$ is satisfied, and finally choose the next demand d.

If $q \le q_0$, we select the demand based on $\max\left\{(\tau_{ij})^{\alpha}.(\eta_{ij})^{\beta}\right\}$, $\eta_{ij} = \left[\text{distance}(v, d) - \text{insert}_j\right]^{-1}$ from Eq. (26).

After visiting a customer, the local pheromone trails are updated according to $\tau_{ij} = (1-\rho)\tau_{ij} + \rho\tau_0$ with τ_0 being the initial pheromone trails.

2) Insertion procedure

We attempt to insert the unvisited customers into the constructed routes. This process is denoted as the insertion procedure, meaning any customers who are not covered due to the vehicle reduction should be inserted into these routes generated. According to the demand quantity, we sort all the unvisited customers in a descending order, and insert these customers one by one into existing routes. An insertion is considered as successful if the time windows and the capacity constraints are not violated. A feasible solution is obtained when all these unvisited customers are inserted successfully. Otherwise, the number of vehicles should not be reduced further.

4.2.2. *Minimizing the length of routes*

In this section, we describe how to minimize the length of a route with a fixed number of vehicles. The procedure presented here is similar to the tour minimization with the exception that we apply local search to minimize the route length using intra-exchange and cross-exchange schemes.

Step 1: Initialize pheromone trails, the best solution $\psi^{a\_s\_v}$ is obtained by the present ant colony, and obtain the temporary best solution ψ^{gb}.

Step 2: For ant m in batch l, construct a solution ψ^{m} = Ant_Solution_Vehicle(m, Insert). If not all customers have been visited, perform the insertion action in an attempt to obtain a feasible solution ψ^{m}.

Step 3: Improve ψ^{m} with local search. If Distance$\left(\psi^{m}\right)$ < Distance$\left(\psi^{asc_vei}\right)$, save the

best solution: let $\psi^{asc\_vei} = \psi^{m}; \psi^{gb} = \psi^{m}$. If Distance$\left(\psi^{asc_vei}\right)$ < Distance$\left(\psi^{gb}\right)$, save the best solution $\psi^{gb} = \psi^{asc\_vei}$, and update pheromone trails globally using the following criteria:

$$\tau_{ij} = (1-\rho)\tau_{ij} + \frac{\rho}{\text{Distance}(\psi^{gb})} \quad \forall (i, j) \in \psi^{gb}$$

$$\tau_{ij} = (1-\rho)\tau_{ij} + \frac{\rho}{\text{Distance}(\psi^{asc\_vei})} \quad \forall (i, j) \in \psi^{asc\_vei}$$

If $m < \max\{m\}$ then $m = m + 1$; go to **step 2**. If $l < \max\{l\}$ then $l = l + 1$; go to **step 2**.

Step 4: End the procedure, and print the solution for ψ^{gb}, complete the procedure.

Next we explain the local search mechanism consisting of intra-exchange and cross-exchange as it differs from traditional tour minimization algorithm.

1) Intra-exchange

The intra-exchange implemented in LACS is called 2-Opt [28], Q7 and the working principle is presented as follows. We choose a pair of demands on a route and exchange them. If the route length becomes shorter, we mark them locally and continue a new pair exchange. Once all demand pairs have been exchanged, we select the demand pair with the largest saving of distance and exchange them (i.e., update the route). This process is repeated until no further improvement can be made on this route.

2) Cross-exchange

Cross-exchange is proposed by Taillard et al. [39]. The idea is similar to intra-exchange. We choose an arc from a route, exchange it with all the arcs in another route. If the total route length becomes smaller, we mark them and continue the next exchange. Once all pairs of arcs have been exchanged, we select the arc pair that results in the largest distance saving and exchange them (i.e., update the solution). This process is repeated until no further improvement can be made on all the routes.

5. Numerical experiments

5.1. *Background of the testing data sets*

We test the performance of the proposed LACS algorithm based on three different data sets. First, we use the Solomon benchmark problem as the test bed, which has been widely used to test the efficiency of vehicle routing algorithms with time windows (VRPTW). Second, we test LACS by solving a series of vehicle routing problems in heterogeneous fleet settings. Third, we apply the LACS algorithm to optimize the vehicle routing for China Post in Guangzhou consisting of 84 nodes. This real-world problem involves multiple vehicle types, time windows, mixed loads, and random orders of linehaul and backhaul tasks. Below we briefly describe the features and scope of these date sets.

The Solomon benchmark problem consists of three classes R/C/RC and six groups of problem sets R1/R2, C1/C2, RC1/RC2 with total of twelve basic problems. These problems vary in terms of the time windows and the percentage of customers within the time window. With one centralized depot, each problem set has three demand groups (i.e., 25/50/100) randomly located across the network. We compare our results to those from MA/SA algorithms by Cheung and Hang [8], ESPPRC algorithm by Jepsen [22] and IPopt algorithm by Kohl et al. [23] using the R1 problem set.

In order to compare the performance of solving the vehicle routing problems with heterogeneous fleet, we use the data sets from Taillard [38] that contain eight problems (numbered from 13 to 20) with three to six types of vehicles. Taillard actually adopted the problem setting from Golden et al. [19] and transformed it into a VRPHF testing bed by allocating relevant parametric data, such as variable cost per unit distance per vehicle type and the number of available vehicles per type. The size of these problems varies between 50 and 100 customers, and there are no restrictions on route length and customer service times. In this section, we compare the results generated by our algorithm with those from Taillard [38] and Brandão [4].

We further use seven postal shuttle-lines of China Post in Guangzhou to demonstrate how the proposed algorithm improves the cost savings in an actual business setting. The postal shuttles deliver packages to the post offices, and also collect and send the packages to the mail distribution center. We use the shuttle dispatch data from Wusan Distribution Center to optimize the vehicle type, vehicle number, and driving paths. The Geographic Information Systems (GIS) data are gathered from SuperMap 6R.

The experiments for each problem data set are conducted by *Ants* = 10, runs with 15 times *AntsBatch* = 15 (i.e., Iteration number is 15), total *RepeatTime* = 10 (i.e., the algorithm runs 10 times) or stop after getting the best optimal solution for five consecutive times with no improvement. Solutions are then averaged for each problem and the results are presented in Tables 2–4. Parameters for the experiments are set as: $q_0 = 0.9$, $\alpha = 1$, and $\beta = 1$, the coefficients for updating the global and the local pheromone trails are $\rho_g = \rho_l = 0.1$. The LACS algorithm is implemented in Java 1.6.0 and all tests are performed on a Dell P4 computer with 2.4 GHz Intel CPU and 1 GB RAM.

Since algorithms in extant literature are executed in different generations of computers, to make a fair comparison, we convert the CPU times of different algorithms suggested by Brandão [4] and Dongarra [11]. We adopt the approach proposed by Gajpal and Abad [15] to offset the speed variation of different computers. The results of the conversion are summarized in Table 1. It shows that the Mflops of our computer is faster than those used in prior studies except for Jepsen's. In other words, had their algorithms been executed on our computer, the CPU times would be equal to the original computing times multiplied by the conversion rate.

5.2. Performance comparison using Solomon R1 benchmark datasets

Comparisons between LACS, MA/SA, ESPPRC and IPopt algorithms are reported in Table 2. According to Cheung and Hang [8], MA ends up with a shorter route but requires longer computational time, while SA produces better solutions but consumes more time. Although ESPPRC can generate the best solutions using exact algorithm, it consumed a much longer CPU time. In Table 2, all bold numbers indicate the best solution among these algorithms.

In terms of the total route length, LACS is better than MA/SA in 26 instances except for R110.50. Even in that case, LACS still saves one trip compared to SA and ESPPRC algorithms. For the number of vehicles, except for R106.50, LACS ends up with a smaller number than both MA/SA and the IPopt algorithm. It is noteworthy that LACS uses fewer vehicles than the IPopt algorithm in eight instances. For the CPU time, LACS is close to that of MA, and almost always less than that of SA. So we can conclude that LACS is more efficient based on the fact that LACS consumes less CPU time than SA.

In comparison with ESPPRC, we find that the results from LACS are almost the same as those from the exact algorithm for small and medium instances. Even for large instances, the gaps are no more than 4.6%. Some results under ESPPRC are missing because their computer could not generate the exact solution in a reasonable CPU time. This implies that LACS offers more flexibility than ESPPRC in finding an optimal or sub-optimal solution for large size problems.

5.3. Performance comparison under heterogeneous fleet

Under fleet heterogeneity, we compare the solution generated from LACS with those from Taillard [38] and Brandão [4], and the results are listed in Table 3. It shows that the LACS is able to find 75% of the best known solutions among all the testing instances. In comparison with the algorithm of Taillard [38], the LACS always ensures a better solution for all the problem instances. In comparison with Brandão [4], both algorithms yield the same number of the best solutions among the eight testing instances, but the equivalent computation time of LACS is always less than that of Brandão [4] across all instances. It is worth mentioning that algorithms by Taillard [38] and Brandão [4] are designed to tackle heterogeneous fleet problems only, while LACS can solve a much wider array of vehicle routing problems that involve heterogeneity and backhaul mixed loads services.

5.4. Application to China Post in Guangzhou

We use China Post in Guangzhou to demonstrate the effectiveness of the LACS algorithm by solving a large postal shuttle transportation problem on VRPHBMTW issue. Located in Guangzhou, the company provides postal services for the downtown area and its vicinities. Its facility consists of eight subordinate units and nine sub-companies with more than 360 outlets, 120 delivery stations and a large fleet of heterogeneous vehicles. Among them, there are more than 60 postal shuttles traveling nearly 7000 km per day. These postal shuttles send the inbound mails from the central processing center to the delivery stations, and also pick up the outbound mails from the post offices and send back to the central processing center. Current routing schedules are established primarily upon experiences and administrative divisions.

In this application, we use LACS to minimize the total vehicle routing cost for two cases, namely, a 48-node network with five routes, and an 84-node network with ten routes. In each case, two scenarios are considered: heterogamous fleet versus homogenous fleet. We compared the costs generated from both vehicle fleets to find which scenario is more competitive in terms of overall cost. The necessary data include three types of vehicles (i.e., 1.5, 3, and 5 tons), variable transportation costs (i.e., 1.5, 2, and 2.2 RMB/km), and a fixed waiting cost (i.e., 6 RMB/h). We define Gap = ((Post – Optimization result) – Pre-Optimization result)/Pre-Optimization result × 100% as the performance metric to measure the potential cost savings generated by LACS algorithm.

Q10 **Table 1**
Mflops for different computers and conversion factors.

Algorithm	Actual computer used	Approx. equivalent comp. available in report	Mflops	Source	Conversion
Taillard [38]	Sun Sparc workstation 50 MHz	Sun Sparc workstation 50 MHz	10	b	0.0070
Cheung and Hang [8]	PII 400 MHz	Pentium II, 333 MHz	69	a	0.0668
Ropke and Pisinger [31]	Pentium IV 1.5 GHz	Pentium IV 1.5 GHz	326	a	0.3156
Li et al. [26]	Athlon 1 GHz	Athlon 1 GHz	450	b	0.3129
Gajpal and Abad [15,16]	Xeon 2.4 GHz	Xeon 2.4 GHz	884	a	0.8558
Brandão [4]	Compaq PresarioX 1000 Intel Pentium M 1.4 GHz	Compaq PresarioX 1000 Intel Pentium M 1.4 GHz	250	b	0.1739
Jepsen [22]	P4 3.0 GHz	Pentium 4, 3.06 GHz	1414	a	1.3688
Ours	P4 2.4 GHz	P4 2200 MHz	1033	a	1

Note: “a” is from Dongarra [11] and “b” is based on Brandão [4].

Q12 **Table 2**
Comparisons using Solomon R1 benchmark data.

Instance	*q*	Distance				Vehicles				Converted CPU time			
		LACS	MA[a]	SA[a]	ESPPRC[b]	LACS	MA	SA	IPopt[c]	LACS	MA	SA	ESPPRC
R101	25	**617.1**	670.4	634.9	617.1	**8**	8	8	8	0.11	0.03	0.13	0.03
	50	1054.5	1132.9	1104.4	1043.4	**12**	12	12	12	0.54	0.13	1.2	0.19
	100	**1650.4**	2034.7	1970.1	1631.2	**19**	21	21	20	2.58	1.2	9.22	27.49
R102	25	**546.4**	635.4	594.1	546.4	**7**	7	7	7	0.12	0.2	4.48	0.18
	50	**919.7**	1062.1	1037	909	**10**	11	11	11	**0.48**	1.2	9.75	0.37
	100	1486.1	1810	1711.1	1466.6	**18**	19	19	18	3.08	2.47	34.27	6.01
R103	25	**463.4**	556.7	529.9	454.6	**4**	5	5	5	0.24	0.2	1.34	0.15
	50	**806.4**	883	864.9	769.3	**8**	9	9	9	1.11	0.67	26.05	6.82
R104	25	417.0	501.4	486.4	416.9	**4**	4	4	4	**0.27**	0.6	4.48	0.16
	50	626.0	802	767.1	619.1	**6**	7	7	6	**1.49**	1.54	27.05	45.57
R105	25	**531.7**	584	602.8	530.5	**5**	6	6	6	**0.2**	0.4	3.41	0.03
	50	**918.6**	1096	1231.9	892.2	**8**	10	10	10	0.88	0.6	7.35	3.81
	100	1377.1	1538.1	1543.2	1346.2	**15**	15	15	15	**2.55**	2.87	32.26	173.78
R106	25	466.5	522.3	498.1	457.3	**3**	3	3	3	**0.27**	0.27	0.27	0.40
	50	**793.1**	912.6	897.6	791.4	7	6	**5**	**5**	**0.91**	0.94	2.07	**1.93**
R107	25	**425.3**	518.3	469.9	424.3	**4**	4	4	4	0.31	0.2	7.41	0.16
	50	726.2	755.2	776.3	707.3	**7**	8	8	7	**1.06**	3.01	13.69	7.61
R108	25	397.2	459.8	448.3	396.9	**4**	4	4	4	0.31	0.2	9.15	0.42
R109	25	442.3	512.1	501.2	441.3	**5**	5	5	5	0.25	0.07	4.21	0.08
	50	**786.8**	1065.3	948.2	775.4	**7**	9	8	8	0.47	0.6	0.53	27.53
R110	25	445.2	512.3	480.1	438.4	**4**	4	4	4	0.28	0.27	0.53	1.59
	50	707.4	767.8	702.4	695.1	**7**	8	8	7	0.44	0.33	0.6	4.63
R111	25	**429.1**	510.2	491.3	427.3	**4**	5	5	5	0.25	0.2	2.14	0.31
R112	25	390.1	443.1	426.4	387.1	**4**	4	4	4	**0.27**	0.27	0.27	1.63

Note: “a” is from Cheung, and “b” is from Jepsen, and “c” is from Kohl.

Table 3
Comparisons in VRP with heterogeneous fleet instances.

Instance	Nodes	Taillard [38]		Brandão [4]		LACS		Cost gap
		Cost	Time	Cost	Time	Cost	Time	
13	50	1536.55	3.29	**1517.84**	9.74	**1517.84**	7.86	0%
14	50	623.05	4.00	**607.53**	9.56	**607.53**	10.13	0%
15	50	1022.05	2.33	**1015.29**	10.26	**1015.29**	7.95	0%
16	50	1159.14	2.43	**1144.94**	16.34	1145.51	11.04	0.05%
17	75	1095.01	15.61	**1061.96**	35.81	**1061.96**	31.12	0%
18	75	1894.73	20.00	1831.36	34.42	**1830.26**	30.42	0.37%
19	100	1156.93	40.56	1120.34	42.25	**1120.34**	36.21	0%
20	100	1592.16	23.66	**1534.17**	52.50	1544.07	32.41	0.65%

The optimal routing schedules for the 48-node network and the 84-node network are depicted in Figs. 2 and 3 respectively. Post offices, delivery stations, and the central processing center are denoted as circles, squares and the ellipse, respectively. Table 4 lists the cost items and the vehicle allocations for the 48-node network. It shows that the total cost is reduced from RMB 2440 to 2377. The maximum cost saving from a single route is Wusan, down by 13.24%. Table 5 lists the cost items and the vehicle allocation scheme for the 84-node network. Prior to the optimization, the daily service cost relying on ten routes is RMB 4628. LACS is able to generate a 9-route solution, and the vehicle used to serve Shahe is no longer needed. Though the local costs for certain routes may increase, the total service cost is down from RMB 4628 to 4252, resulting in an 8.12% saving. As the network size increases (i.e., extend to the metropolitan Guangzhou), the potential cost savings of using LACS generated routing solution becomes more promising.

Table 4
Solutions for China Post in Guangzhou using LACS (48 Nodes).

No.	Pre-optimization				Post-optimization				Gap
	Route	Nodes	Vh. type (ton)	Cost (RBM)	Routes	Nodes	Vh. type (ton)	Cost (RMB)	
1	Wusan	7	1.5	272	Wusan	**6**	1.5	**236**	**−13.2%**
2	Zhujiang Newtown	11	5	348	Zhujiang Newtown	12	5	389	11.8%
3	East Guangyuan	9	3	660	East Guangyuan	**8**	3	**602**	**−8.8%**
4	Longdong	11	5	664	Longdong	**11**	5	**646**	**−2.7%**
5	Yantang	10	3	496	Yantang	11	5	504	1.6%
	Total	48		2440	Total	**48**		**2377**	**−2.6%**

Table 5
Solutions for China Post in Guangzhou using LACS (84 Nodes).

No.	Pre-optimization				Post-optimization				Gap
	Route	Nodes	Vh. type (ton)	Cost (RBM)	Routes	Nodes	Vh. type (ton)	Cost (RMB)	
1	Wusan	7	1.5	272	Wusan	**6**	1.5	**236**	**−13.2%**
2	Zhujiang Newtown	11	5	348	Zhujiang Newtown	12	5	400	14.9%
3	East Guangyuan	9	3	660	East Guangyuan	**8**	3	**600**	**−9.1%**
4	Longdong	11	5	664	Longdong	12	5	676	1.8%
5	Yantang	10	3	496	Yantang	12	5	532	7.3%
6	Liuhua	8	3	440	Liuhua	9	3	448	1.8%
7	Shahe	5	1.5	416	Shahe	**0**	1.5	**0**	**−100.0%**
8	Wuyangcun	8	3	442	Wuyangcun	9	3	453	2.5%
9	Haiyin	9	5	520	Haiyin	12	5	558	7.3%
10	Tiyudong	6	1.5	370	Tiyudong	**4**	1.5	**349**	**−5.7%**
	Total	84		4628	Total	**84**		**4252**	**−8.1%**

Table 6
Comparisons between heterogeneous and homogenous vehicles.

Instance	Status	Vh. type (ton)	Vehicle num.	Cost (RMB)	Gap	Solution for Vh. num (Vh. type)
48 nodes	Pre-LACS	1.5/3/5	5	2440	Baseline	1(1.5), 2(3), 2(5)
	Post-LACS	1.5/3/5	5	2377	−2.6%	1 (1.5), 1(3), 3(5)
	Post-LACS	1.5	10	2580	5.8%	10(1.5)
	Post-LACS	3	7	2514	3.0%	7(3)
	Post-LACS	5	5	2467	1.1%	5(5)
84 nodes	Pre-LACS	1.5/3/5	10	4628	Baseline	3(1.5), 4(3), 3(5)
	Post-LACS	1.5/3/5	9	4252	−8.1%	3(1.5), 3(3), 4(5)
	Post-LACS	1.5	18	5056	9.2%	18(1.5)
	Post-LACS	3	12	4865	5.1%	12(3)
	Post-LACS	5	10	4773	3.1%	10(5)

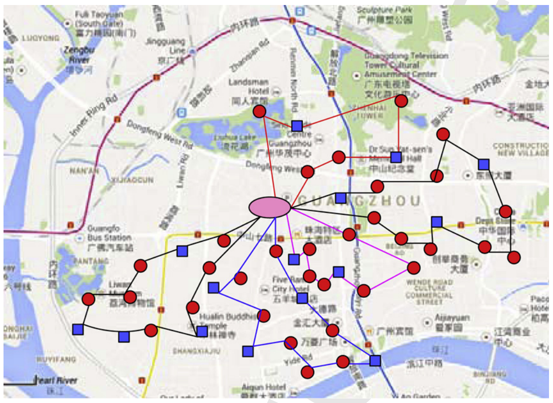
Fig. 2. 48-Node Shuttle Service Area.

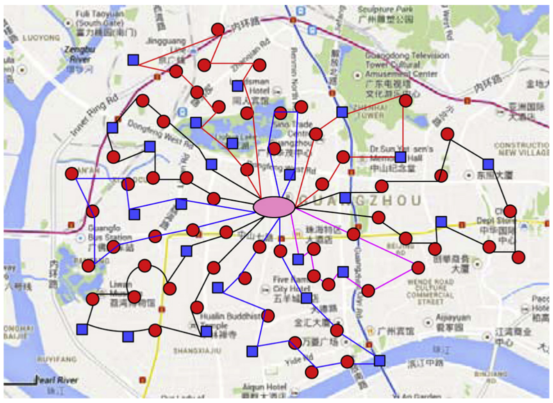
Fig. 3. 84-Node Shuttle Service Area.

Cost comparisons between heterogeneous and homogenous vehicles are also made in Table 6. It shows that the operational costs incurred by the homogenous vehicle fleet are always higher than that of the heterogeneous fleet. As the network nodes increases, the cost of using small size vehicles grows much faster than using large size vehicles. These managerial insights generated by LACAS can aid the regional China Post in determining the optimal vehicle mix to realize cost savings, yet without compromising the service quality.

6. Conclusion

This paper formulates and solves a class of vehicle routing problems considering vehicle heterogeneity, backhauls, mixed-load, and time windows. Both linehauls and backhauls can be served in random order. Our study intends to fill the research gap where vehicle heterogeneity and backhaul mixed-loads are often handled separately. A two-stage label-based ant colony optimization algorithm is developed to minimize the total travel cost. That is, in stage one we optimize the vehicle type and vehicle quantity, and in stage two we further optimize the travel routes based on the previous stage decision. The proposed hybrid algorithm possesses the multi-attribute labeling features and the swarm intelligence, and its advantages are demonstrated in terms of global optimality and computational time. The real-world case study in China Post of Guangzhou shows that a heterogeneous fleet of vehicles can lower the service cost up to 9.2% than a homogeneous fleet. As of future research efforts, we want to extend the algorithm to more general situations including periodic vehicle assignment, and simultaneous pickup and delivery in a multi-depot environment. We also want to tackle the problem by using directly multi-objective evolutionary algorithm to obtain the non-dominant solution set, and the results can be further compared with the ant colony algorithm in terms of computational time and global optimality.

Acknowledgments

Q8 This research is supported in part by XXX (#XX), and by XXX (XXX).

References

[1] S. Anily, The vehicle-routing problem with delivery and back-haul options, Nav. Res. Logist. 43 (3) (1996) 415–434.
[2] F. Belmecheri, C. Prins, F. Yalaoui, L. Amodeo, Particle swarm optimization algorithm for a vehicle routing problem with heterogeneous fleet, mixed backhauls, and time windows, J. Intell. Manuf. 24 (4) (2013) 775–789.
[3] R. Bent, P. Van Hentenryck, A two-stage hybrid algorithm for pickup and delivery vehicle routing problems with time windows, Comput. Oper. Res. 33 (4) (2006) 875–893.
[4] J. Brandão, A tabu search algorithm for the heterogeneous fixed fleet vehicle routing problem, Comput. Oper. Res. 38 (1) (2011) 140–151.
[5] J. Brito, F.J. Martínez, J.A. Moreno, J.L. Verdegay, An ACO hybrid metaheuristic for close–open vehicle routing problems with time windows and fuzzy constraints, Appl. Soft Comput. 32 (7) (2015) 154–163.
[6] D. Casco, B. Golden, E. Wasil, Vehicle routing with backhauls, in: B.L. Golden, Q9 A.A. Assad (Eds.), Vehicle Routing: Methods and Studies, North-Holland, Amsterdam, 1988, pp. 127–147.
[7] J. Chen, T. Wu, Vehicle routing problem with simultaneous deliveries and pickups, J. Oper. Res. Soc. 57 (5) (2006) 579–587.
[8] R.K. Cheung, D.D. Hang, Multi-attribute label matching algorithms for vehicle routing problems with time windows and backhauls, IIE Trans. 35 (3) (2003) 191–205.
[9] R.K. Cheung, N. Shi, W.B. Powell, H.P. Simao, An attribute-decision model for cross-border drayage problem, Transp. Res. E 44 (2) (2008) 217–234.
[10] Y. Collette, P. Siarry, Multiobjective Optimization: Principles and Case Studies, Springer, 2003 (2011).
[11] J.J. Dongarra, Performance of Various Computers using Standard Linear Equations Software, Technical Report, University of Tennessee, 2011.
[12] M. Dorigo, L.M. Gambardella, Ant colony system a cooperative learning approach to the traveling salesman problem, IEEE Trans. Evol. Comput. 1 (1) (1997) 53–66.
[13] C. Duhamel, J. Potvin, J. Rousseau, A tabu search heuristic for the vehicle routing problem with backhauls and time windows, Transp. Sci. 31 (1) (1997) 49–59.
[14] B. Eksioglu, A.V. Vural, A. Reisman, The vehicle routing problem: a taxonomic review, Comput. Ind. Eng. 57 (4) (2009) 1472–1483.
[15] Y. Gajpal, P.L. Abad, An ant colony system (ACS) for vehicle routing problem with simultaneous delivery and pickup, Comput. Oper. Res. 36 (12) (2009) 3215–3223.
[16] Y. Gajpal, P.L. Abad, Multi-ant colony system (MACS) for a vehicle routing problem with backhauls, Eur. J. Oper. Res. 196 (1) (2009) 102–117.
[17] L.M. Gambardella, E.D. Taillard, G. Agazzi, MACS-VRPTW a multiple colony system for vehicle routing problems with time windows, in: D. Corne, M. Dorigo, F. Glover (Eds.), New Ideas in Optimization, McGraw-Hill, London, UK, 1999, pp. 63–76.
[18] F.P. Goksal, I. Karaoglan, F. Altiparmak, A hybrid discrete particle swarm optimization for vehicle routing problem with simultaneous pickup and delivery, Comput. Ind. Eng. 65 (1) (2013) 39–53.
[19] B. Golden, A. Assad, L. Levy, F. Gheysens, The fleet size and mix vehicle routing problem, Comput. Oper. Res. 11 (1) (1984) 49–66.
[20] B. Golden, E. Baker, J. Alfaro, J. Schaffer, The vehicle routing problem with backhauling: two approaches, in: Proceedings of the Twenty-First Annual Meeting of the S.E. TIMS, Myrtle Beach, SC, USA, 1985.
[21] F. Hennig, B. Nygreen, M.E. Lübbecke, Nested column generation applied to the crude oil tanker routing and scheduling problem with split pickup and split delivery, Nav. Res. Logist. 59 (3–4) (2012) 298–310.
[22] M.K. Jepsen, Branch-and-cut and Branch-and-cut-and-price Algorithms for Solving Vehicle Routing Problems, Technical University of Denmark, Kgs. Lyngby, Copenhagen, 2011.
[23] N. Kohl, J. Desrosiers, O.B.G. Madsen, M.M. Solomon, F. Soumis, 2-path cuts for the vehicle routing problem with time windows, Transp. Sci. 33 (1) (1999) 101–116.
[24] İ. Küçükoğlu, N. Öztürk, An advanced hybrid meta-heuristic algorithm for the vehicle routing problem with backhauls and time windows, Comput. Ind. Eng. 86 (8) (2015) 60–68.
[25] G. Laporte, Fifty years of vehicle routing, Transp. Sci. 43 (4) (2009) 408–416.
[26] F. Li, B. Golden, E. Wasil, A record-to-record travel algorithm for solving the heterogeneous fleet vehicle routing problem, Comput. Oper. Res. 34 (9) (2007) 2734–2742.
[27] T.L. Li, C. Chang, Solve pick up and delivery problem with time windows via a guided adaptive ant colony system, Transp. Plan. J. 39 (1) (2010) 99–132.
[28] S. Lin, B.W. Kernighan, An effective heuristic algorithm for the traveling-salesman problem, Oper. Res. 21 (2) (1973) 498–516.
[29] R. Liu, X. Xie, V. Augusto, C. Rodriguez, Heuristic algorithms for a vehicle routing problem with simultaneous delivery and pickup and time windows in home health care, Eur. J. Oper. Res. 230 (3) (2013) 475–486.
[30] W.B. Powell, W. Snow, R.K. Cheung, Adaptive labeling algorithms for the dynamic assignment problem, Transp. Sci. 34 (1) (2000) 50–66.
[31] S. Ropke, D. Pisinger, A unified heuristic for a large class of vehicle routing problems with backhauls, Eur. J. Oper. Res. 171 (3) (2006) 750–775.
[32] M. Reed, A. Yiannakou, R. Evering, An ant colony algorithm for the multi-compartment vehicle routing problem, Appl. Soft Comput. 15 (2) (2014) 169–176.
[33] S. Salhi, G. Nagy, A cluster insertion heuristic for single and multiple depot vehicle routing problems with backhauling, J. Oper. Res. Soc. 50 (10) (1999) 1034–1042.
[34] S. Salhi, N. Wassan, M. Hajarat, The fleet size and mix vehicle routing problem with backhauls: formulation and set partitioning-based heuristics, Transp. Res. E 56 (2013) 22–35.
[35] D. Shishebori, M.S. Jabalameli, Improving the efficiency of medical services systems: a new integrated mathematical modeling approach, Math. Probl. Eng. 2013 (2013) 1–13 (Article ID 649397).
[36] M.M. Solomon, Algorithms for the vehicle routing and scheduling problems with time window constraints, Oper. Res. 35 (2) (1987) 254–265.
[37] S. Tasan, M. Gen, A genetic algorithm based approach to vehicle routing problem with simultaneous pick-up and deliveries, Comput. Ind. Eng. 62 (3) (2012) 755–761.
[38] E.D. Taillard, A heuristic column generation method for the heterogeneous fleet VRP, RAIRO: Oper. Res. 33 (1) (1999) 1–14.
[39] E.D. Taillard, P. Badeau, M. Gendreau, F. Guertin, J.Y. Potvin, A tabu search heuristic for the vehicle routing problem with soft time windows, Transp. Sci. 31 (2) (1997) 170–186.
[40] P. Toth, D. Vigo, An exact algorithm for the vehicle routing problem with backhauls, Transp. Sci. 31 (4) (1997) 372–385.
[41] T. Vidal, T.G. Crainic, M. Gendreau, C. Prins, A Unified Solution Framework for Multi-attribute Vehicle Routing Problems. Tech. Report, CIRRELT, Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport, Université de Montréal, Montréal, QC, Canada, 2012.
[42] T. Vidal, T.G. Crainic, M. Gendreau, C. Prins, Heuristics for multi-attribute vehicle routing problems: a survey and synthesis, Eur. J. Oper. Res. 231 (1) (2013) 1–21.