Computers & Industrial Engineering 133 (2019) 95–106

Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

A multi-compartment vehicle routing problem with time windows for urban distribution – A comparison study on particle swarm optimization algorithms

Jiumei Chen[a,b], Jing Shi[c,*]

[a] School of Business Planning, Chongqing Technology and Business University, Chongqing 400067, China
[b] Chongqing Key Laboratory of Electronic Commerce and Supply Chain System, Chongqing Technology and Business University, Chongqing 400067, China
[c] Department of Mechanical & Materials Engineering, College of Engineering and Applied Science, University of Cincinnati, Cincinnati, OH 45221, USA

ARTICLE INFO

Keywords:
Multi-compartment vehicle routing problem
Vehicle routing problem with time windows
Hybrid particle swarm optimization
Simulated annealing
Urban distribution

ABSTRACT

In this paper, we introduce a multi-compartment vehicle routing problem with time window (MCVRPTW) arising from urban distribution, which essentially reflects the "last mile" delivery challenge for modern logistics. The distinguishing feature of MCVRPTW is that products must be transported in independent vehicle compartments because they cannot be mixed together due to differences in their characteristics. Based on the mathematical formulation established for the optimization problem, we propose and compare two solution approaches, with one being the hybrid particle swarm optimization (HPSO) with simulated annealing, and the other being the conventional particle swarm optimization (PSO). Based on Solomon's vehicle routing problem with time windows (Solomon, 1987), experimental instances with 25 customers, 50 customers and 100 customers are developed to investigate the performance of the proposed solution approaches. The results indicate that both approaches are reasonably efficient for the MCVRPTW. Moreover, the HPSO algorithm has overall better performance, particularly in delivering the best solution, for all cases, and the HPSO algorithm becomes more efficient than the PSO algorithm as the problem size increases. On the other hand, the PSO algorithm shows a slight edge in terms of the worst solution and standard deviation.

1. Introduction

Vehicle Routing Problem (VRP) was first developed by Dantzig and Ramser in 1959, which is a typical NP-hard problem in combinatorial optimization. VRP has been studied extensively and rich research results have been achieved (Golden, Raghavan, & Wasil, 2008; Lahyani, Khemakhem, & Semet, 2015; Laporte, 2009; Toth & Vigo, 2014). Most of the literature only considers one type of product. However, in many practical applications of VRP, more than one type of products is involved. Typical examples include cold chain logistics with refrigerated and non-refrigerated products, domestic waste collection with many types of products from different recycling boxes, fuel delivery with multiple petroleum products. In those cases, different types of products cannot be mixed together in the same compartment during the transportation. Considering the transport efficiency, the capacity of a vehicle can be split into several compartments to service different types of products simultaneously. Therefore, the multi-compartment vehicle routing problems (MCVRP) are coined (Brown & Graves, 1981), which consider several compartments in a vehicle to serve more than one product type at the same time. Because MCVRP was originally proposed for the application of fuel delivery, there are many studies reported in this field (Cornillier, Boctor, Laporte, & Renaud, 2008a,b; Cornillier, Laporte, Boctor, & Renaud, 2009; Popović, Vidović, & Radivojević, 2012; Vidović, Popović, & Ratković, 2014). The MCVRP for other applications have also been studied, such as bulk shipping (Caramia & Guerriero, 2010; Fagerholt & Christiansen, 2000; Hvattum, Fagerholt, & Armentano, 2009), recycling and waste management (Elbek & Wøhlk, 2016; Rabbani, Farrokhi-asl, & Rafiei, 2016; Silva, 2016), transportation of live animals to slaughterhouses (Oppen & Løkketangen, 2008), olive oil collection (Lahyani, Coelho, Khemakhem, Laporte, & Semet, 2015), distribution of livestock feed (Kandiller, Eliiyi, & Taşar, 2017).

In this paper, we introduce a new MCVRP arising from urban distribution. Driven by consumer demands and technological advancement, the retail industry has started to move away from the form of single channel value chain to meet the fragmentation, individualization and timeliness of consumer orders. This new development is driving the

* Corresponding author.
E-mail address: jing.shi@uc.edu (J. Shi).

https://doi.org/10.1016/j.cie.2019.05.008
Received 26 November 2018; Received in revised form 4 May 2019; Accepted 6 May 2019
Available online 08 May 2019
0360-8352/

transformation of logistics industry. In the new form of business, business flow determines logistics, and the chain position and core competitiveness of urban distribution logistics enterprises are increasingly prominent. Naturally, the safety, speed and efficiency of the "last mile" distribution and transportation are the important targets pursued by many urban distribution logistics enterprises. Compared with the aforementioned applications such as fuel delivery, bulk shipping and recycling, the urban distribution has its own unique characteristics. For example, a variety of products are often required to be delivered together to reduce the delivery times. The compartment size is usually fixed since different products need to be transported at different compartments to avoid odor or mutual contamination. Also, the customers usually have preferred time windows because they may not be willing or able to wait for the delivery for the entire day.

The MCVRP with time window is often referred to as MCVRPTW. The MCVRPTW is a generalization of the VRPTW, in which each vehicle has only one compartment. The MCVRPTW deals with meeting the customer demand for different products. The products should be stored in different compartments of the same vehicle while being transported. Vehicles are partitioned into a number of compartments with certain capacities. Customers are assigned to routes so that the demand of customers assigned to any route for certain product does not exceed the capacity of the reserved compartment for this product and the customers are served in their preferred time windows. The objective is to minimize the total transportation cost. Therefore, the MCVRPTW is a complex problem that requires innovative solution approaches. Considering that MCVRP is an extension of VRP and particle swarm optimization (PSO) algorithm has good performance in solving the VRP (Golden et al., 2008; Lahyani, Khemakhem, et al., 2015; Laporte, 2009; Toth & Vigo, 2014), in this paper we develop a hybrid particle swarm optimization algorithm (HPSO) to solve the MCVRPTW. The HPSO algorithm integrates the process of simulated annealing (SA), in that SA has the property of accepting inferior solutions with a certain probability, which helps PSO to effectively break out of local optimum. Meanwhile, for comparison purpose, the performance is compared between the HPSO algorithm and the PSO algorithm without SA.

The remainder of this paper is organized as follows. Section 2 reviews the relevant literature. Section 3 defines the MCVRPTW and formulates a mathematical model for the problem. Section 4 describes a novel solution approach based on the HPSO algorithm for MCVRPTW, which can be simplified into a classical PSO algorithm. Section 5 presents and discusses the computational results from both algorithms. Section 6 summarizes the major findings and draws conclusions.

2. Literature survey

Unlike VRP, MCVRP has not attracted much attention until recent years. It is a combinatorial optimization problem, and developing the solution methods has been a major research challenge. Overall, the existing solution efforts can be categorized into two groups, namely, exact solution algorithms and heuristic algorithms. Relevant studies are surveyed in the following based on the classification.

2.1. Exact solution algorithms

For the combinatorial optimization problems, the optimal solution can be found within acceptable time when the problem scale is small. The exact algorithms for solving MCVRP mainly include branch-and-price, branch-and-cut, and branch-price-and-cut methods. Archetti, Campbell, and Speranza (2014) compared strategies for handling separately and jointly of distribution of MCVRP. In the joint situation, there are no-split and split problems depending on whether a customer's multiple products must be delivered by one vehicle or not. A mixed-integer linear programming model for the split problem was presented, and a branch-and-cut algorithm was developed as the solution method. Archetti, Bianchessi, and Speranza (2015) formulated a set partitioning model by making use of an exponential number of variables for the split MCVRP, developed a branch-price-and-cut solution approach. Coelho and Laporte (2015) introduced four main categories of MCVRP according to the split or no-split situation for compartments and tanks. Two mixed-integer linear programming formulations were proposed for each case, and specialized models were also suggested for some particular versions of the problem. It was indicated that a branch-and-cut algorithm is applicable to all variants of MCVRP. Mirzaei and Wøhlk (2017) presented the no-split and split versions of MCVRP, and presented a Branch-and-Price algorithm for solving to optimality and compared the optimal costs of the two versions.

2.2. Heuristic algorithms

2.2.1. MCVRP

MCVRP is NP-hard since it is a special case of the VRP, which is well known to be NP-hard. No exact algorithm can guarantee the global optimal solution within reasonable computing time for many large-scale problems. As a result, many heuristic algorithms have been proposed to find the satisfactory solution in the reasonable computation time. The heuristic algorithms for solving MCVRP include Lagrangean relaxation, memetic algorithm, ant colony algorithm, tabu search, etc. Chajakis and Guignard (2003) introduced a MCVRP related to the distribution to convenience stores. They formulated a mixed integer programming model and proposed a heuristic algorithm based on Lagrangean relaxation for solution. Fallahi, Prins, and Calvo (2008) introduced a MCVRP based on livestock feed distribution with a maximum route length, and proposed a solution approach based on memetic algorithm and tabu search. Muyldermans and Pang (2010) proposed a solution algorithm for MCVRP, which includes local search, mechanisms of neighbor lists and guided local search meta-heuristic. Derigs et al. (2011) introduced an integer program formulation of MCVRP and presented a heuristic, which covers a broad range of alternative approaches for construction, local search, large neighbourhood search. Silvestrin and Ritt (2014) formulated a MCVRP with two special constraints, namely, the total time travelled by each vehicle must not exceed a maximum time, and the demand of a customer must be serviced in one visit. A tabu search approach was developed, which features initial solution generation by a modified version of the savings method of Clarke and Wright. Reed, Yiannakou, and Evering (2014) introduced a MCVRP based on the collection of recycling waste from households, and proposed an ant colony algorithm with k-means clustering and 2-opt to solve it. Abdulkader, Gajpal, and ElMekkawy (2015) introduced a MCVRP with the constraints of maximum length of route, and proposed a hybrid ant colony algorithm and local search as the solution method. Silvestrin and Ritt (2017) proposed a tabu search heuristic and embedded it into an iterated local search to solve a MCVRP. Hübner and Ostermeier (2018) considered loading and unloading costs, and proposed a large neighborhood search (LNS) with specific removal and reinsert operators. Alinaghian and Shokouhi (2018) introduced a MCVRP with multi-depot and a solution approach of hybrid adaptive large neighborhood search was proposed. In addition to the research on MCVRP, efforts have also been made to study the extension problems of MCVRP which arise from practical applications, and they are summarized below.

2.2.2. MCVRP with flexible compartment sizes

The multi-compartment vehicle routing problem with flexible compartment sizes (MCVRPFCS) is a variant of MCVRP. If the compartment size can only be selected from a set of potential options, it is called the multi-compartment vehicle routing problem with discretely flexible compartment sizes (MCVRPDFCS). If the size can be selected arbitrarily, it is called the multi-compartment vehicle routing problem with continuously flexible compartment sizes (MCVRPCFCS). Henke, Speranza, and Wäscher (2015) introduced a MCVRPDFCS in the context of glass waste recycling in Germany in that glass of different colors

should be collected separately. A variable neighborhood search algorithm was developed as the solution approach. Meanwhile, Koch, Henke, and Wäscher (2016) introduced a MCVRPCFCS which considers the maximal number of compartments that can be used in one vehicle may be equal to or smaller than the number of product types. A genetic algorithm was proposed to solve the model. Henke, Speranza, and Wäscher (2017) introduced a new formulation and a branch-and-cut algorithm which includes subtour-elimination cuts, capacity cuts, and several valid inequalities to solve the MCVRPCFCS.

2.2.3. MCVRP with stochastic demands

Multi-compartment vehicle routing problem with stochastic demands (MCVRPSD) is another variant of MCVRP. Mendoza, Castanier, Guéret, Medaglia, and Velasco (2010) treated the MCVRPSD as a stochastic programming problem with recourse, and proposed a memetic algorithm for solution. Considering its stochastic characteristics, a novel individual evaluation and reparation strategy was introduced in the algorithm. Mendoza, Castanier, Guéret, Medaglia, and Velasco (2011) formulated MCVRPSD as a two-stage stochastic programming model, and developed three constructive heuristics including an extension scenario of a savings-based algorithm, look-ahead heuristic, and stochastic 2-Opt heuristic. Goodson (2015) developed methods to calculate the expected time of initial arrival to a particular customer on an a priori route, and described a cyclic-order simulated annealing algorithm for MCVRPSD.

2.2.4. MCVRP with time windows

MCVRP with time windows (MCVRPTW) allows each customer to designate a given time windows to be serviced. There are two categories according to the underlying assumptions: the hard time windows and the soft time windows. The former refers to the situation that the customers must be serviced in the time windows, and the latter refers to that the time windows can be violated if the penalty is paid. Melechovský (2013) proposed a selective MCVRPTW with a limited number of identical vehicles available to serve. The problem was described as a bi-objective optimization problem with one objective maximizing the collected profit and the second objective minimizing the traveled distance, and solved by a variable neighborhood search algorithm. Similar efforts on selective MCVRPTW were made by Kaabi (2016), in which a hybrid approach based on the genetic algorithm (GA) and the iterated local search (ILS) was developed as the solution approach. Also, Kabcome and Mouktonglang (2015) presented mathematical models for three cases of MCVRP, in which the first one considers multiple trips, the second one considers hard time windows, and the third one considers soft time windows. Small cases of 5, 10, and 15 customers were solved by using commercial solvers within a reasonable time. Note that all existing works on MCVRPTW are application specific and limited.

The limited existing works on MCVRPTW are application specific, and the solution approaches are not suitable for urban distribution applications. The works of Melechovský (2013) and Kaabi (2016) may lead to giving up providing services to some customers thanks to the consideration of profit maximization. However, in urban distribution, the individual terminal customers are the distribution targets. Any declined distribution service brings great inconvenience to their living. Also, from the viewpoint of distribution enterprises, doing so for profit maximization may seriously affect customer satisfaction in the long run. Meanwhile, the work of Kabcome and Mouktonglang (2015) is applicable to the small-scale problems and their solution approach is based on a commercial software. Nevertheless, the number of customers in urban distribution is often large. With the increase of problem size, more efficient solution approaches should be developed. Therefore, considering the uniqueness of urban distribution, a specific type of MCVRPTW is proposed, which tackles the optimization to reduce the distribution cost of distribution enterprises under the condition of meeting the distribution needs of each customer. To solve the optimization problem with higher number of customers, two solution approaches, including particle swarm optimization (PSO) and hybrid PSO with simulated annealing, are proposed and compared in this paper.

3. Model formulation

3.1. Problem description

The MCVRPTW for urban distribution can be defined as follows. A depot has a series of homogeneous vehicles. Each vehicle has multiple fixed size compartments. The size of the compartments can be different and each compartment is assigned to a product. Customers have demand for different type of products which must be put into different compartments. Each customer receives service only once by a vehicle. Vehicles deliver products to customers within a given time windows. The time to start service when the vehicle arrives at a customer or the depot shall be within the specified time window, and there is a certain service time of the vehicle providing service for a customer or the depot. Each vehicle starts and ends at the depot. Travel times and travel costs are symmetric. The objective is to find a set of routes visiting customers such that all customer requests are fully satisfied at minimal travel cost under vehicle compartments capacity constraints and time constraints. An example of MCVRPTW problem is given in Fig. 1. It shows a distribution system including a depot and eight customers. One vehicle is loaded with products P1, P2 and P3 in three compartments, respectively. Starting from the depot “0”, within the time window $[e_1, l_1]$, the vehicle delivers products P1, P2 and P3 to customer 1; within the time window $[e_2, l_2]$, the vehicle delivers products P1 and P2 to customer 2; and within the time window $[e_3, l_3]$, the vehicle delivers products P2 and P3 to customer 3; and then the vehicle returns to the depot “0”. Similarly, the other vehicle starts from the depot “0”, and in the corresponding time window, finishes the distribution of the corresponding products to customers 4–8, and then returns to the depot “0”.

3.2. Mathematical formulation

To facilitate reading, symbols related to sets, parameters and decision variables used in the mathematical model are expressed in a tabulated form, as shown in Table 1.

With the notations shown in Table 1, the MCVRPTW can be modeled as follows:

Minimize

$$Z = \sum_{i\in N}\sum_{j\in N} c_{ij}y_{ij} \tag{1}$$

Subject to:

$$\sum_{k\in K} x_{ik} = 1 \quad \forall\, i \in N' \tag{2}$$

$$x_{ik} \leqslant x_{0k} \;\forall\, i \in N',\, k \in K \tag{3}$$

$$\sum_{i\in N} y_{ij} = \sum_{i\in N} y_{ji} \quad \forall\, j \in N \tag{4}$$

$$\sum_{i\in N} x_{ik}y_{ij} = x_{jk} \quad \forall\, j \in N,\, k \in K \tag{5}$$

$$\sum_{i\in N'} x_{ik}d_{ip} \leqslant Q_p \quad \forall\, p \in P,\, k \in K \tag{6}$$

$$\sum_{i\in S}\sum_{j\in S} y_{ij} \leqslant |S| - 1 \quad |S| \geqslant 2,\, \forall\, S \subseteq N' \tag{7}$$

$$b_i + (s_i + t_{ij})y_{ij} - M(1 - y_{ij}) \leqslant b_j,\, \forall\, i, j \in N \tag{8}$$

$$e_i \leqslant b_i \leqslant l_i \;\forall\, i \in N \tag{9}$$

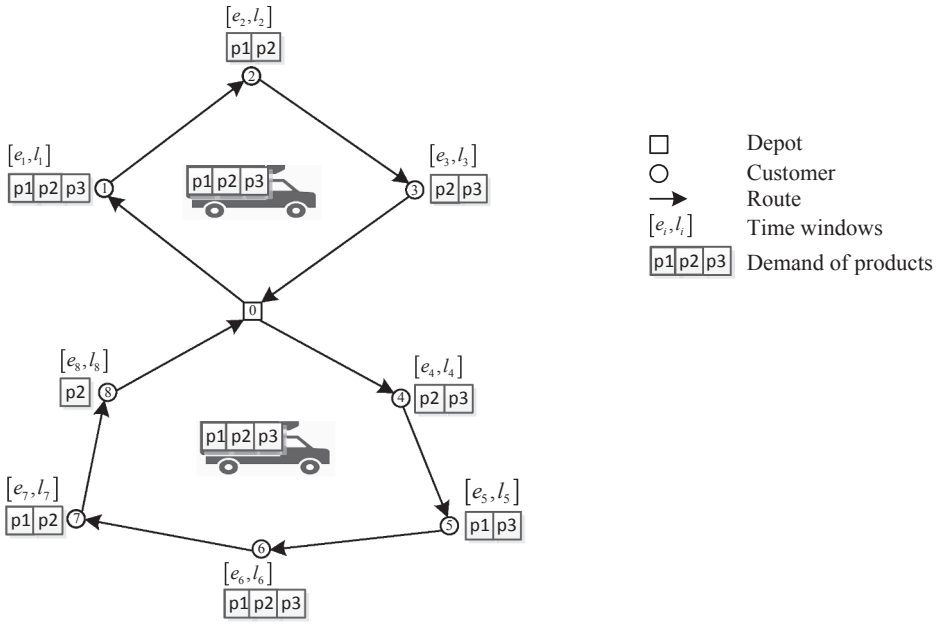

Fig. 1. Schematic of a MCVRPTW example.

Table 1
Notation of the mathematical model.

	Description
Sets	
$N = \{0, 1, 2, \cdots, n\}$	node set, 0 is the depot, and 1,2, $\cdots, n$ are customers
$N' = N \backslash \{0\}$	the set of customers
$A = \{(i, j) \mid i, j \in N, i \neq j\}$	the set of arcs
$K = \{1, 2, \cdots, k\}$	the set of homogeneous vehicles
$P = \{1, 2, \cdots, p\}$	the set of products or compartments
Parameters	
Q_p	capacity of the compartment p
d_{ip}	demand of customer i for product p
c_{ij}	travel cost of a vehicle on arc (i, j)
t_{ij}	travel time of a vehicle on arc (i, j)
e_i	the earliest beginning time of node i to be served
l_i	the latest beginning time of node i to be served
s_i	the service time of node i
b_i	the start serve time of node i
Z	Objective function value
M	sufficiently large number
Decision variables	
x_{ik}	1 if node i is served by vehicle k, 0 otherwise.
y_{ij}	1 if node i is visited right before node j, 0 otherwise.

$$x_{ik} \in \{0, 1\} \ \forall \ i \in N, k \in K \tag{10}$$

$$y_{ij} \in \{0, 1\} \ \forall \ i, j \in N \tag{11}$$

The objective function (1) minimizes the total travel cost. Constraints (2) ensure that each customer is visited by exactly one vehicle. Constraints (3) represent that if vehicle$k$visits customer$i$, it must visit the distribution center; if the vehicle does not visit node 0, it will not visit any customer. Constraints (4) ensure the continuity of each route, that is: a vehicle that visits node$j$must leave node j. Constraints (5) state that if there is a vehicle travel from node i to node j, they are visited by the same vehicle. Constraints (6) state that a compartment must not be loaded more than the capacity. Constraints (7) are classical subtour elimination constraints. Constraints (8) ensure that the starting time of the next node j has to consider the start serve time b_i plus the service time s_i of the previous node i in addition to the travel time t_{ij} in case that node i is a customer node or the depot. Among them $b_0 = 0$, $s_0 = 0$. Constraints (9) guarantees that delivery to a node must be within a given time windows. Constraints (10) and (11) define the variable domains.

In VRPTW, a vehicle delivers only one product at a time. In MCVRPTW, multi-compartment vehicles are used to transport different types of products. This can not only satisfy the customer's demand but also make full use of the vehicle space. When the numbers of compartments and products are both equal to 1, MCVRPTW is reduced to VRPTW.

4. Solution approach

Particle Swarm Optimization (PSO), proposed by Eberhart and Kennedy (1995), is an evolutionary algorithm based on the simulation of social behavior of birds within a flock. In PSO, individuals are referred to as particles in the multidimensional space. Each particle represents a feasible solution with two attributes of velocity and position. The PSO is initialized to a collection of random particles (random solutions). Then the optimal solution is found through iteration. In each iteration, the particle updates itself by tracking two "extremum" values. The first is the personal best-known solution found by the particle itself, which is called individual extreme value *pbest*. The other extreme value is the population best-known solution currently found for the whole population, and this extreme value is global extreme value *gbest*. Each particle iterates to update its attributes according to inertial behavior, individual cognitive behavior and social learning behavior. The updated equation is as follows:

$$V_{t+1} = \omega V_t + c_1 rand()(X_{pbest} - X_t) + c_2 rand()(X_{gbest} - X_t) \tag{12}$$

$$X_{t+1} = X_t + V_{t+1} \tag{13}$$

where ω is the inertial weight; c_1 and c_2 are the acceleration constants; $rand()$ is the random number (0,1); V_t is the current velocity of the particle in the tth iteration, also known as the inertial behavior; $X_t$is the current position of the particle in the tth iteration; X_{pbest} is the personal best-known solution, $X_{pbest} - X_t$ represents the individual cognitive behavior; X_{gbest} is the population best-known solution, $X_{gbest} - X_t$ represents the social learning behavior; V_{t+1} is the current velocity of the particle in the $(t + 1)^{th}$ iteration; $X_{t+1}$is the current position of the particle in the $(t + 1)^{th}$ iteration.

The initial position and velocity of particle swarm are generated randomly, and then iterate according to Eqs. (12) and (13) until a

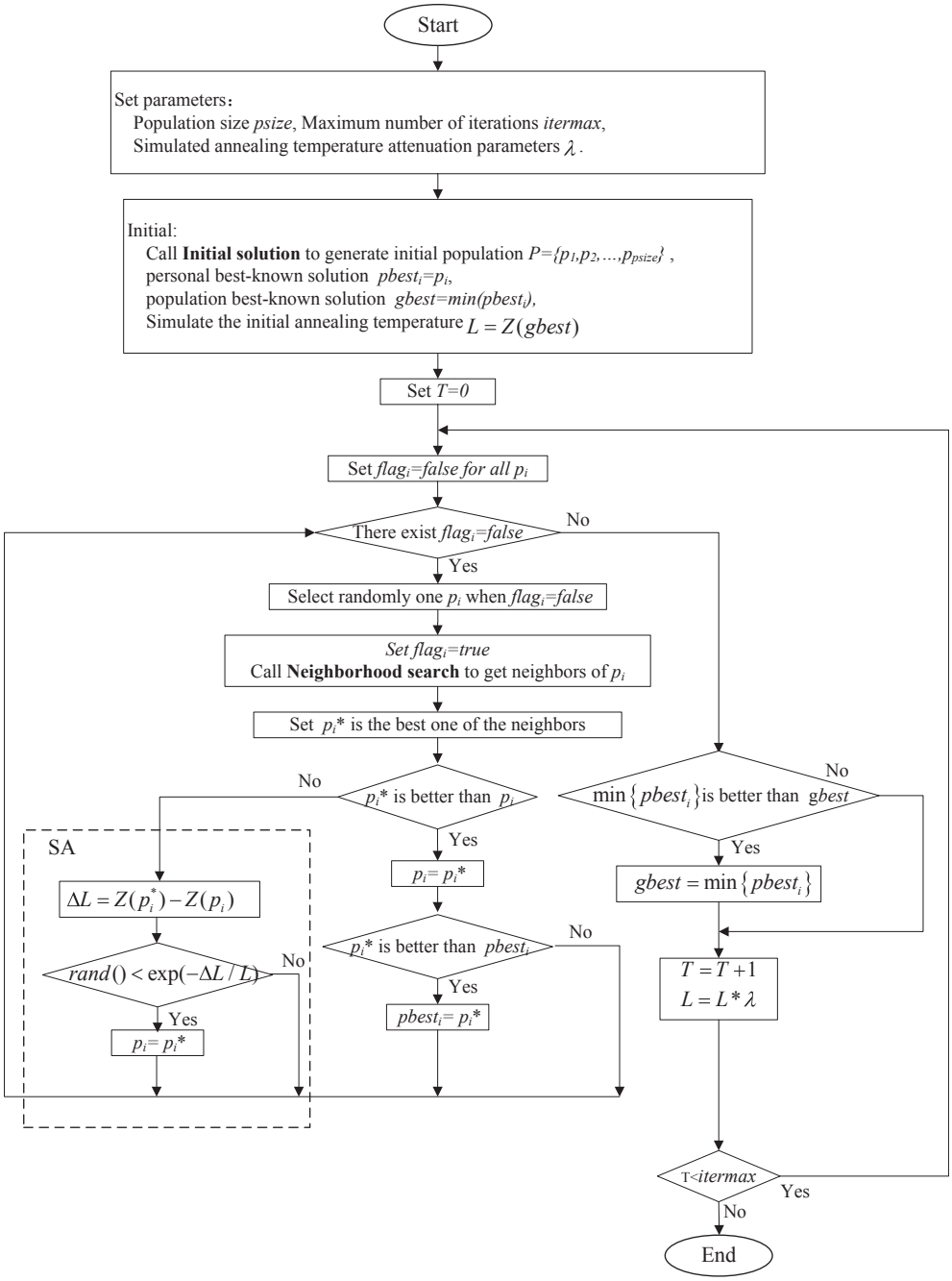

Fig. 2. HPSO flowchart.

satisfactory solution is found. According to the problem description, it would be challenging for solving MCVRPTW by the basic PSO algorithm based on an updating equation in that the MCVRPTW problem is a discrete optimization problem while the basic PSO algorithm based on an updating equation is usually effective for continuous optimization problems. Inspired by the idea of local search and path relinking strategy, this paper applies local search for the current particle $pcurrent_i$ to express the influence of the inertial behavior, applies the path relinking from $pcurrent_i$ to its personal best-known solution $pbest_i$ to express the influence of individual cognitive behavior, and applies the path relinking from $pcurrent_i$ to the population best-known solution $gbest$ to express the influence of the social learning behavior. In addition, considering that the particle swarm algorithm has a shortage of premature convergence, simulated annealing is integrated into the solution approach to help PSO jump out of local optimum. This approach is then called a hybrid particle swarm optimization (HPSO). The flowchart of HPSO is shown in Fig. 2.

In the HPSO flow chart shown in Fig. 2, if the SA related components are removed, the hybrid solution approach becomes a PSO algorithm. The components include the dotted SA block in Fig. 2, as well as the parameter setting for SA temperature cooling parameter λ and the initialization for annealing temperature L. After each iteration, gradual attenuation of L is expressed as λL. The key difference between PSO and HPSO executions is as follows. In PSO, when the new solution p_i^* is worse than the current solution p_i, the dotted SA block is directly skipped to the next solution whose $flag_i = true$ and the process continues. On the other hand, the HPSO algorithm will execute the dotted SA block when the new solution p_i^* is worse than the current solution p_i.

4.1. Initial solution

Considering the quality and diversity, the initial solutions are generated by random scanning and greedy algorithm. The process of generating initial solution by random scanning is as follows: The polar coordinate is used to indicate the customers. The depot is the origin of polar coordinates. There are two steps for generating the initial solution. In the first step, a customer is randomly selected to produce a ray from the polar coordinates and through the customer. At the same time, a route from the depot through the customer and back to the depot is created. In the second step, the ray scans clockwise to meet a customer. If this customer can be successfully inserted into the current route by the greedy algorithm under the constraints of capacity and time window, the current route is updated. Otherwise, a new route from the depot through this customer and back to the depot is created. The second step loops until all the customers are on routes.

4.2. Neighborhood search

Based on the initial solutions, neighborhood search is employed to find neighbor solutions. Local searches are applied in a certain solution, and path relinking is applied between two different solutions. Local search includes swap, move, and 2-opt based on the current solution. The process of generating the neighborhood solution is as follows: randomly select from the three local searches to generate a solution; personal best-known solution and population known optimal solution as the guidance are used as the guide solution in the process of path relinking for current solution to generate two solutions. The best solution of the generated solution is selected as the neighborhood solution of the current solution. Three types of local search and path relinking are detailed below.

4.2.1. Local search

In order to search around individuals, three operators including customer swap, customer move and 2-opt are introduced. Two routes are given for the convenience of illustrating the following example. Two routes called $R1$ ($0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 0$) and $R2$ ($0 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 0$). In the route, “0” represents the depot, “1, 2, 3, …7” represent the customers. The check of the vehicle capacity constraints and time windows of customers mentioned below are based on constraints (6) and (8) respectively. The check of vehicle capacity is to determine whether the total demand of the customers in the new route exceeds the capacity of vehicle or not. However, the check of time windows is more complicated. As long as there is a node changed in the original route, the vehicle arrival time of each node after that node will be changed, and that may lead to an infeasible solution. Therefore, it is necessary to check whether the vehicle arriving time at each node is before the upper limit of the time window of that node or not.

(1) Swap

Swap operator is to exchange two customers randomly selected from the same route or from two different routes. The main steps of customer swap include selecting two customers, checking constraints and swap customers to obtain a neighborhood solution. For example, suppose customer 1 and customer 3 are selected from $r1$ to swap. $r1$ is changed into $r1'$ ($0 \rightarrow 3 \rightarrow 2 \rightarrow 1 \rightarrow 4 \rightarrow 0$). Since the chosen customer 1 and customer 3 are from the same route, the customer has not changed on the path, the capacity of the vehicle is unchanged, so there is no need to check vehicle capacity constraints of $r1'$. On the other hand, since the order of vehicle visits customers on $r1'$ is different from that on $r1$, the arrival time of vehicle for customers has been changed, so the time windows constraints must be checked for customers in $r1'$.

(2) Move

Move operator is to move a randomly selected customer from one position to another in routes. These two positions may be in the same route or in the different routes. The main steps of customer move include selecting the customer to be moved out from the current position and finding the position to insert the moved customer into. The selected customer can be inserted before or after another customer. Select a better position of the target function to insert. The position of the insertion is determined by inserting the selected customer after or before another customer - whichever gives the better objective function value will be chosen. For example, suppose customer 2 is selected to move and another customer 4 is selected. If customer 2 moving after customer 4 is better than moving before, $r1$ is changed into $r1'$ ($0 \rightarrow 1 \rightarrow 3 \rightarrow 4 \rightarrow 2 \rightarrow 0$). Also, suppose customer 2 is selected to move and customer 5 is selected. If customer 2 moving before Customer 5 better than moving after, $r1$ and $r2$ are changed into $r1'$ ($0 \rightarrow 1 \rightarrow 3 \rightarrow 4 \rightarrow 0$) and $r2'$ ($0 \rightarrow 2 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 0$). Capacity and time windows constraints have to be checked.

(3) 2-opt

2-opt was first introduced for solving the Travel Salesman Problem (Croes, 1958). Nowadays, 2-opt is widely used to solve VRP and its extended problems. In this paper, 2-opt is designed to disconnect any two non-adjacent edges and make the best interconnection. Two non-adjacent edges may be chosen from the same route or different routes. For example, suppose the edges ($0 \rightarrow 1$) and ($3 \rightarrow 4$) in $r1$ are selected to disconnect. If the reconnecting of ($0 \rightarrow 3$) and ($1 \rightarrow 4$) is the best feasible reconnecting, $r1$ is changed into $r1'$ ($0 \rightarrow 3 \rightarrow 2 \rightarrow 1 \rightarrow 4 \rightarrow 0$). In another example, suppose the edge ($2 \rightarrow 3$) in $r1$ and the edge ($6 \rightarrow 7$) in $r2$ are selected to disconnect. If the reconnecting of ($2 \rightarrow 7$) and ($6 \rightarrow 3$) is the best feasible reconnecting, $r1$ and $r2$ are changed into $r1'$ ($0 \rightarrow 1 \rightarrow 2 \rightarrow 7 \rightarrow 0$) and $r2'$ ($0 \rightarrow 5 \rightarrow 6 \rightarrow 3 \rightarrow 4 \rightarrow 0$). Relative constraints must be checked similarly as customer swap.

4.2.2. Path relinking

Path relinking (PR) (Glover, Laguna, & Martí, 2000) is an efficient heuristic search strategy for rapidly obtaining a new solution by establishing the relinking path between the current solution and the guide solution. The guide solution is chosen from some better solutions. The new solution contains some properties of the guide solution and the current solution.

In the algorithm of this paper, path relinking based on route is proposed. That means a route will be chosen from the guide solution in the relinking process. The new solution including the chosen route will be obtained from the current solution. Two steps would be fulfilled in path relinking. The first step is deleting all customers in the chosen route from the current solution. The next step is obtaining the new solution by adding the chosen route. In order to clarify path relinking based on route, an illustrative example can be seen in Fig. 3.

In the example, a distribution system including a depot with 20 customers is shown. The route is represented by r, where ri means the ith route of the solution. The current solution is shown in Fig. 3(a), and the guidance solution is shown in Fig. 3(b). Suppose the chosen route from the guidance solution is $r2$. According to the above instructions, the first step is deleting customers 1, 16, 8, 3, 19, 20, and 4 from the current solution, as shown in Fig. 3(c). The next step is obtaining the new solution by adding $r2$ ($0 \rightarrow 1 \rightarrow 16 \rightarrow 8 \rightarrow 3 \rightarrow 19 \rightarrow 20 \rightarrow 4 \rightarrow 0$), as shown in Fig. 3(d).

The first step is to do the delete operation on the existing path, without the need for capacity and time window constraints. In the second step, because it is directly added to the selected path in the directed solution, the customer point in this path has no change, and

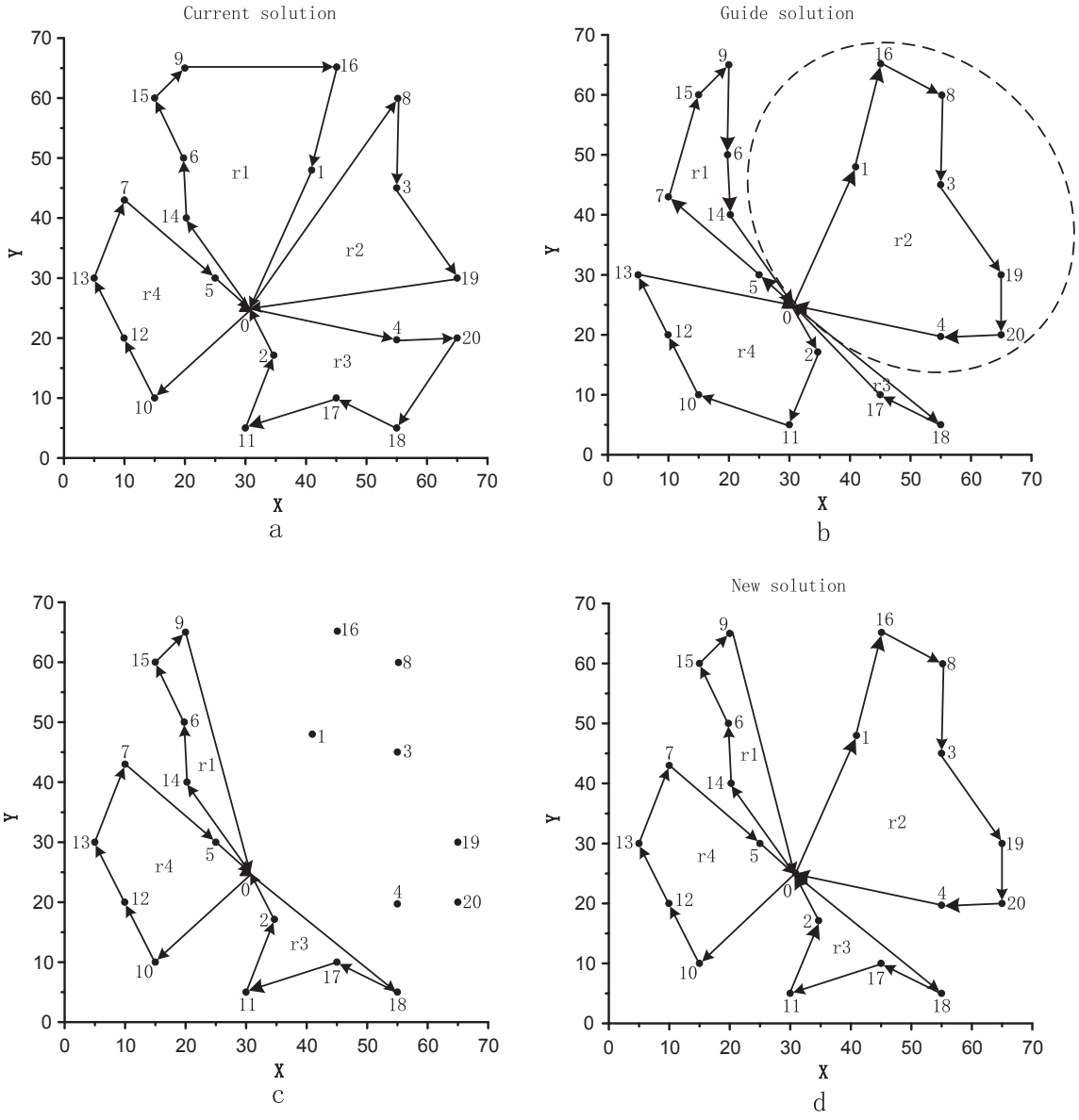

Fig. 3. Example for path relinking based on route.

the capacity and time window constraints are not required.

4.3. Simulated annealing

Simulated annealing (SA) was first proposed by Metropolis in 1953. It is a random optimization algorithm based on the Monte-Carlo iterative solution strategy. Its starting point is the similarity between the annealing process of physical solid matters and the general combinatorial optimization problem. The simulated annealing algorithm starts from a high initial temperature. With the continuous decrease of temperature parameters, it combines with the probability jump to find the global optimal solution of the objective function randomly in the solution space. That is, the local optimal solution is able to jump out probabilistically and ultimately tends to the global optimal.

The basic steps of simulated annealing are as follows:

(1) initialization: initial temperature $L = Z(gbest)$, temperature cooling parameters $\lambda < 1$, initial solution p, the number of iterations T.
(2) for $k = 1, 2, \cdots, T$, do steps (3)–(6).
(3) generate new solution p^*.
(4) calculate $\Delta L = Z(p^*) - Z(p)$, in which $Z(p)$ is the evaluation function
(5) if $\Delta L < 0$,then accept p^* as the new current solution, else, generate a random number $rand() \in [0, 1]$; if $rand() < \exp(-\Delta L/L)$, then accept p^* as the new solution, else don't accept p^*as the new current solution.
(6) if the termination condition is met, output the current solution as the optimal solution and terminate the program.
(7) $L = \lambda L$, then go to step 2.

5. Numerical experiments

In order to analyze the performance of the proposed HPSO approach and its simpler version (i.e., PSO) for MCVRPTW, we implement the algorithms using Microsoft Visual Studio 2017. A computer with Intel (R) Core(TM) i7-6500U CPU is adopted to run the code for all instances, and the operating system is Windows 10. By first referring to relevant literature (Babaee et al., 2016; Birim, 2016; Fathi, Rodríguez, Fontes, & Alvarez, 2016; Marinakis, Migdalas, & Sifaleras, 2017; Wang, Jagannathan, Zuo, & Murray, 2017; Wang, Lu, Wei, Ji, & Yang, 2016; Zhang, Yang, & Weng, 2018) and then conducting preliminary experiments, the algorithm parameters, which provide good efficiency and

accuracy, are determined as follows: the total number of iterations is set to 1000, the population of the particle swarm is set to 50, initial temperature is set to 100, the number of iterations and the temperature cooling parameter of simulated annealing is set to 10 and 0.9, respectively. The results are obtained with 10 random runs.

5.1. Test instances

Note that there are no internationally recognized instances for MCVRPTW. Therefore, we adopt the method proposed by Reed (2014) to generate the MCVRPTW instances based on the VRPTW instances of Solomon (1987). The data files of the VRPTW instances are available online at: http://neo.lcc.uma.es/vrp/vrp-instances/capacitated-vrp-with-time-windows-instances/. Those VRPTW instances have three set instances involving 25, 50, and 100 customers, respectively. For convenience, the following instances of 25 customers are referred to as small scale instances, 50 customers are referred to as medium scale instances, and 100 referred to as customers are large scale instances. According to the customer position distribution, instances of each set are divided into clustered distribution C classes, scattered distribution R classes, and partial scattered and partial clustered distribution RC classes. According to the customer time windows size, each class is further subdivided into the narrower time window "1" classes, and the wider time window "2" classes. Therefore, Solomon's instances are subdivided into the C1/C2/R1/R2/RC1/RC2 six classes. For simplicity, the following results show only the first four instances of each class of each customer size. A set of instances are generated for MCVRPTW. The VRPTW instances are adapted into MCVRPTW instances according to the method developed by (Reed, 2014), in which the data is obtained by splitting the vehicle capacity into two compartments using a 3:1 ratio, while the customer demands are obtained using a 2:1 ratio, except that the demands on the sub-region ($x_{\min} \leqslant x < x_{\min} + \frac{x_{\max} - x_{\min}}{2}$, $y_{\min} \leqslant y < y_{\min} + \frac{y_{\max} - y_{\min}}{2}$), are split using a 3:1 ratio. In the representation of the sub-region, x and y represent the horizontal and vertical coordinates of the customer position, $x_{\max}$ and $y_{\max}$ represent the maxima of the horizontal and vertical coordinates, $x_{\min}$and $y_{\min}$ represent the minima of its horizontal and vertical coordinates.

5.2. Computational results

The detailed results obtained by the PSO and HPSO algorithms are reported in Tables 2, 3 and 4, which summarize the performance metrics of the 25, 50, and 100-customer instances. In the tables, columns "Best" and "Worst" present the objective function values of the best solution and the worst solution of 10 random runs, columns "Avg." and "Std. dev" are the average and the standard deviation of the objective function value of 10 random runs, and column "Time(s)" is the average time of complete 1000 iterations for each run.

Table 2 shows that both PSO and HPSO can produce relatively satisfactory solutions in a relatively short time for the small-scale instances of 25 customers. Especially for C101/C201/C202/ RC201/ RC204, two algorithms give the same best solution by random run of 10 times. From the point of view of the best solution, two algorithms produce the same solution for 17 of the 24 instances, and HPSO outperforms PSO for the remaining 7 instances. From the point of view of the worst solution, 9 of the 24 instances have the same solution, and HPSO outperform PSO for 8 of the remaining 15 instances. Regarding the average value of the target function value, 5 of the 24 instances show the same results, and HPSO outperforms PSO for the remaining 19 instances. In terms of the standard deviations of the objective function values, they are equal for 5 of the 24 instances, and HPSO produces smaller values than PSO only for 7 of remaining 19 instances. In addition, when the average calculation time is compared, HPSO is more efficient than PSO only for 5 of the 24 instances. As a result, for small-scale instances with 25 customers, HPSO has the advantage over PSO in terms of the best solution and the average value of the objective function, but not the standard deviation and solution time.

Table 3 shows that, although the number of customer points in the instances is doubled from 25 to 50, both the PSO and HPSO algorithms can obtain relatively satisfactory solutions in a relatively short time. For instances C101/C201/C204/R202/RC201, the two algorithms develop the same best solution, while for 14 of 19 remaining instances, HPSO shows better results than PSO. However, for the worst solution values obtained, HPSO shows a worse performance than PSO for 13 instances. Meanwhile, there are 14 instances in which the mean value of objective function from HPSO is better than that of PSO, while PSO shows the

Table 2
Detailed comparison between PSO and HPSO for instances of 25 customers.

Instance	PSO					HPSO					HPSO-PSO				
	Best	Worst	Avg.	Std. dev	Time(s)	Best	Worst	Avg.	Std. dev	Time(s)	Best	Worst	Avg.	Std. dev	Time(s)
C101	191.81	191.81	191.81	0.00	245.24	191.81	191.81	191.81	0.00	318.58	0.00	0.00	0.00	0.00	73.34
C102	190.74	191.81	191.06	0.84	307.14	190.74	191.81	190.95	0.30	328.62	0.00	0.00	−0.11	−0.54	21.48
C103	190.74	198.42	194.81	2.17	209.30	190.74	198.42	194.22	2.56	342.81	0.00	0.00	−0.60	0.39	133.51
C104	190.74	191.92	190.97	0.31	278.24	190.74	195.22	191.48	1.56	225.59	0.00	3.30	0.51	1.26	−52.66
C201	215.54	215.54	215.54	0.00	176.75	215.54	215.54	215.54	0.00	324.96	0.00	0.00	0.00	0.00	148.21
C202	223.31	223.31	223.31	0.00	153.29	223.31	223.31	223.31	0.00	277.50	0.00	0.00	0.00	0.00	124.21
C203	223.31	223.31	223.31	0.00	163.82	223.31	224.97	223.48	0.15	294.32	0.00	1.66	0.17	0.15	130.49
C204	223.35	232.63	225.12	3.37	162.28	223.35	232.83	225.92	3.78	300.46	0.00	0.20	0.81	0.41	138.19
R101	618.33	627.13	621.91	3.92	305.59	618.33	619.17	618.41	0.13	330.14	0.00	−7.96	−3.50	−3.79	24.55
R102	548.11	551.93	549.14	1.15	272.03	548.11	557.12	549.01	2.68	308.24	0.00	5.19	−0.13	1.53	36.21
R103	455.70	464.83	460.26	4.48	257.46	455.70	466.05	459.47	4.66	279.81	0.00	1.22	−0.79	0.18	22.36
R104	417.96	423.49	419.07	2.50	224.15	417.96	441.68	424.62	9.64	253.40	0.00	18.19	5.55	7.14	29.25
R201	506.52	533.91	524.79	8.27	205.84	464.38	529.78	490.34	21.21	245.64	−42.14	−4.13	−34.46	12.94	39.80
R202	415.31	444.87	424.73	7.16	200.16	412.18	429.05	417.17	6.06	236.51	−3.13	−15.82	−7.56	−1.10	36.34
R203	403.40	427.27	415.56	7.74	195.38	394.70	430.10	408.63	10.84	224.67	−8.70	2.83	−6.92	3.10	29.29
R204	367.49	389.44	371.79	7.22	181.07	360.48	374.68	366.59	4.55	212.87	−7.01	−14.76	−5.20	−2.67	31.79
RC101	515.03	537.49	534.16	6.65	306.23	462.16	476.96	474.00	5.64	296.59	−52.87	−60.53	−60.16	−1.01	−9.64
RC102	451.54	451.74	451.56	0.18	318.52	401.79	451.54	436.62	22.91	289.17	−49.75	−0.20	−14.94	22.73	−29.35
RC103	388.17	389.65	388.46	0.15	290.98	388.17	389.65	388.32	0.10	250.52	0.00	0.00	−0.15	−0.05	−40.46
RC104	362.36	362.67	362.40	0.12	291.07	361.61	362.36	361.91	0.29	261.76	−0.75	−0.30	−0.48	0.16	−29.31
RC201	361.24	361.24	361.24	0.00	251.87	361.24	361.24	361.24	0.00	331.90	0.00	0.00	0.00	0.00	80.03
RC202	376.12	413.40	379.85	11.07	221.80	376.12	413.40	383.57	14.79	268.60	0.00	0.00	3.72	3.72	46.80
RC203	328.44	366.71	352.72	14.40	259.02	328.44	360.09	350.01	11.96	292.84	0.00	−6.62	−2.71	−2.44	33.81
RC204	329.89	329.89	329.89	0.00	170.49	329.89	329.89	329.89	0.00	251.46	0.00	0.00	0.00	0.00	80.97

Table 3
Detailed comparison between PSO and HPSO for instances of 50 customer nodes.

Instance	PSO					HPSO					HPSO-PSO				
	Best	Worst	Avg.	Std. dev	Time(s)	Best	Worst	Avg.	Std. dev	Time(s)	Best	Worst	Avg.	Std. dev	Time(s)
C101	418.42	418.42	418.42	0.00	364.35	418.42	419.19	418.50	0.16	423.44	0.00	0.77	0.08	0.16	59.10
C102	418.11	430.02	421.33	4.78	531.23	417.34	431.41	420.58	5.01	431.88	−0.77	1.39	−0.75	0.23	−99.36
C103	416.88	434.63	420.46	5.57	519.17	416.06	442.31	424.93	9.82	366.97	−0.82	7.67	4.46	4.26	−152.20
C104	360.51	427.72	390.95	26.57	505.19	360.83	461.17	408.86	27.25	497.35	0.32	33.45	17.90	0.68	−7.84
C201	373.55	388.14	379.90	6.77	477.27	373.55	388.14	376.92	5.73	299.21	0.00	0.00	−2.98	−1.04	−178.05
C202	388.52	441.26	418.05	24.35	433.89	366.78	388.52	372.64	8.18	285.46	−21.74	−52.74	−45.40	−16.17	−148.43
C203	366.82	374.18	370.93	2.98	435.99	371.81	376.90	372.56	1.53	286.56	4.99	2.72	1.63	−1.46	−149.43
C204	365.86	411.77	379.21	12.64	430.57	365.86	401.50	375.24	10.49	283.64	0.00	−10.27	−3.97	−2.15	−146.93
R101	1048.04	1057.54	1052.98	3.08	593.75	1046.70	1066.76	1053.24	5.32	478.43	−1.34	9.22	0.26	2.23	−115.33
R102	912.24	924.37	916.17	3.82	641.80	911.44	997.17	923.96	25.57	443.21	−0.80	72.80	7.79	21.75	−198.59
R103	783.45	798.88	786.35	4.36	519.69	775.65	820.08	784.89	12.41	407.66	−7.80	21.20	−1.46	8.05	−112.03
R104	651.45	657.72	653.48	2.01	189.83	642.13	661.44	647.68	7.66	410.39	−9.32	3.73	−5.80	5.65	220.57
R201	802.07	845.03	816.50	10.88	391.37	808.53	854.37	823.74	12.92	389.26	6.46	9.34	7.24	2.04	−2.11
R202	714.19	772.25	735.30	14.52	385.42	714.19	771.23	736.59	18.50	344.50	0.00	−1.01	1.29	3.98	−40.92
R203	615.08	673.72	647.06	21.18	354.84	620.59	675.02	639.52	16.54	452.96	5.51	1.30	−7.53	−4.64	98.12
R204	511.40	546.25	519.35	12.91	315.28	515.12	544.80	527.90	10.69	390.42	3.72	−1.45	8.55	−2.22	75.14
RC101	968.80	982.97	976.81	3.44	615.51	958.59	973.79	965.57	6.03	440.11	−10.21	−9.18	−11.24	2.58	−175.40
RC102	887.07	904.79	894.27	4.84	545.13	886.47	901.82	891.44	4.66	422.86	−0.60	−2.96	−2.84	−0.18	−122.27
RC103	833.68	844.12	836.85	3.74	535.04	823.98	972.06	846.46	42.64	406.89	−9.69	127.94	9.61	38.90	−128.15
RC104	646.09	688.60	675.40	10.78	371.55	639.28	719.64	672.29	21.04	376.56	−6.81	31.04	−3.11	10.26	5.01
RC201	686.31	764.68	725.55	39.00	493.88	686.31	756.72	693.35	21.00	340.85	0.00	−7.97	−32.20	−18.00	−153.04
RC202	684.20	781.68	708.43	28.90	522.23	615.04	725.06	643.61	35.44	328.94	−69.16	−56.62	−64.81	6.55	−193.29
RC203	596.01	696.43	650.48	43.59	399.84	559.68	675.21	600.77	33.79	319.58	−36.33	−21.21	−49.71	−9.80	−80.26
RC204	516.81	523.66	521.38	2.22	331.35	471.82	521.21	481.36	19.18	300.20	−45.00	−2.45	−40.02	16.95	−31.16

Table 4
Detailed comparison between PSO and HPSO for instances of 100 customer nodes.

Instance	PSO					HPSO					HPSO-PSO				
	Best	Worst	Avg.	Std. dev	Time(s)	Best	Worst	Avg.	Std. dev	Time(s)	Best	Worst	Avg.	Std. dev	Time(s)
C101	956.49	980.57	968.11	8.50	1011.05	944.32	1008.16	960.38	18.31	805.29	−12.18	27.59	−7.73	9.81	−205.76
C102	952.65	981.06	962.20	10.38	980.63	952.65	1014.79	968.78	16.03	753.60	0.00	33.73	6.58	5.66	−227.03
C103	963.26	1017.97	972.23	16.35	957.16	963.26	1046.89	979.34	25.90	732.16	0.00	28.92	7.11	9.54	−225.00
C104	941.07	966.45	948.97	7.13	912.60	934.46	978.39	947.91	13.78	702.99	−6.61	11.95	−1.05	6.65	−209.61
C201	609.22	610.89	609.39	0.12	822.80	609.22	610.89	609.73	0.23	657.06	0.00	0.00	0.33	0.11	−165.74
C202	640.24	675.27	646.58	12.69	826.13	591.56	672.37	634.30	27.96	639.72	−48.68	−2.90	−12.28	15.27	−186.41
C203	630.99	736.57	662.20	34.91	776.06	605.21	748.91	665.37	42.40	626.05	−25.78	12.34	3.17	7.49	−150.00
C204	671.22	712.90	687.72	13.59	758.56	666.75	762.20	687.81	29.86	599.68	−4.47	49.30	0.09	16.28	−158.88
R101	1660.63	1715.88	1684.46	18.79	1272.79	1692.29	1916.84	1733.48	64.62	1032.37	31.66	200.96	49.02	45.82	−240.42
R102	1526.03	1569.85	1535.68	13.23	1205.03	1541.15	1796.69	1574.51	76.13	973.87	15.12	226.84	38.83	62.90	−231.16
R103	1288.22	1314.70	1295.57	7.24	1087.65	1255.60	1421.54	1316.38	54.03	866.33	−32.62	106.84	20.81	46.79	−221.32
R104	1052.25	1058.86	1055.96	2.41	994.93	1036.32	1093.27	1049.42	19.29	799.24	−15.93	34.41	−6.54	16.88	−195.69
R201	1175.71	1254.02	1211.01	19.91	843.12	1191.36	1282.87	1211.55	27.18	679.68	15.65	28.85	0.54	7.28	−163.44
R202	1106.84	1189.07	1124.96	23.06	817.23	1084.14	1160.48	1104.07	22.31	675.69	−22.70	−28.59	−20.89	−0.75	−141.54
R203	921.28	947.90	927.49	8.11	781.06	902.62	993.83	920.46	26.32	641.54	−18.66	45.93	−7.04	18.21	−139.52
R204	782.75	855.47	823.57	24.49	768.46	797.09	858.06	809.45	18.25	619.19	14.34	2.59	−14.12	−6.24	−149.27
RC101	1711.29	1823.66	1745.00	37.12	877.45	1702.69	1817.71	1734.31	32.62	957.71	−8.60	−5.95	−10.69	−4.50	80.26
RC102	1572.96	1655.62	1596.55	20.73	914.41	1581.29	1762.09	1635.07	47.29	921.41	8.33	106.47	38.52	26.57	7.00
RC103	1363.25	1497.43	1430.34	34.38	756.43	1376.89	1594.12	1445.86	62.37	850.37	13.64	96.69	15.52	27.99	93.94
RC104	1244.43	1336.93	1286.16	29.79	701.04	1229.81	1347.32	1268.14	37.26	789.13	−14.62	10.39	−18.02	7.47	88.09
RC201	1311.41	1463.03	1359.92	41.07	810.28	1282.35	1519.85	1344.57	80.19	655.43	−29.06	56.82	−15.35	39.12	−154.85
RC202	1163.48	1327.96	1204.64	50.26	620.58	1108.01	1292.33	1150.00	55.70	768.26	−55.47	−35.63	−54.64	5.45	147.68
RC203	980.71	1135.07	1031.19	50.80	626.83	1014.43	1162.27	1040.02	42.56	661.75	33.72	27.20	8.83	−8.24	34.93
RC204	854.84	897.54	871.27	11.97	738.65	834.73	917.86	851.42	23.96	568.79	−20.11	20.32	−19.85	11.99	−169.86

upper hand in the remaining 10 instances. With regards to the standard deviation of objective function value, HPSO has lower values for 9 instances and higher values for 15 instances compared with PSO. In addition, in terms of the average solution time, HPSO outperforms PSO in 19 instances. In general, for the medium scale instances with 50 customers, HPSO has advantages over PSO in the best solution, the average of objective function value, and the solution time.

Table 4 shows that for large-scale instances of 100 customers, the two algorithms can still achieve relatively satisfactory solution within a reasonable time. With the same best solutions on three instances, the best solutions of HPSO are superior to those of PSO for 14 out of the remaining 21 instances. Nevertheless, PSO shows the upper hand in generating the lower values of the worst solution in 19 instances. In terms of the average value of the objective function, HPSO and PSO are tied, with each claiming victory in 12 instances. With regards to the standard deviation of objective function value, HPSO shows tighter values than PSO only in 4 instances. However, in terms of solving time, HPSO are more efficient than PSO in 19 instances. Therefore, for large-scale cases, HPSO has overall advantage in terms of the best solution obtained and the average solution time, but it still needs to be improved in terms of the worst solution and the standard deviation of the objective function values.

Table 5
Comparison of PSO and HPSO for 17 instances with the same best solutions.

Instance	PSO			HPSO		
	#B	Total iterations	Average iteration	#B	Total iterations	Average iteration
C101	10	10	1	10	10	1
C102	7	240	34	8	266	33
C103	1	625	625	3	883	294
C104	8	2623	328	6	1166	194
C201	10	10	1	10	43	4
C202	10	10	1	10	10	1
C203	10	2991	299	9	1876	208
C204	7	546	78	7	730	104
R101	3	1298	433	9	2009	223
R102	7	2682	383	9	2403	267
R103	5	876	175	6	2434	406
R104	8	2311	289	6	1069	178
RC103	8	1699	212	9	816	91
RC201	10	232	23	10	311	31
RC202	9	345	38	8	191	24
RC203	1	106	106	1	519	519
RC204	10	593	59	10	556	56
Avg.	7		182	8		155

Moreover, 17 instances with the same best solution in 25 customers are taken to further analyze the solution effect of PSO and HPSO, as shown in Table 5. Column "#B" represents the number of runs to obtain the best solution in 10 random runs, and Column "total iterations" represents the sum of the first iteration when the best solution is obtained in each run of "#B". Column "average iteration" is calculated to obtain the averages iteration to get the best solution of the instance. Table 5 shows that, for these 17 instances, although both PSO and HPSO develop the same best solution in 10 random runs, HPSO obtains the best solution in 8 out of 10 runs, while PSO does it in 7 out of 10 runs on average. Also, on average PSO finds the first best solution in 182 iterations, while HPSO finds it using 155 iterations. As such, for the instances with the same best solution, HPSO is still superior to PSO in terms of the number of runs in which the best solution is obtained and the average iterations to obtain the best solution. Note that from Tables 3 and 4, it is obvious that for 50 customer instances and 100 customer instances, the best solution of HPSO is overall much better than that of PSO. As such, no further comparison is made for the small number of instances with the same best solution for both 50-customer and 100-customer cases.

To further analyze whether the relative merits of the two algorithms are affected by the number of customer, the geographical position distribution type of customers, and the width of the time window of customers, the averages of the objective function values and solution time are calculated respectively for the 6 main classes of C1/C2/R1/R2/RC1/RC2 at different levels of customer numbers. The "HPSO-PSO" differences in the calculated average values between the HPSO and PSO algorithms are reported in Figs. 4 and 5. A negative value indicates that HPSO is more effective or efficient, and vice versa.

Fig. 4 illustrates the difference in the averaged objective function values between HPSO and PSO. It can be seen that for the small-scale instances of 25 customers, the average objective function values obtained by HPSO are all equal to or better than those obtained by PSO regardless of the geographical position distribution type of customers, and the width of customer time window. On the other hand, the effectiveness of HPSO and PSO trades winning and losing for both medium-scale instances of 50 customers and large-scale instances of 100 customers. For the medium-scale instances of 50 customers, the average objective function values of C2/RC1/RC2 obtained by HPSO are smaller than those obtained by PSO, but the opposite is observed for classes C1/R1/R2. Also, for the large-scale instances of 100 customers, the average objective function values of C2/R2/RC2 obtained by HPSO are smaller than those obtained by PSO, but the opposite is observed for

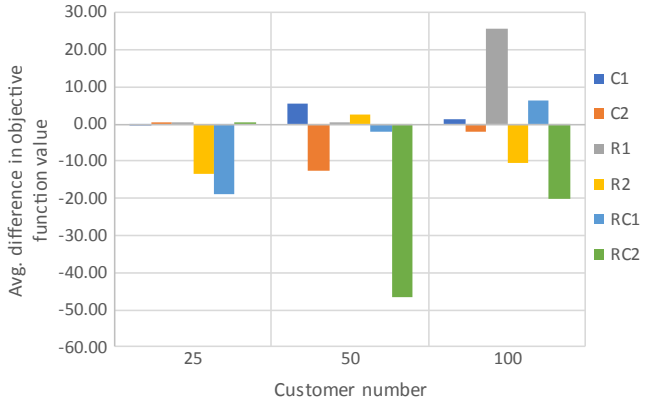

Fig. 4. Average difference in solution quality between HPSO and PSO for instances of 25, 50 and 100 customers.

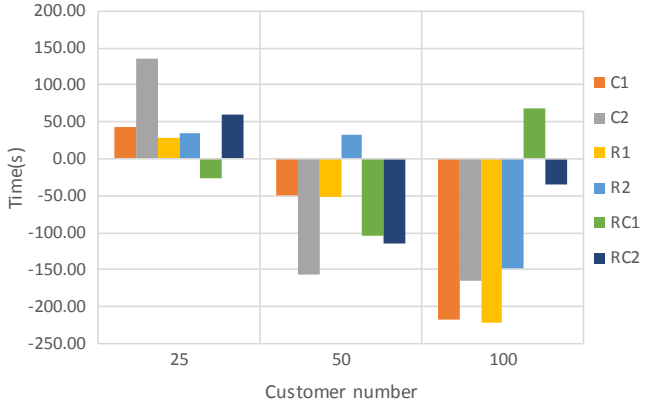

Fig. 5. Average difference in solution time between HPSO and PSO for instances of 25, 50 and 100 customers.

classes C1/R1/RC1. Meanwhile, the pairwise comparisons between C1/C2, R1/R2, and RC1/RC2 reveal the effect of the width of customer time window. It is observed that HPSO is more effective than PSO when the wider time window (i.e., class "2") is involved, with the exception of RC1/RC2 at the level of 25 customers and R1/R2 at the level of 50 customers. In addition, as far as the geographical position distribution type of customers is concerned, the performance difference between HPSO and PSO cannot be generalized.

Similarly, Fig. 5 shows the difference in the average solution time between HPSO and PSO. It can be seen that PSO is generally more efficient than HPSO for the small-scale instances of 25 customers, but the trend reverses for the medium-scale and large-scale instances. More importantly, HPSO becomes increasingly more efficient than PSO as the customer number increases. Meanwhile, the pairwise comparisons between C1/C2, R1/R2, and RC1/RC2 does not reveal a general trend regarding the effect of the width of customer time window on computational efficiency comparison between HPSO and PSO. The exception is at the level of 25 customers, in which HPSO appears to become less efficient when the wider time window (i.e., class "2") is involved. In addition, no particular trends can be observed regarding the effect of geographical position distribution type of customers.

6. Conclusions

In this paper, the multi-compartment vehicle routing problem with time window (MCVRPTW) for urban distribution is proposed. This intriguing MCVRPTW is an extension of the classical vehicle routing problem with time window (VRPTW) where different products are transported together in one vehicle with multiple compartments. Based

on the mathematical model of the problem, a hybrid approach (HPSO) that combines particle swarm optimization algorithm (PSO) and simulated annealing (SA) is proposed, and this approach is compared with the PSO algorithm without SA integration. According to the instance data of Solomon's vehicle routing problem with time window, instances with 25 customers, 50 customers and 100 customers are adapted to test the performance of PSO and HPSO algorithms with the identical parameters. The results show that both algorithms can solve the relevant cases within a reasonable time range - even the large-scale instances of 100 customer points can be solved within 1200 s. Although the PSO algorithm shows slight advantage in terms of the worst solution and standard deviation, the HPSO algorithm consistently delivers better results on the best solution than PSO for the 25, 50, and 100-customer cases. Also, the efficiency of HPSO algorithm becomes higher than that of PSO algorithm when the case size increases.

There are a few exciting research extensions in the future. One direction is the development of other solution approaches for the MCVRPTW problem, as well as the investigation on how they perform against the PSO-based algorithms developed in this paper. Also, new optimization models and solution approaches should be developed for MCVRPTW problem when more stochastic nature is taken into consideration. In addition, we plan to consider the increasing complex electronic commerce environment faced by urban distribution, such as the multi-compartment vehicle routing problem with pickup and delivery, the time-dependent multi-compartment vehicle routing problem.

Acknowledgements

This research was supported in part by the Major Program of the National Social Science Foundation of China (Grant number 15ZDB169).

References

Abdulkader, M. M. S., Gajpal, Y., & ElMekkawy, T. Y. (2015). Hybridized ant colony algorithm for the multi compartment vehicle routing problem. *Applied Soft Computing, 37*, 196–203.

Alinaghian, M., & Shokouhi, N. (2018). Multi-depot multi-compartment vehicle routing problem, solved by a hybrid adaptive large neighborhood search. *Omega, 76*, 85–99.

Archetti, C., Bianchessi, N., & Speranza, M. G. (2015). A branch-price-and-cut algorithm for the commodity constrained split delivery vehicle routing problem. *Computers & Operations Research, 64*, 1–10.

Archetti, C., Campbell, A. M., & Speranza, M. G. (2014). Multicommodity vs. single-commodity routing. *Transportation Science, 50*(2), 461–472.

Babaee, Tirkolaee, E., Alinaghian, M., Bakhshi, Sasi, M., Seyyed, & Esfahani, M. M. (2016). Solving a robust capacitated arc routing problem using a hybrid simulated annealing algorithm: A waste collection application. *Journal of Industrial Engineering and Management Studies. 3*(1), 61–76.

Birim, Ş. (2016). Vehicle routing problem with cross docking: A simulated annealing approach. *Procedia – Social and Behavioral Sciences, 235*, 149–158.

Brown, G. G., & Graves, G. W. (1981). Real-time dispatch of petroleum tank trucks. *Management Science. 27*(1), 19–32.

Caramia, M., & Guerriero, F. (2010). A milk collection problem with incompatibility constraints. *Interfaces, 40*(2), 130–143.

Chajakis, E. D., & Guignard, M. (2003). Scheduling deliveries in vehicles with multiple compartments. *Journal of Global Optimization, 26*(1), 43–78.

Coelho, L. C., & Laporte, G. (2015). Classification, models and exact algorithms for multi-compartment delivery problems. *European Journal of Operational Research, 242*(3), 854–864.

Cornillier, F., Boctor, F. F., Laporte, G., & Renaud, J. (2008b). An exact algorithm for the petrol station replenishment problem. *Journal of the Operational Research Society, 59*(5), 607–615.

Cornillier, F., Boctor, F. F., Laporte, G., & Renaud, J. (2008a). A heuristic for the multi-period petrol station replenishment problem. *European Journal of Operational Research, 191*(2), 295–305.

Cornillier, F., Laporte, G., Boctor, F. F., & Renaud, J. (2009). The petrol station replenishment problem with time windows. *Computers & Operations Research, 36*(3), 919–935.

Croes, G. A. (1958). A method for solving traveling-salesman problems. *Operations Research, 6*(6), 791–812.

Dantzig, G. B., & Ramser, J. H. (1959). The truck dispatching problem. *Management Science, 6*(1), 80–91.

Derigs, U., Gottlieb, J., Kalkoff, J., Piesche, M., Rothlauf, F., & Vogel, U. (2011). Vehicle routing with compartments: Applications, modelling and heuristics. *OR Spectrum, 33*(4), 885–914.

Eberhart, R., & Kennedy, J. (1995). *Particle swarm optimization* (pp. 1942–1948). Washington, DC: IEEE Computer Society.

Elbek, M., & Wøhlk, S. (2016). A variable neighborhood search for the multi-period collection of recyclable materials. *European Journal of Operational Research, 249*(2), 540–550.

Fagerholt, K., & Christiansen, M. (2000). A combined ship scheduling and allocation problem. *Journal of the Operational Research Society, 51*(7), 834–842.

Fallahi, El. A., Prins, C., & Calvo, R. W. (2008). A memetic algorithm and a tabu search for the multi-compartment vehicle routing problem. *Computers & Operations Research, 35*(5), 1725–1741.

Fathi, M., Rodríguez, V., Fontes, D. B., & Alvarez, M. J. (2016). A modified particle swarm optimisation algorithm to solve the part feeding problem at assembly lines. *International Journal of Production Research, 54*(3), 878–893.

Glover, F., Laguna, M., & Martí, R. (2000). Fundamentals of scatter search and path relinking. *Control and Cybernetics, 29*(3), 653–684.

Golden, B. L., Raghavan, S., & Wasil, E. A. (2008). *The vehicle routing problem: Latest advances and new challenges*. Springer Science & Business Media.

Goodson, J. C. (2015). A priori policy evaluation and cyclic-order-based simulated annealing for the multi-compartment vehicle routing problem with stochastic demands. *European Journal of Operational Research, 241*(2), 361–369.

Henke, T., Speranza, M. G., & Wäscher, G. (2015). The multi-compartment vehicle routing problem with flexible compartment sizes. *European Journal of Operational Research, 246*(3), 730–743.

Henke, T., Speranza, G., & Wäscher, G. (2017). A branch-and-cut algorithm for the multi-compartment vehicle routing problem with flexible compartment sizes. *Annals of Operations Research,* 1–18.

Hübner, A., & Ostermeier, M. (2018). A multi-compartment vehicle routing problem with loading and unloading costs. *Transportation Science*. https://doi.org/10.1016/j.cor.2016.12.023 (in press).

Hvattum, L. M., Fagerholt, K., & Armentano, V. A. (2009). Tank allocation problems in maritime bulk shipping. *Computers & Operations Research, 36*(11), 3051–3060.

Kaabi, H. (2016). Hybrid metaheuristic to solve the selective multi-compartment vehicle routing problem with time windows. *Proceedings of the second international Afro-European conference for industrial advancement AECIA 2015* (pp. 185–194). Cham.: Springer.

Kabcome, P., & Mouktonglang, T. (2015). Vehicle routing problem for multiple product types, compartments, and trips with soft time windows. *International Journal of Mathematics and Mathematical Sciences*. https://doi.org/10.1155/2015/126754 Article ID 126754.

Kandiller, L., Eliiyi, D. T., & Taşar, B. (2017). A multi-compartment vehicle routing problem for livestock feed distribution. *Operations research proceedings 2015* (pp. 149–155). Cham: Springer.

Koch, H., Henke, T., & Wäscher, G. (2016). *A genetic algorithm for the multi-compartment vehicle routing problem with flexible compartment sizes (No. 160004)*. Otto-von-Guericke University Magdeburg, Faculty of Economics and Management.

Lahyani, R., Coelho, L. C., Khemakhem, M., Laporte, G., & Semet, F. (2015b). A multi-compartment vehicle routing problem arising in the collection of olive oil in Tunisia. *Omega, 51*, 1–10.

Lahyani, R., Khemakhem, M., & Semet, F. (2015a). Rich vehicle routing problems: From a taxonomy to a definition. *European Journal of Operational Research, 241*(1), 1–14.

Laporte, G. (2009). Fifty years of vehicle routing. *Transportation Science, 43*(4), 408–416.

Marinakis, Y., Migdalas, A., & Sifaleras, A. (2017). A hybrid particle swarm optimization–variable neighborhood search algorithm for constrained shortest path problems. *European Journal of Operational Research, 261*(3), 819–834.

Melechovský, J. (2013). A variable neighborhood search for the selective multi-compartment vehicle routing problem with time windows. *Lecture Notes in Management Science, 5*, 159–166.

Mendoza, J. E., Castanier, B., Guéret, C., Medaglia, A. L., & Velasco, N. (2010). A memetic algorithm for the multi-compartment vehicle routing problem with stochastic demands. *Computers & Operations Research, 37*(11), 1886–1898.

Mendoza, J. E., Castanier, B., Guéret, C., Medaglia, A. L., & Velasco, N. (2011). Constructive heuristics for the multicompartment vehicle routing problem with stochastic demands. *Transportation Science, 45*(3), 346–363.

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., & Teller, E. (1953). Simulated annealing. *Journal of Chemical Physics, 21*, 1087–1092.

Mirzaei, S., & Wøhlk, S. (2017). Erratum to: A branch-and-price algorithm for two multi-compartment vehicle routing problems. *EURO Journal on Transportation and Logistics, 6*(2), 185–218.

Muyldermans, L., & Pang, G. (2010). On the benefits of co-collection: Experiments with a multi-compartment vehicle routing algorithm. *European Journal of Operational Research, 206*(1), 93–103.

Oppen, J., & Løkketangen, A. (2008). A tabu search approach for the livestock collection problem. *Computers & Operations Research, 35*(10), 3213–3229.

Popović, D., Vidović, M., & Radivojević, G. (2012). Variable neighborhood search heuristic for the inventory routing problem in fuel delivery. *Expert Systems with Applications, 39*(18), 13390–13398.

Rabbani, M., Farrokhi-asl, H., & Rafiei, H. (2016). A hybrid genetic algorithm for waste collection problem by heterogeneous fleet of vehicles with multiple separated compartments. *Journal of Intelligent & Fuzzy Systems, 30*(3), 1817–1830.

Reed, M., Yiannakou, A., & Evering, R. (2014). An ant colony algorithm for the multi-compartment vehicle routing problem. *Applied Soft Computing, 15*, 169–176.

Silva, R. F. R. (2016). *The multi-compartment vehicle routing problem in the collection of recyclable municipal solid waste (Doctoral dissertation)*. Portugal: Iscte Business School.
Silvestrin, P. V., & Ritt, M. (2014). *A Tabu search for the multi-compartment vehicle routing problem.* (accessed on September 10, 2018).
Silvestrin, P. V., & Ritt, M. (2017). An iterated tabu search for the multi-compartment vehicle routing problem. *Computers & Operations Research, 81*, 192–202.
Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research, 35*(2), 254–265.
Toth, P., & Vigo, D. (2014). *Vehicle routing: Problems, methods, and applications* (2nd ed.). Philadelphia, PA, USA: Society for Industrial and Applied Mathematics.
Vidović, M., Popović, D., & Ratković, B. (2014). Mixed integer and heuristics model for the inventory routing problem in fuel delivery. *International Journal of Production Economics, 147*, 593–604.
Wang, J., Jagannathan, A. K. R., Zuo, X., & Murray, C. C. (2017). Two-layer simulated annealing and tabu search heuristics for a vehicle routing problem with cross docks and split deliveries. *Computers & Industrial Engineering, 112*, 84–98.
Wang, S., Lu, Z., Wei, L., Ji, G., & Yang, J. (2016). Fitness-scaling adaptive genetic algorithm with local search for solving the multiple depot vehicle routing problem. *Simulation, 92*(7), 601–616.
Zhang, J., Yang, F., & Weng, X. (2018). An evolutionary scatter search particle swarm optimization algorithm for the vehicle routing problem with time windows. *IEEE Access, 6*, 63468–63485.